IBM® DB2® Universal Database

Руководство администратора: Производительность

Версия 7

SH43-0145-01

IBM® DB2® Universal Database

Руководство администратора: Производительность

Версия 7

SH43-0145-01

Перед тем, как использовать данный документ и продукт, описанный в нем, прочтите общие сведения под заголовком “Приложение F. Замечания” на стр. 659.

Чтобы заказать публикации, обратитесь к вашему представителю IBM или в местное отделение IBM, или позвоните по телефону 1-800-879-2755 в Соединенных Штатах или 1-800-IBM-4YOU в Канаде.

Отсылая информацию IBM, вы тем самым даете IBM неисключительное право использовать или распространять эту информацию любым способом, как фирма сочтет нужным, без каких-либо обязательств перед вами.

Содержание

Об этой книге

В трехтомном Руководстве администратора содержится информация, необходимая для пользования продуктами системы управления реляционными базами данных DB2* (RDBMS) и управления ими, в том числе:

- Информация о проектировании баз данных (том *Administration Guide: Planning*)
- Информация о реализации баз данных и управлении ими (том *Administration Guide: Implementation*)
- Информация о конфигурировании и настройке среды вашей базы данных для повышения производительности (том *Administration Guide: Performance*).

Для многих из задач, описанных в этой книге, существуют различные интерфейсы:

- **Процессор командной строки**, позволяющий обращаться к базам данных и работать с ними через графический интерфейс. При помощи этого интерфейса можно также выполнять операторы SQL и утилиты DB2. Большинство примеров в этой книге иллюстрируют использование данного интерфейса. Дополнительную информацию об использовании процессора командной строки смотрите в руководстве *Command Reference*.
- **интерфейс прикладного программирования**, позволяющий вызывать утилиты DB2 из прикладной программы. Дополнительную информацию об использовании интерфейс прикладного программирования смотрите в книге *Administrative API Reference*.
- **Центр управления**, позволяющий графически выполнять задачи управления, например, конфигурирование системы, управление каталогами, резервное копирование и восстановление системы, составление расписаний заданий и управление носителями. Центр управления позволяет выполнять также Управление репликацией для графического задания репликации данных между системами. Кроме того, Центр управления позволяет выполнять утилиты DB2 при помощи графического пользовательского интерфейса. В зависимости от вашей платформы есть различные методы вызова Центра управления. Например, можно ввести команду `db2cc` в командной строке, выбрать значок Центра управления в папке DB2 (в OS/2) или использовать панели запуска на платформах Windows. Начальные справочные сведения смотрите можно узнать, выбрав **С чего начать** в выпадающем меню **Справка** окна Центр управления. Из Центра управления можно вызвать инструменты **Наглядное объяснение** и **Монитор производительности**.

Для выполнения задач управления существуют также другие инструменты. К ним относятся:

- Центр сценариев для хранения небольших прикладных программ - сценариев. Эти сценарии могут содержать операторы SQL, команды DB2, а также команды операционной системы.
- Центр предупреждений для слежения за сообщениями других операций DB2.
- Параметры инструментов для изменения параметров Центра управления, Центра предупреждений и Репликации.
- Журнал для планирования автоматического запуска заданий.
- Центр хранилищ данных для управления объектами хранилищ.

Для кого предназначена эта книга

Эта книга адресована прежде всего администраторам баз данных, системным администраторам, администраторам защиты и системным операторам, которым нужно проектировать, реализовывать и обслуживать базу данных для обращения к ней локальных или удаленных клиентов. Она может также оказаться полезной для программистов и других пользователей, которым необходимо разобраться в управлении и работе системы управления реляционными базами данных DB2.

Структура этой книги

Эта книга содержит сведения на следующие основные темы:

Вводные замечания по производительности

- В разделе Глава 1. Элементы производительности излагаются основные понятия и особенности управления производительностью DB2 UDB и ее повышения.
- В разделе Глава 2. Обзор архитектуры и процессов излагаются основы архитектуры и процессов DB2 Universal Database.

Настройка производительности прикладных программ

- В разделе Глава 3. Особенности прикладного программирования описываются некоторые методы повышения производительности базы данных при разработке ваших прикладных программ.
- В разделе Глава 4. Факторы среды описываются некоторые методы повышения производительности базы данных при задании параметров среды базы данных.
- В разделе Глава 5. Статистика системного каталога описывается, как можно собрать статистику о ваших данных и использовать ее для обеспечения оптимальной производительности.
- В разделе Глава 6. Компилятор SQL описывается, что происходит с оператором SQL при его компиляции с помощью компилятора SQL.

- В разделе Глава 7. Возможность объяснения SQL описываются средства объяснения, позволяющие исследовать варианты доступа к вашим данным, выбранные компилятором SQL.

Настройка и конфигурирование вашей системы

- В разделе Глава 8. Производительность работы дается обзор использования памяти менеджером баз данных и рассматриваются другие особенности, влияющие на производительность во время выполнения.
- В разделе Глава 9. Использование утилиты ограничения ресурсов излагаются начальные сведения об использовании утилиты ограничения ресурсов для управления некоторыми аспектами работы с базой данных.
- В разделе Глава 10. Увеличение числа процессоров в конфигурации рассматриваются некоторые вопросы и задачи, связанные с ростом размера ваших систем баз данных.
- В разделе Глава 11. Перераспределение данных между разделами баз данных обсуждаются задачи, возникающие в среде многораздельных баз данных в связи с перераспределением данных между разделами.
- В разделе Глава 12. Измерение производительности дается обзор вопросов и способов измерения производительности.
- В разделе Глава 13. Конфигурирование DB2 обсуждаются файлы конфигурации менеджера баз данных и базы данных и значения параметров конфигурации.

Приложения

- В разделе Приложение A. Переменные реестра и среды DB2 приводятся значения реестра профиля и переменных среды.
- В разделе Приложение B. Таблицы объяснения и их определения приводится информация о таблицах, используемых средствами объяснения DB2, и о том, как создавать эти таблицы.
- В разделе Приложение C. Инструменты объяснения SQL приводится информация об средствах объяснения DB2: db2expln и dynexpln.
- В разделе Приложение D. db2exfmt - Инструмент форматирования таблицы объяснения описывается формат содержимого таблиц объяснения DB2.
- В разделе Приложение E. Использование библиотеки DB2 приводится информация о структуре библиотеки DB2, включая мастера, электронную справку, сообщения и книги.

Краткий обзор других томов Руководства администратора

Руководство администратора: Планирование

Книга *Administration Guide: Planning* посвящена разработке базы данных. В ней освещаются вопросы, связанные с логическими и физическими устройствами,

распределенными транзакциями и высокой доступностью. Отдельные главы и приложения из этого тома кратко описаны здесь:

В мире DB2 Universal Database

- В разделе ″Управление DB2 Universal Database″ содержатся предварительные сведения о DB2 Universal Database и ее обзор.

Концепции баз данных

- В разделе ″Основные концепции реляционных баз данных″ приводится обзор объектов баз данных, включая объекты восстановления, объекты хранения и системные объекты.
- В разделе ″Системы объединения″ обсуждаются системы объединения - системы управления базами данных (СУБД), которые дают возможность программам и пользователям выполнять операторы SQL, обращающиеся (в одном операторе) к нескольким СУБД или нескольким базам данных.
- В разделе ″Параллельные системы баз данных″ излагаются начальные сведения о типах параллелизма, доступных при работе с DB2.
- В разделе ″О работе с хранилищами данных″ дается обзор работы с хранилищами данных и возникающих при этом задач.
- В разделе ″О Spatial Extender″ дается понятие о модуле Spatial Extender, объясняется его назначение и обсуждаются обрабатываемые с его помощью данные.

Проектирование баз данных

- В разделе ″Логическая структура базы данных″ обсуждаются концепции логической структуры базы данных и даются указания по разработке баз данных.
- В разделе ″Физическая структура базы данных″ содержатся указания по разработке физической структуры базы данных, в том числе по вопросам хранения данных.

Распределенная обработка транзакций

- В разделе ″Проектирование распределенных баз данных″ обсуждается, как обращаться к нескольким базам данных в ходе одной транзакции.
- В разделе ″Проектирование для менеджера транзакций″ обсуждается, как использовать ваши базы данных в среде обработки распределенных транзакций, например, CICS.

Системы высокой доступности

- Раздел ″Высокая доступность и восстановление после отказов - введение″ содержит обзор поддержки высокой доступности восстановления после отказов, обеспечиваемой DB2.

Приложения

- В приложении ″Планирование перенастройки базы данных″ дается информация о перенастройке баз данных в Версию 7.
- В приложении ″Несоответствия между выпусками″ рассматриваются несовместимости, возникающие при переходе от выпуска к выпуску до Версии 7.
- В приложении ″Поддержка национальных языков (NLS)″ описывается поддержка национальных языков в DB2, включая информацию о странах, языках и кодовых страницах.

Руководство администратора: Реализация

Книга *Administration Guide: Implementation* посвящена реализации вашего проекта базы данных. Отдельные главы и приложения из этого тома кратко описаны здесь:

Управление при помощи Центра управления

- В разделе ″Управление DB2 при помощи инструментов GUI″ описываются инструменты графического пользовательского интерфейса (GUI) для управления базой данных.

Реализация вашего проекта

- В разделе ″Перед созданием базы данных″ описываются предварительные требования для создания базы данных.
- В разделе ″Создание базы данных″ рассматриваются задачи, связанные с созданием базы данных и связанных с ней объектов базы данных.
- В разделе ″Изменение базы данных″ обсуждается, что необходимо сделать перед изменением базы данных и задачи, связанные с модификацией или отбрасыванием базы данных или связанных с ней объектов базы данных.

Защита базы данных

- В разделе ″Управление доступом к базе данных″ описывается, как управлять доступом к ресурсам вашей базы данных.
- В разделе ″Аудит активности DB2″ описывается, как обнаруживать и отслеживать нежелательные или непредвиденные попытки обращения к данным.

Перемещение данных

- Раздел ″Утилиты для перемещения данных″ - это краткое введение в различные способы перемещения данных; он адресует вас к книге *Data Movement Utilities Guide and Reference*.

Восстановление

- В одностраничном введении ″Восстановление базы данных″ изложены принципы резервного копирования, восстановления и повтора транзакций для базы данных. Более подробную информацию можно найти в руководстве *Data Recovery and High Availability Guide and Reference*.

Приложения

- В разделе ″Использование служб каталога распределенной вычислительной среды (DCE)″ обсуждается использование служб каталога DCE.
- В разделе ″Обработчик пользователя для восстановления базы данных″ обсуждается, каким можно использовать обработчики пользователя с файлами журнала базы данных, и описываются некоторые примеры обработчиков пользователя.
- В разделе ″Выдача команд нескольким серверам разделов баз данных″ обсуждается использование сценариев оболочки *db2_all* и *rah* для отправки команд всем разделам в среде многораздельных баз данных.
- В разделе ″Как DB2 for Windows NT работает с защитой Windows NT″ описывается, как DB2 работает с защитой Windows NT.
- В книге ″Использование монитора производительности Windows NT″ приводится информация о регистрации DB2 с монитором производительности Windows NT и об использовании сведений о производительности.
- В разделе ″Работа с серверами разделов баз данных Windows NT или Windows 2000″ приводится информация об имеющихся утилитах для работы с серверами разделов баз данных Windows NT или Windows 2000.
- В разделе ″Конфигурирование нескольких логических узлов″ описывается, как конфигурировать несколько логических узлов в среде многораздельной базы данных.
- В разделе ″Высокоскоростная связь между узлами″ описывается, как разрешить использование Virtual Interface Architecture с DB2 Universal Database.
- В разделе ″Использование служб каталога протокола LDAP″ приводится информация о том, как пользоваться службами каталога LDAP.
- В разделе ″Расширение Центра управления″ приводится информация о том, как расширить Центр управления, добавив новые кнопки на панель инструментов, новые действия, новые определения объектов и действий.

Часть 1. Производительность -- Введение

Глава 1. Элементы производительности

Производительность - это мера того, насколько компьютерная система справляется с конкретной рабочей нагрузкой. Производительность оценивается по одному или нескольким показателям: времени ответа системы, пропускной способности и доступности. На нее влияют:

- Доступные ресурсы
- Эффективность применения и совместного использования этих ресурсов.

Как правило, если вы хотите улучшить отдачу от затрат на систему, следует провести настройку производительности. При этом могут решаться следующие задачи:

- Повышение объема или сложности рабочей нагрузки без увеличения стоимости обработки. (Например, увеличение рабочей нагрузки без приобретения новых аппаратных средств или роста времени работы процессора.)
- Уменьшение времени ответа системы или повышение пропускной способности без увеличения стоимости обработки.
- Снижение стоимости обработки без отрицательного влияния на обслуживание пользователей.

Перевод показателей производительности с технических на экономические меры - задача непростая. Настройка производительности, естественно, стоит денег (это время работы сотрудников и процессорное время), поэтому прежде, чем принять проект по настройке производительности, взвесьте все затраты и возможные выгоды. К материальным выгодам относится:

- Более эффективное использование ресурсов
- Возможность добавить в систему больше пользователей.

Другие выгоды, например, большая удовлетворенность пользователей в связи с сокращением времени ответов системы, нематериальны. Все эти выгоды будут рассмотрены.

В системе DB2 есть встроенные мастера, которые помогут выполнить некоторые связанные с производительностью задачи администраторов. Обычно, решая такие задачи, вы тратите мало времени, и можете добиться значительного улучшения производительности. Мастера помогут выполнить каждую задачу, проведя вас последовательно по шагам необходимых действий. Эти мастера доступны в Центре управления и в Ассистенте конфигурирования клиента.

Мастер по настройке производительности помогает настроить производительность базы данных изменением параметров конфигурации в соответствии с вашими требованиями. Этот мастер и, в какой-то степени, мастер по созданию баз данных помогут улучшить производительность базы данных. Другие мастера помогают улучшить производительность для отдельных таблиц и операций общего доступа к данным. К таким мастерам относится мастер по созданию таблиц, мастер по индексам и мастер конфигурирования многоузлового изменения. Эти мастера можно вызвать из Центра управления, щелкнув правой кнопкой мыши по объекту.

Рекомендации по настройке

Следующие рекомендации должны вам помочь разработать общий подход по настройке производительности.

Помните закон снижения эффекта: Самая большая выгода в производительности обычно получается в результате начальных действий. Дальнейшие изменения обычно приводят к меньшим выгодам при больших затратах усилий.

Не выполняйте настройку ради самой настройки: настройка нужна для того, чтобы уменьшить влияние реальных ограничений. Если вы настроите ресурсы, которые не являются первичной причиной низкой производительности, это почти или совсем не повлияет на время ответа, пока вы не затронете главные ограничения; фактически это может даже затруднить последующий процесс настройки. Если есть потенциальная возможность значительно улучшить производительность, она заключается в улучшении использования тех ресурсов, который являются определяющими факторами времени ответа.

Рассматривайте систему в целом: Никогда нельзя настраивать один параметр или часть системы в изоляции от остальных. Перед какой-либо корректировкой рассмотрите ее влияние на всю систему.

За один раз изменяйте один параметр: Не изменяйте несколько параметров настройки производительности одновременно. Если даже вы уверены, что все изменения будут выгодны, у вас не будет способа оценить вклад каждого изменения. Кроме того, вы не сможете эффективно оценить замену, которую сделали, изменив несколько параметров одновременно. Каждый раз при корректировке параметра с целью улучшения в одной области вы почти всегда затрагиваете минимум одну другую область, которую, возможно, упустили из вида. Изменение единственного параметра за раз позволяет определить, достигает ли изменение требуемой цели.

Измеряйте и реконфигурируйте по уровням: По тем же причинам, по которым вы изменяете один параметр за раз, за один раз настраивайте только один уровень системы. Для руководства можно использовать следующий список уровней в системе:

- Аппаратные средства
- Операционная система
- Сервер и реквестер прикладных программ
- Менеджер баз данных
- Операторы SQL
- Прикладные программы

Проверьте ошибки аппаратных средств и программного обеспечения: Некоторые ошибки производительности можно исправить, воспользовавшись служебными программами для аппаратных средств, для программного обеспечения либо для того и другого. Не тратьте лишнее время на мониторинг и настройку системы там, где простое использование служебной программы может сделать это ненужным.

Поймите суть проблемы перед тем, как менять аппаратные средства: Даже если кажется, что увеличение объема памяти или мощности процессора могут сразу же улучшить производительность, потратьте время, чтобы понять, где находятся узкие места. Вы можете истратить деньги на дополнительную дисковую память, а потом понять, что для ее эксплуатации не хватает мощности процессора или каналов.

Перед настройкой предусмотрите процедуры возврата к прежним параметрам: Как отмечалось раньше, иногда настройка может привести к неожиданным результатам. Если настройка приведет к снижению производительности, следует вернуться к прежнему проверенному варианту. Если сохранить прежнюю настройку так, чтобы ее легко можно было восстановить, убрать неудачные значения станет проще.

Дисковая память

Мы уже говорили, что оборудование вашей системы может влиять на ее производительность. В качестве примера влияния аппаратных средств на производительность рассмотрим некоторые факторы, связанные с дисковой памятью.

Управление дисковой памятью влияет на производительность по четырем аспектам:

- **Разделение дисковой памяти**:

Как вы распределите ограниченные объемы памяти между индексами и данными, табличными пространствами, а также пулами буферов, в большой степени определит производительность работы в различных ситуациях.

- **Лишняя память**:

  Лишняя память может не влиять на производительность использующей ее системы, но этим ресурсом можно воспользоваться, чтобы улучшить производительность в другом месте.

- **Распределение дискового ввода/вывода**:

  Правильность балансирования требований дискового ввода/вывода по нескольким устройствам дисковой памяти и контроллерам может повлиять на скорость получения информации с дисков менеджером баз данных.

- **Нехватка памяти**:

  Достижение предела доступной памяти может привести к снижению общей производительности.

Процесс улучшения производительности

Для улучшения производительности любой системы воспользуйтесь следующим процессом:

1. Определите целевые параметры производительности.
2. Установите индикаторы производительности.
3. Разработайте план мониторинга производительности.
4. Выполните этот план.
5. Проанализируйте результаты измерений, чтобы определить, достигнуты ли ваши цели. Если цели достигнуты, попробуйте сократить число выполняемых измерений, так как сам мониторинг производительности тоже расходует системные ресурсы. В противном случае выполните следующий шаг.
6. Определите главные ограничения в системе.
7. Определите, чем можно пожертвовать и какие ресурсы смогут принять на себя дополнительную нагрузку. (Почти при любой настройке приходится выбирать между системными ресурсам и различными элементами производительности.)
8. Скорректируйте конфигурацию системы. Если возможно изменение нескольких опций настройки, изменяйте их по одной за раз. Если никаких возможностей улучшения ни на одном из уровней нет, это значит, что достигнут предел для ваших ресурсов, и необходимо менять аппаратные средства.
9. Вернитесь к шагу 4 выше и продолжайте слежение за системой.

Регулярно или после существенного изменения системы или рабочей нагрузки:

- Вернитесь к шагу 1.

- Снова проверьте целевые параметры и индикаторы.
- Совершенствуйте свою стратегию мониторинга и настройки.

Чего можно достичь настройкой системы?

Существуют пределы того, насколько можно улучшить эффективность системы. Подсчитайте, сколько времени и денег вы должны потратить на улучшение производительности системы, и насколько затраты дополнительного времени и денег помогут пользователям этой системы.

Система может работать правильно и без настройки вообще, но такая система, вероятно, не сможет полностью реализовать свои потенциальные возможности. Каждая база данных уникальна. Как только вы разработаете собственную базу данных и программы для ее использования, исследуйте доступные параметры настройки и научитесь подстраивать свои параметры под конкретную ситуацию. При некоторых обстоятельствах польза от настройки системы будет незначительной, однако в большинстве случаев выгода может оказаться существенной.

Из Центра управления доступны мастера, оказывающие помощь в настройке параметров базы данных. Мастер по настройке производительности можно вызвать, щелкнув правой кнопкой мыши по базе данных, которую вы хотите настроить из Центра управления.

Если при работе системы возникают ситуации с критической производительностью, вероятно, что настройка окажется эффективной. Если производительность близка к максимально возможной, и вы увеличите число пользователей системы примерно на десять процентов, время ответа, скорее всего, возрастет намного больше, чем на 10 процентов. В этой ситуации нужно определить, как с помощью настройки системы компенсировать такое снижение производительности. Однако существует предел, за которым настройка не поможет. В этот момент следует пересмотреть свои цели и ожидаемые результаты для вашей среды. Другой вариант - изменить системную среду, рассмотрев такие возможности, как увеличение дисковой памяти, повышение скорости процессора, применение дополнительных процессоров, увеличение основной памяти, повышение скорости каналов связи, а также сочетание этих изменений.

Менее формальный подход

Если у вас недостаточно времени, чтобы определить критические параметры производительности и провести всесторонний мониторинг и настройку, узнайте мнение о производительности у ваших пользователей. Выясните, есть ли у них связанные с производительностью проблемы. Можно диагностировать проблему или определить, с чего начинать ее поиск, задав несколько простых вопросов. Например, можно провести следующий опрос:

- Что вы понимаете под “медленным ответом”? Медленнее ожидаемого на десять процентов или медленнее в десять раз?
- Когда вы заметили эти ошибки? Только недавно, или они были всегда?
- Знаете ли вы других пользователей, кто жалуется на эту проблему? Жалуется один пользователь, двое или целая группа?
- (Если трудности испытывает целая группа пользователей, подключены ли они к одной и той же локальной сети?)
- Связаны ли проблемы, которые вы испытываете, с определенной транзакцией или прикладной программой?
- Проблемы проявляются в определенное время, например, в обеденный перерыв, или постоянно?

Подведем итоги

Важно понимать базовую архитектуру DB2, поскольку знание основных концепций и процессов поможет вам решить проблемы производительности. В следующей главе дается начальное представление об архитектуре хранения, управлении данными, модели обработки и модели памяти. Дополнительную информацию смотрите в разделе “Глава 2. Обзор архитектуры и процессов” на стр. 11.

Настройка производительности прикладных программ рассматривается в темах, где говорится о производительности программ и их взаимодействии с базой данных. Рассматриваются темы, относящиеся непосредственно к программам: параллелизм, блокировка, классы оптимизации, управление наборами результатов запросов, блокирование строк, использование составных операторов SQL. Кроме того, кратко обсуждаются: преобразование символов (как относящееся к производительности программ), хранимые процедуры, активация баз данных и преимущества параллельной обработки. Дополнительную информацию смотрите в разделе “Глава 3. Особенности прикладного программирования” на стр. 43.

Темы, относящиеся к оптимизации запросов: параметры конфигурации, влияющие на оптимизацию запросов, воздействие на оптимизацию запросов группировок узлов и табличных пространств, а также существенное влияние, которое могут оказывать на оптимизацию запросов индексы. Дополнительную информацию смотрите в разделе “Глава 4. Факторы среды” на стр. 95.

Статистика системного каталога оказывает существенное влияние на эффективность обращения программ к данным. Со статистикой связаны следующие темы: утилита RUNSTATS, статистика распределения, статистика индексов и те статистические показатели, которые могут изменять пользователи. Дополнительную информацию смотрите в разделе “Глава 5. Статистика системного каталога” на стр. 117.

Компилятор SQL берет каждую программу и определят для нее наилучший план доступа. В пределах программы проводится оценка каждого запроса; к нему может быть применено несколько различных операций, разработанных для наиболее ясного определения цели запроса. Затем рассматриваются различные методы доступа (просмотры и объединения) для определения самого быстрого способа получения данных, запрашиваемых этим запросом. Кроме того, рассматривается влияние параллелизма. Дополнительную информацию смотрите в разделе “Глава 6. Компилятор SQL” на стр. 155.

В продукте DB2 есть различные инструменты, помогающие понять, что происходит с запросами программы. Эти инструменты объясняют, что влияет на производительность программ. Дополнительную информацию смотрите в разделе “Глава 7. Возможность объяснения SQL” на стр. 225.

Кроме настройки отдельных программ, следует рассмотреть и производительность базы данных, где эти программы запускаются. Производительность базы данных большей частью определяется эффективностью использования памяти. В следующих темах, затрагивающих вопросы памяти, обращается внимание на производительность: пулы буферов, предварительное чтение данных, возможности параллельной сортировки ввода/вывода, необходимость реорганизации данных в таблицах и концепция агентов базы данных. Дополнительную информацию смотрите в разделе “Глава 8. Производительность работы” на стр. 253.

Для управления способом использования программами базы данных можно установить утилиту ограничения ресурсов. Дополнительную информацию смотрите в разделе “Глава 9. Использование утилиты ограничения ресурсов” на стр. 299.

Для улучшения производительности базы данных можно увеличить число процессоров и разделов базы данных. Дополнительную информацию смотрите в разделе “Глава 10. Увеличение числа процессоров в конфигурации” на стр. 315.

Увеличив число разделов базы данных, желательно проверить правильность распределения данных по этим разделам. Дополнительную информацию смотрите в разделе “Глава 11. Перераспределение данных между разделами баз данных” на стр. 327.

Для определения эффективности работы базы данных можно провести тестовое измерение производительности. Методика измерения производительности, подготовки к измерению производительности, создание программы измерения производительности и запуск тестов измерения производительности описаны в темах, посвященных производительности. Дополнительную информацию смотрите в разделе “Глава 12. Измерение производительности” на стр. 335.

Обширный набор параметров менеджера баз данных и конфигурации базы данных представлен в разделе “Глава 13. Конфигурирование DB2” на стр. 349.

С темами производительности связана дополнительная информация. Эта информация приводится в приложениях:

- “Приложение A. Переменные реестра и среды DB2” на стр. 521
- “Приложение B. Таблицы объяснения и их определения” на стр. 557
- “Приложение C. Инструменты объяснения SQL” на стр. 591
- “Приложение D. db2exfmt - Инструмент форматирования таблицы объяснения” на стр. 637

Глава 2. Обзор архитектуры и процессов

Для работы над производительностью операций с базами данных в DB2 требуются некоторые элементарные знания об основных понятиях, касающихся архитектуры и процессов DB2. В этой главе даются достаточные сведения для понимания того, как работает DB2 Universal Database. Хотя подробности некоторых из затронутых здесь тем приведены в следующих главах, материал этой главы создает контекст для понимания дальнейшего.

На первом рисунке показана общая схема архитектуры и процессов в DB2 UDB.

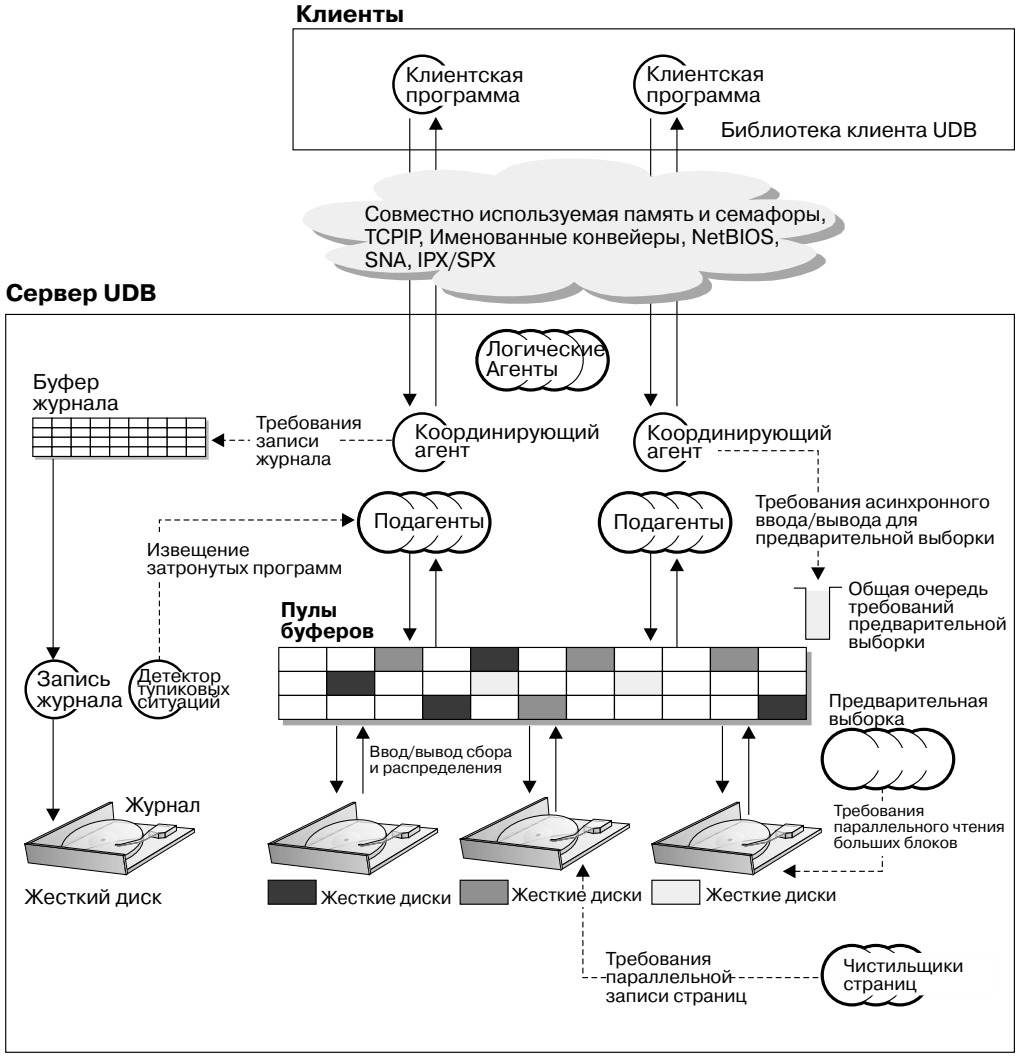


Рисунок 1. Общая схема архитектуры и процессов

Со стороны клиента находятся локальные и/или удаленные прикладные программы, связанные с клиентской библиотекой DB2 Universal Database.

Между клиентами и сервером DB2 Universal Database показано “облако”, изображающее средства связи между локальными или удаленными клиентами и сервером. Локальные клиенты связываются при помощи совместной памяти и семафоров; удаленные клиенты используют протоколы, например, NPIPE (Named Pipes - именованные конвейеры), TCP/IP, NetBIOS, IPX/SPX и SNA.

Со стороны сервера работой управляют **управляемые единицы ядра** (EDU - engine dispatchable unit). На всех рисунках в этой главе EDU изображены кружками или группами кружков. На платформах на основе Windows и в OS/2 EDU реализуются как потоки (все в рамках одного процесса), а в UNIX - как процессы. Самый распространенный тип EDU - агенты DB2. Эти EDU выполняют всю основную обработку SQL для прикладных программ. Другие примеры EDU - программы предварительной выборки DB2 и чистильщики страниц, отвечающие за разного рода операции ввода/вывода. Дополнительную информацию смотрите в разделе “Агенты базы данных” на стр. 287.

Каждой клиентской программе назначается уникальная EDU, называемая “агентом координатора”, которая координирует обработку для прикладной программы и поддерживает с ней связь. Кроме того, целый набор подагентов может быть назначен для работы по запросам клиентской программы. Если сервер расположен на компьютере с несколькими процессорами (например, в симметрической многопроцессорной среде), назначение нескольких подагентов позволяет использовать эти процессоры для выполнения запросов прикладной программы.

Все агенты и подагенты управляются при помощи алгоритма пула, который минимизирует создание и/или уничтожение EDU.

Пул буферов - это область памяти, в которую страницы данных пользовательских таблиц, индексов и каталогов помещаются для временного хранения, и возможно, для модификации. Пул буферов существенно повышает производительность всей базы данных, поскольку доступ к данным из памяти намного быстрее, чем с диска. Чем больше нужных программам данных присутствует в пуле буферов, тем больше выигрыш по времени обращения к этим данным по сравнению с затратами времени на получение тех же данных из дисковой памяти. Дополнительную информацию смотрите в разделе “Управление пулом буферов базы данных” на стр. 261.

От конфигурации пула буферов, а также от EDU предварительной выборки и очистки страниц зависит скорость обращения к данным и, как следствие, доступность данных, нужных прикладным программам.

Предварительная выборка - это извлечение данных с диска и помещение их в пул буферов до того, как они потребуются прикладным программам. Если бы не было предварительной выборки, прикладным программам, требующим просмотра большого объема данных, пришлось бы ожидать перемещения данных с диска в пул буферов. Агенты прикладной программы посылают асинхронные требования предварительного чтения в общую очередь предварительной выборки. По мере появления доступных единиц управления предварительной выборки они реализуют эти требования, перенося запрошенные страницы с диска в пул буферов при помощи чтения крупными блоками или выборочного чтения. Хранение информации на нескольких дисках

позволяет распределять данные. Такое распределение позволяет агентам предварительной выборки одновременно читать данные с нескольких дисков. Дополнительные сведения смотрите в разделах “Предварительная выборка данных в пул буферов” на стр. 269 и “Настройка серверов ввода/вывода для предварительной выборки и параллельного ввода/вывода” на стр. 273.

Предварительная выборка служит для переноса данных с диска в пул буферов. Очистка страниц служит для перемещения данных из пула буферов обратно на диск.

Чистильщики страниц - это фоновые EDU, независимые от агентов прикладных программ, которые находят и записывают на диск страницы, которые больше не нужны в пуле буферов. Работа чистильщиков страниц обеспечивают свободное место в пуле буферов для извлечения страниц агентами предварительной выборки.

Если бы не было независимых EDU предварительной выборки и очистки страниц, агентам прикладных программ пришлось бы самим выполнять всю работу по чтению в пул буферов и записи на диски.

При одновременной работе нескольких прикладных программ с одной базой данных эти программы могут попасть в “тупиковую ситуацию”. Тупиковая ситуация изображена на следующем рисунке.

Понятие тупиковой ситуации

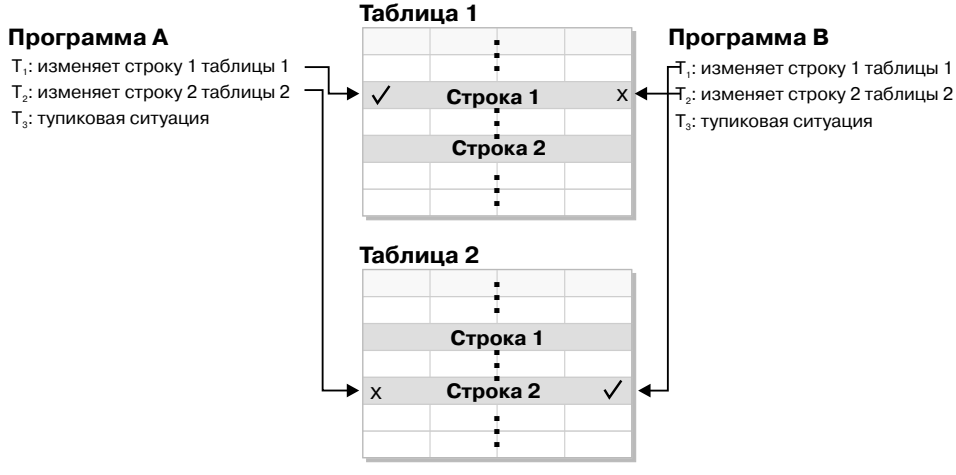


Рисунок 2. Детектор тупиковых ситуаций

“Тупиковая ситуация” означает, что не менее двух прикладных программ ожидают друг от друга освобождения блокировки данных. Каждая из ожидающих программ удерживает своей блокировкой данные, нужные другим

программам. Заблокированные данные нужны одной или нескольким другим программам, которые, в свою очередь, сами удерживают данные, нужные этой программе. Взаимное ожидание, пока другая программа освободит удерживаемую блокировку данных, приводит к тупиковой ситуации: каждая прикладная программа может до бесконечности ждать, что не она, а "другая" прикладная программа освободит блокировку удерживаемых данных. Но другая прикладная программа не освободит блокировку нужных ей данных добровольно. Для выхода из тупиковой ситуации нужно вмешательство особого процесса.

Как следует из его названия, детектор тупиковых ситуаций выявляет агенты, ожидающие блокировок. Детектор тупиковых ситуаций произвольно выбирает одну из прикладных программ, попавших в тупиковую ситуацию, и выбранная программа в "добровольно-принудительном" порядке освобождает удерживаемые ею блокировки. Освобождение блокировок этой программой делает доступными данные, требуемые остальным ожидающим программам. Находившиеся в состоянии ожидания программы получают свободу действий в отношении нужных данных и могут завершить свои операции над базой данных.

Изменения страниц данных в пуле буферов регистрируются в журнале. Существует буфер журнала, который ассоциирован с EDU ведения журнала. Агенты, делающие записи в базе данных, изменяют соответствующие страницы в пуле буферов и делают записи в журнале. Журнальная запись содержит информацию, необходимую для повтора или отмены изменения. Для лучшей производительности ни страница из пула буферов, ни запись в буфере журнала не переносятся на диск сразу. EDU журнала и менеджер пула буферов сообща реализуют протокол WAL (write ahead logging - предварительная запись в журнал), который не допускает запись страницы данных на диск, пока не будет сделана соответствующая запись в журнале. Протокол WAL обеспечивает наличие в журнале информации, позволяющей после аварии возобновить работу и восстановить согласованность базы данных. Если на диск оказалось записано непринятое изменение страницы, аварийное восстановление использует информацию отмены из соответствующей записи в журнале и отменяет изменение. Если принятое изменение не записано на диск, аварийное восстановление использует информацию повтора из соответствующей записи в журнале и повторяет изменение.

Примечание: Оператор COMMIT заносит все журнальные записи транзакции на диск, если они еще не были сохранены.

Архитектура хранения информации

Описывая архитектуру памяти, мы рассмотрим:

Каталог баз данных

При создании базы данных информация о ней, включая информацию по умолчанию, помещается в некоторый каталог. Место для создания структуры каталогов выбирается, исходя из информации, которую вы задаете в команде CREATE DATABASE. Если вы не укажете каталог или диск при создании базы данных, используется каталог по умолчанию.

Рекомендуется явно объявлять, где вы хотите создать базу данных.

В каталоге, указанном вами в команде CREATE DATABASE, создается подкаталог с именем экземпляра. Это позволяет создавать в разных экземплярах базы данных, задавая при этом один и тот же каталог. В подкаталоге с именем экземпляра создается подкаталог с именем NODE0000. Этот подкаталог используется, чтобы различать разделы в среде баз данных с несколькими логическими разделами. В подкаталоге узла создается подкаталог SQL00001. Этот подкаталог получает имя маркера базы данных и представляет создаваемую базу данных. Он служит также, чтобы различать базы данных, созданные в этом экземпляре, внутри каталога, заданного в команде CREATE DATABASE.

Получается примерно такая структура каталогов:

```
<ваш_каталог>/<ваш_экземпляр>/NODE0000/SQL00001/
```

Каталог базы данных будет содержать несколько файлов, которые все созданы командой CREATE DATABASE. Информация о пуле буферов содержится в файлах SQLBP.1 и SQLBP.2. Информация о табличном пространстве содержится в файлах SQLSPCS.1 и SQLSPCS.2. В обоих случаях два одинаковых файла нужны для резервного копирования информации.

Информация о конфигурации базы данных содержатся в файле SQLDBCON. Вы можете читать файл хронологии DB2RHIST.ASC и его резервную копию DB2RHIST.BAK; они содержат информацию о прошлых резервных копированиях, восстановлениях, загрузке таблиц, реорганизации таблиц, изменениях табличного пространства и других изменениях в базе данных.

Файл управления журналами SQLOGCTL.LFH содержит информацию об активных журналах. При восстановлении информация этого файла помогает определить, с какого момента зафиксированной в журналах истории начать восстановление. Подкаталог SQLOGDIR содержит действующие файлы журналов.

Примечание: Следует задать для подкаталога журналов диск, отличный от дисков ваших данных. В таком случае проблемы с каким-либо диском затронут либо только данные, либо только журналы, но не то и другое одновременно. Кроме того, это дает и существенную прибавку производительности, так как файлы журналов и

контейнеры базы данных не конкурируют за перемещение одних и тех же головок дисков. Положение подкаталога журналов можно изменить при помощи параметра конфигурации базы данных *newlogpath*.

Создаются подкаталоги SQLT*, которые содержат табличные пространства SMS (System Managed Space - пространство, управляемое системой) по умолчанию; эти табличные пространства нужны операционной базе данных. По умолчанию создается три табличных пространства:

- Подкаталог SQLT0000.0 содержит табличное пространство каталога с таблицами системного каталога.
- Подкаталог SQLT0001.0 содержит временное табличное пространство по умолчанию.
- Подкаталог SQLT0002.0 содержит табличное пространство пользовательских данных по умолчанию.

Изучая табличные пространства, вы читали и о “контейнерах”. В случае табличных пространств SMS контейнерами служат каталоги операционной системы.

В каждом подкаталоге или контейнере создается файл SQLTAG.NAM. Этот файл помечает подкаталог как используемый, чтобы в дальнейшем при создании табличных пространств не было попыток использовать эти подкаталоги. В подкаталогах контейнеров создаются также другие файлы, расширения которых зависят от типа хранимых в них данных. Используются следующие расширения:

- SQL*.DAT (содержит данные таблицы, кроме данных типа типа long)
- SQL*.LF (содержит данные типа LONG VARCHAR или LONG VARGRAPHIC)
- SQL*.LB (содержит данные BLOB, CLOB или DBCLOB)
- SQL*.LBA (содержит информацию о выделении памяти и свободном пространстве файлов SQL*.LB)
- SQL*.INX (содержит данные таблиц индексов)
- SQL*.DTR (содержит временные данные для реорганизации файла SQL*.DAT)
- SQL*.LFR (содержит временные данные для реорганизации файла SQL*.LF)
- SQL*.RLB (содержит временные данные для реорганизации файла SQL*.LB)
- SQL*.RBA (содержит временные данные для реорганизации файла SQL*.LBA)

Табличные пространства

Поддерживается два типа табличных пространств: SMS (System Managed Space - пространство, управляемое системой) и DMS (Database Managed Space - пространство, управляемое базой данных). Характеристики этих пространств различны, что позволяет применять их в разных средах. Дополнительную информацию о разработке и выборе табличных пространств смотрите в разделе *Administration Guide: Planning*.

Табличные пространства SMS

Табличные пространства SMS (System Managed Space - пространство, управляемое системой) хранят данные в файлах операционной системы. Данные в табличных пространствах распределены в виде экстентов по всем контейнерам системы. **Экстент** - это группа последовательных страниц, число которых задается для конкретной базы данных. Каждой таблице в табличном пространстве дается свое имя файла, которое используется всеми контейнерами. Расширение файла обозначает тип данных, хранимых в файле. Начальный экстент для каждой таблицы попадает в тот или иной контейнер в порядке “ротации”. Благодаря этому требования пространства равномерно распределяются по всем контейнерам табличного пространства. Это немаловажно при большом числе малых таблиц.

Выделение пространства производится по мере возникновения требований дополнительного пространства. По умолчанию за один раз отводится одна страница.

Табличные пространства DMS

В случае табличных пространств DMS (Database Managed Space - пространство, управляемое базой данных) пространством памяти управляет менеджер баз данных. При определении табличного пространства DMS выбирается список устройств или файлов, выделяемых под табличное пространство. Пространство в этих устройствах или файлах управляется менеджером баз данных DB2. Как и в табличных пространствах и контейнерах SMS, табличные пространства DMS и менеджер баз данных используют разделение по экстентам для равномерности распределения данных по всем контейнерам.

В отличие от табличных пространств SMS, в табличных пространствах DMS пространство выделяется при создании табличного пространства, а не по мере надобности.

Кроме того, размещение данных может происходить по-разному в этих типах табличных пространств. Например, рассмотрим потребность в эффективном просмотре таблицы; в этом случае важно, чтобы страницы в экстенте физически шли друг за другом. В случае SMS за физическое расположение логических страниц файла отвечает файловая система операционной системы. Страницы не обязательно отводятся последовательно - это зависит от уровня активности других процессов в файловой системе и от алгоритма, используемого для выбора места. Напротив, в случае DMS менеджер баз данных может обеспечить физически последовательное расположение страниц, поскольку он взаимодействует с диском напрямую.

Есть одно исключение из этого общего утверждения относительно последовательного размещения страниц в памяти. Существуют два варианта контейнеров при работе с табличными пространствами DMS: непосредственные устройства и файлы. При работе с файловыми контейнерами менеджер баз

данных отводит весь контейнер в момент создания табличного пространства. Вследствие этого начального отведения всего табличного пространства физическое размещение в типичном случае (хотя и не обязательно) оказывается непрерывным, несмотря на то, что им управляет файловая система. При работе с контейнерами непосредственных устройств менеджер баз данных берет на себя управление всем устройством и гарантирует последовательность страниц экстента.

В отличие от табличных пространств SMS, контейнеры табличного пространства DMS не обязательно близки по своей емкости. Рекомендуется, однако, делать контейнеры равными или почти равными по емкости. Кроме того, если контейнер заполняется, свободное пространство в остальных контейнерах можно использовать в табличном пространстве DMS.

При работе с табличными пространствами DMS есть смысл назначить каждому контейнеру свой диск. Это позволит увеличить емкость табличного пространства и воспользоваться преимуществами параллельных операций ввода/вывода.

Оператор CREATE TABLESPACE создает новое табличное пространство в пределах базы данных, назначает контейнеры для табличного пространства и записывает определение и атрибуты табличного пространства в каталог. При создании табличного пространства, в частности, определяется размер экстента. Размер экстента - это единица отведения пространства в пределах табличного пространства. По существу, экстент - это несколько последовательных страниц. Размер экстента - это число последовательных страниц. Страницы одного экстента может использовать только одна таблица (или другой объект, например, индекс). Все объекты (таблицы, индексы и другие), созданные в табличном пространстве, размещены в экстентах на карте логических адресов табличного пространства. Экстент не может одновременно принадлежать нескольким объектам. Выделение экстентов управляется через SMP (Space Map Pages - страницы карт пространства).

Первый экстент на карте логических адресов табличного пространства - это заголовок табличного пространства, содержащий служебную информацию. Второй экстент - это первый экстент SMP для этого пространства. Экстенты SMP распределены по табличному пространству через регулярные интервалы. Каждый экстент SMP - это просто битовая карта всех экстентов от текущего экстента SMP до следующего экстента SMP. На битовой карте записывается, какие экстенты из этой группы заняты.

Следующий экстент, идущий за SMP - это таблица объектов табличного пространства. Таблица объектов - это служебная таблица для учета, какие пользовательские объекты существуют в табличном пространстве и где находятся их первые экстенты EMP (extent map pages - страницы карт экстентов). У каждого объекта есть свои собственные EMP, обеспечивающие

картографирование всех страниц объекта, которые нанесены на карту логических адресов табличного пространства.

На следующем рисунке показана карта логических адресов для табличного пространства DMS.

Рисунок 3. Табличные пространства DMS

Таблица объектов - это служебная реляционная таблица, которая ставит в соответствие идентификатору объекта положение первого экстента EMP таблицы. Через экстент EMP можно прямо или косвенно получит любой экстент объекта. Каждая страница EMP содержит массив записей. Каждая запись отображает номер экстента для объекта в номер страницы табличного пространства, где находится экстент объекта. Прямые записи EMP прямо отображают адреса относительно объекта в адреса относительно табличного пространства. Последняя страница EMP в первом экстенте EMP содержит косвенные записи. Косвенные записи EMP отображают в страницы EMP, которые затем отображают в страницы объектов. Последние 16 записей в последней странице EMP в первом экстенте EMP содержат косвенные записи второго порядка.

Экстенты из карты логических адресов табличного пространства путем ротации распределяются по контейнерам, связанным с табличным пространством.

Сравнение табличных пространств SMS и DMS

Сравнивая табличные пространства SMS и DMS, заметим, что табличные пространства SMS хорошо подходят для общих целей. Табличные пространства SMS обеспечивают хорошую производительность с очень низкими управленческими расходами. Табличные пространства DMS позволяют ценой некоторых усилий достичь максимальной производительности. Контейнеры устройств обеспечивают высшую производительность, поскольку при переносе данных при помощи файловых контейнеров или табличных пространств SMS может производиться двойная буферизация. (Двойная буферизация может происходить, если буферизация данных сначала выполняется на уровне менеджера баз данных, а затем еще раз - на уровне файловой системы.)

Управление данными

После создания базы данных, табличного пространства и таблицы и помещения данных в таблицу вам будет интересно знать, как организована таблица и как используются индексы при получении данных таблицы.

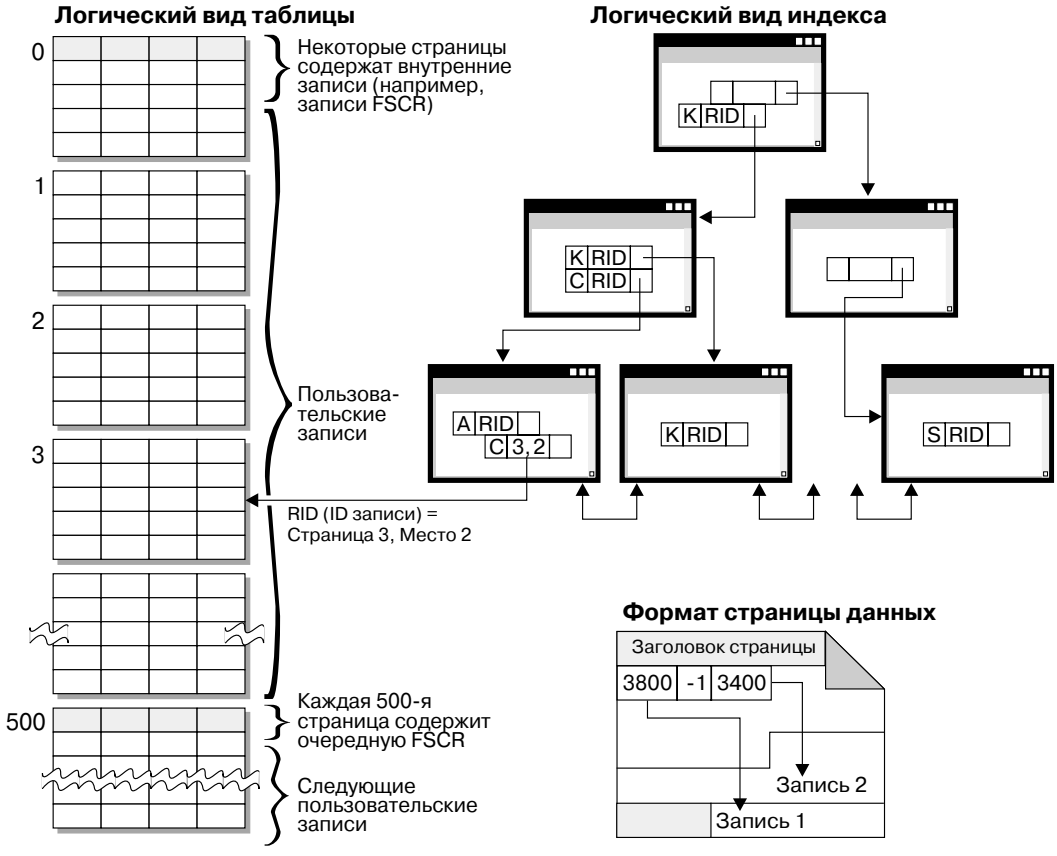


Рисунок 4. Таблицы, записи и индексы

Логически данные таблицы организованы как список страниц данных. Эти страницы данных логически сгруппированы в зависимости от размера экстента табличного пространства. Например, если размер экстента равен четырем, в первый экстент войдут страницы с нулевой по третью, во второй экстент - страницы с четвертой по седьмую, и так далее.

Число записей на каждой странице данных может меняться в зависимости от страницы данных и размера записей. Максимальное число записей на одной странице - 255. Большинство страниц содержит только пользовательские записи. Однако небольшое число страниц содержит служебные записи, которые DB2 использует для управления таблицей. Например, на каждой 500-й странице данных есть запись FSCR (Free Space Control Record - запись управления свободным пространством). В этих записях указывается, сколько свободного пространства для новых записей существует на каждой из следующих 500 страниц данных (до следующей записи FSCR). Это доступное свободное пространство используется при вставке записей в таблицу.

Логически страницы индекса организованы как B-дерево, которое позволяет быстро находить среди данных таблицы записи с заданном значением ключа. Число элементов на странице индекса не фиксировано, оно зависит от размера ключа. Для таблиц в табличных пространствах DMS идентификаторы записей (RID) на страницах индекса используют номера страниц относительно табличного пространства, а не номера страниц относительно объекта. Это позволяет при **просмотре индекса** обращаться прямо к страницам данных, не пользуясь отображением через страницы EMP (Extent Map page - страница карт экстентов).

Ниже описан формат страниц данных. Каждая страница данных начинается с заголовка страницы. После заголовка страницы идет каталог гнезда. Каждый элемент в каталоге гнезда соответствует своей записи на странице. Элемент представляет собой смещение в байтах на странице данных, указывающее начало записи. Элементы со значением минус один (-1) соответствуют удаленным записям.

Идентификаторы записей и страницы

Идентификатор записи (RID) - это трехбайтный номер страницы, за которым следует однобайтовый номер гнезда. После того, как при помощи индекса получены идентификаторы RID, при помощи каждого RID можно получить номер нужной страницы данных и гнезда на этой странице. Содержимое гнезда представляет собой байтовое смещение в пределах страницы до начала искомой записи. После того, как записи присвоен RID, он не изменяется до реорганизации таблицы.

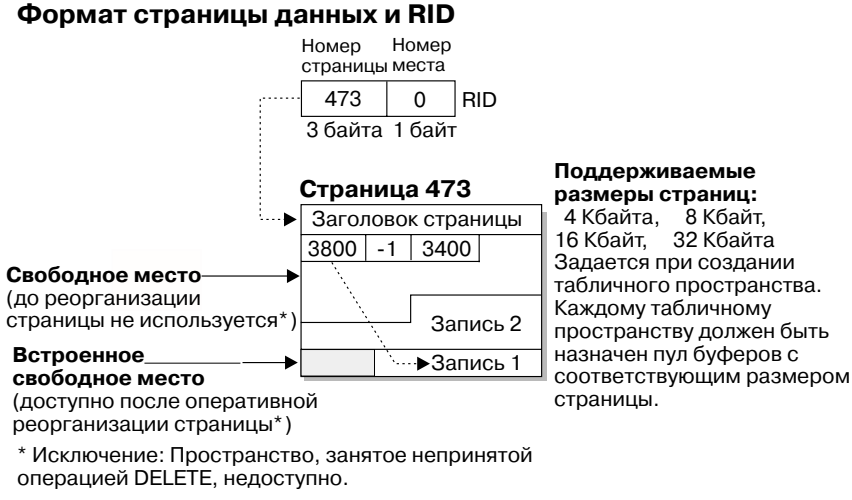


Рисунок 5. Страница данных и формат RID

Когда таблица реорганизуется, скрытое свободное пространство, остающееся на странице данных после удаления записи, преобразуется в доступное свободное

пространство. Все RID переопределяются с учетом перемещения данных по странице данных, необходимого для вовлечения в оборот свободного пространства.

DB2 поддерживает разные размеры страниц. Большие размеры страниц полезны, когда рабочая нагрузка предполагает частый доступ к последовательным строкам. Например, последовательный доступ используется для прикладных программ поддержки решений и в случае интенсивного использования временных таблиц. Малые размеры страниц полезны, когда рабочая нагрузка предполагает преимущественно произвольный доступ. Например, произвольный доступ используется в средах OLTP.

Дополнительную информацию о реорганизации таблиц смотрите в разделе “Реорганизация каталогов и пользовательских таблиц” на стр. 281.

Управление пространством

Оператор SQL INSERT служит для записи в таблицу новой информации. В процессе вставки для завершения работы используется поисковый алгоритм INSERT. Вначале при помощи FSCR (Free Space Control Records - записей управления свободным пространством) отыскивается страница с достаточным пространством. Но даже когда FSCR указывает страницу с достаточным свободным пространством, оно может оказаться недоступным, если оно “защищено” непринятой операцией DELETE из другой транзакции. Следовательно, необходимо, чтобы все транзакции часто выполняли принятие (COMMIT), в противном случае непринятое свободное пространство будет оставаться недоступным.

Не все FSCR в таблице просматриваются при поиске. Переменная реестра DB2MAXFSCRSEARCH ограничивает число записей FSCR, учитываемых при попытке вставки. По умолчанию значение этой переменной реестра равно пяти. Если в пределах пяти FSCR пространство не найдено, вставляемая запись добавляется в конец таблицы. Чтобы повысить скорость INSERT, последующие записи также добавляются в конец таблицы, пока не заполнятся два экстента. После заполнения двух экстентов следующая операция INSERT возобновляет поиск, начиная с той FSCR, где закончился предыдущий поиск.

Примечание: Значение DB2MAXFSCRSEARCH играет важную роль. Чтобы повысить скорость INSERT (ценой, возможно, более быстрого роста таблицы), задайте этой переменной реестра меньшее значение. Чтобы усилить повторное использование пространства (ценой, возможно, замедления операция INSERT), задайте этой переменной реестра большее значение.

После того, как при поиске будет просмотрена вся таблица, вставка записи будет производиться добавлением в конец таблицы без дополнительного поиска.

Поиск при помощи FSCR не возобновляется, пока где-либо в таблице не появится пространство (например, вследствие операции DELETE).

Есть еще две опции алгоритма поиска. Первая - это APPEND MODE. В этом режиме новые строки всегда добавляются в конец таблицы. Поиск среди записей FSCR не проводится и сами записи не поддерживаются. Эта опция включается при помощи оператора ALTER TABLE APPEND ON и может повысить производительность для таблиц, которые только растут, например, журналов. Вторая опция позволяет определить на таблице индекс кластеризации. В этом случае менеджер баз данных пытается вставлять записи на той же странице, что и остальные записи с близкими значениями ключа индекса. Если на данной странице нет пространства, делается попытка поместить запись на соседние страницы. Если и это не удается, используется описанный выше алгоритм просмотра записей FSCR – с одним небольшим отличием: вместо принципа принятия первого подходящего варианта используется принцип принятия худшего подходящего варианта. Этот принцип требует выбирать страницы с большим свободным пространством. Это делается для того, чтобы основать новую область кластеризации для строк с данным значением ключа.

Определив на таблице индекс кластеризации, выполните оператор ALTER TABLE... PCTFREE, прежде чем загружать или реорганизовывать таблицу. Условие PCTFREE оставляет заданный процент свободного пространства на странице данных таблицы после загрузки и реорганизации. Это повышает вероятность того, что при операции кластерного индекса свободное пространство на нужной странице будет найдено.

Страницы данных и записи переполнения

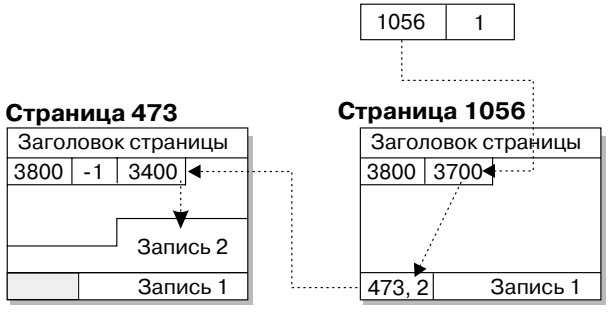


Рисунок 6. Страница данных и записи переполнения

Записи переполнения могут выполняться, когда затребованное изменение увеличивает существующую запись настолько, что она не умещается на текущей странице. Увеличенная запись вставляется как запись переполнения на другой странице, где достаточно места. Первоначальный RID преобразуется в запись указателя, содержащую новый RID записи переполнения. В индексе таблицы

сохраняется первоначальный RID, так что при требовании получить данные этой записи потребуется дополнительное чтение страниц. Большое число записей переполнения ведет к многочисленным дополнительным чтениям страниц и замедляет доступ к таблице. Реорганизация таблицы устраняет записи переполнения. Однако по мере возможности следует избегать изменений, увеличивающих записи, чтобы не допускать записей переполнения.

Управление индексом

Индексы DB2 - это реализация оптимизированного B-дерева, основанная на эффективном и обеспечивающем высокую степень параллелизма методе управления индексом при помощи регистрации с предварительной записью.

В реализации оптимизированного B-дерева используются двусторонние указатели на конечных страницах, что позволяет по одному индексу проводить просмотр как в прямом, так и в обратном направлениях (но не в обоих одновременно). Обычно страницы индекса разбиваются точно пополам, кроме страниц верхних ключей, где используется разбиение 90/10. Это значит, что верхние десять процентов ключей индекса размещаются на новой странице. Этот тип разбиения страниц индекса полезен при рабочей нагрузке, где требования INSERT часто завершаются с новыми верхними ключами.

Страницы индекса освобождаются при удалении последнего ключа индекса на странице. Исключение из этого правила - создание индекса с условием MINPCTUSED. Использование этого условия разрешает фоновую реорганизацию индекса; заданное в этом условии значение - это порог минимального процента занятого пространства на конечных страницах индекса. Если после удаления ключа со страницы индекса процент пространства, использованного на странице, меньше или равен заданному значению, делается попытка объединить оставшиеся ключи с ключами на соседней странице. Если места достаточно, слияние производится и конечная страница индекса удаляется. Использование этого условия может улучшить повторное использование пространства; с другой стороны, если задать слишком большое значение, время, потраченное на попытки слияния, вырастет, а вероятность успешного слияния упадет. Рекомендуется, чтобы значение в этом условии было ниже пятидесяти процентов.

Условие INCLUDE оператора CREATE INDEX позволяет в дополнение к столбцам ключей включать в конечные страницы индекса значения одного или нескольких заданных столбцов. Это может повысить долю запросов, для которых потребуется обращаться только к индексу. Однако это может также увеличить требования к пространству индексов и, возможно, стоимость поддержания индекса, если заданные столбцы часто изменяются. Упорядочение B-дерева индекса производится только по столбцам ключей, дополнительные столбцы не используются.

Блокировка

Менеджер баз данных обеспечивает управление одновременным выполнением, предотвращая при помощи блокировок неконтролируемый доступ к ресурсам и данным. **Блокировка** связывает прикладную программу с ресурсом менеджера баз данных или с записью данных. Блокировка управляет возможным доступом других прикладных программ к тому же ресурсу или той же записи данных.

Менеджер баз данных использует блокировку уровня записи или же уровня таблицы с учетом следующих факторов:

- Уровень изоляции, заданный во время прекомпиляции или при связывании прикладной программы с базой данных. Возможные уровни изоляции: Чтение непринятого (UR), Стабильность на уровне указателя (CS), Стабильность чтения (RS) и Многократное чтение (RR). Разные уровни изоляции используются для управления доступом к непринятым данным, предотвращения потерь изменений, допущения чтения данных без повторений и предотвращения чтения фантомных строк. Используйте минимальный уровень изоляции, который удовлетворяет потребностям вашей прикладной программы.
- План доступа, выбираемый программой оптимизации. Просмотр таблиц, индексов и другие методы доступа к данным требуют различных типов доступа к данным.
- Атрибут таблицы LOCKSIZE. Условие LOCKSIZE в операторе ALTER TABLE задает минимальный размер блокировки при обращении к этой таблице. Возможные значения - ROW (блокировка строк) и TABLE (блокировка таблицы). Используйте ALTER TABLE... LOCKSIZE TABLE для таблиц только для чтения. Это уменьшит число блокировок, требуемых при работе базы данных.
- Объем памяти, отводимой для блокировок. Объем памяти, отводимой для блокировок, управляется параметром *locklist* конфигурации базы данных. При заполнении списка блокировок возможно снижение производительности из-за расширения блокировок и снижения параллелизма в отношении объектов совместного доступа в базе данных. Если вы сталкиваетесь с частыми расширениями блокировок, увеличьте значение *locklist* и/или *maxlocks*.

Убедитесь, что все транзакции часто выполняют принятие, освобождая удерживаемые блокировки.

Блокировка, вообще говоря, производится на уровне записей, кроме следующих ситуаций:

- Выбран уровень изоляции Чтение непринятого (UR).
- Выбран уровень изоляции Многократное чтение (RR), а план доступа требует просмотра без предикатов.
- Атрибут таблицы LOCKSIZE имеет значение “TABLE”.
- Список блокировок заполнен и происходит расширение блокировки.

- Блокировка таблицы получена по явному требованию оператора LOCK TABLE. Оператор LOCK TABLE запрещает процессам конкурирующих программ изменять или использовать таблицу.

Расширение блокировки - это преобразование одной или нескольких блокировок записей в блокировку таблицы. Монопольное расширение блокировки - это расширение блокировки, при котором производится монопольная блокировка таблицы. Расширения блокировки снижают параллелизм, поэтому их следует избегать.

Продолжительность блокировки записи зависит от используемого уровня изоляции:

- Просмотр при уровне Чтение непринятого (UR): Блокировки удерживаются только для изменяемых записей.
- Просмотр при уровне Стабильность на уровне указателя (CS): Блокировки записей удерживаются, только пока указатель находится на записи.
- Просмотр при уровне Стабильность чтения (RS): На протяжении транзакции удерживаются только блокировки определяющих записей.
- Просмотр при уровне Многократное чтение (RR): Все блокировки записей удерживаются на протяжении транзакции. Если в вашей среде этот уровень изоляции не нужен или не желателен, используйте переменную реестра DB2_RR_TO_RS. Тем самым вы укажете менеджеру баз данных избегать лишних блокировок, требуемых для обеспечения семантики RR, и в результате повысится производительность.

Дополнительную информацию по этой теме смотрите в разделе “Блокировка” на стр. 52.

Ведение журнала

Доступны две стратегии ведения журнала:

- Циклическая запись - после заполнения файлов журнала первоначальные записи в первом файле журнала переписываются. Записи, стертые при переписывании, не допускают восстановления.
- Сохранение файлов журнала - при заполнении файла журнала записями файл архивируется. Для ведения журнала открывается доступ к новым файлам журнала. При сохранении файлов журнала возможно **восстановление с повтором транзакций**. Восстановление с повтором транзакций повторяет изменения базы данных с учетом завершенных единиц работы (транзакций), зарегистрированных в журнале. Можно задать восстановление с повтором всех транзакций вплоть до окончания файлов журнала или до заданного момента времени, предшествующего окончанию журнала.

Все изменения обычных страниц данных и страниц индекса, независимо от выбранной стратегии, записываются в буфер журнала. Данные в буфере журнала записываются на диск только:

- Перед записью на диск соответствующих страниц данных. Это называется “опережающей записью журнала”.
- При выполнении принятия или после достижения заданного числа группируемых принятий (параметр *mincommit* конфигурации базы данных).
- Когда буфер журнала почти заполнен. Процесс, записывающий журнал, сохраняет данные журнала на диск, чтобы избежать условия “переполнения буфера журнала”.

Примечание: В тот момент, когда транзакция завершается при помощи оператора COMMIT, все измененные страницы сохраняются на диск для обеспечения возможности восстановления.

Если транзакции коротки, операции ввода/вывода журнала могут стать “узким местом” из-за частой записи журнала при каждом принятии. В таких средах задание параметру конфигурации *mincommit* значения больше единицы может устранить “узкое место”. При значении больше единицы происходит принятия нескольких транзакций группируются вместе. Принятие первой транзакции откладывается до поступления еще (*mincommit* - 1) требований COMMIT от других транзакций; затем журнал записывается на диск, и все транзакции начинают отвечать своим программам. В результате вместо нескольких отдельных операций ввода/вывода журнала производится только одна.

Чтобы избежать чрезмерного ухудшения времени ответа, каждая транзакция не более одной секунды ожидает операторы COMMIT от (*mincommit* - 1) других транзакций. По истечении этой секунды транзакция сама инициирует запись журнала и получает возможность ответить своей прикладной программе. Благодаря этому можно задать *mincommit* без большого риска снижения производительности при обработке редких транзакций.

Изменения больших объектов и длинных переменных (LONG VARCHAR) отслеживаются посредством теневой подкачки страниц. Изменения столбцов больших объектов не регистрируются в журнале, кроме случаев, когда используются сохранение файлов журнала, и для столбца больших объектов в операторе CREATE TABLE не задано условие NOT LOGGED. Изменения на страницах размещения для типов данных LONG и больших объектов регистрируются в журнале как обычные страницы данных.

Что происходит при изменении данных

Что происходит с журналом и страницей данных, когда агент изменяет страницу? Описанный здесь протокол минимизирует операции ввода/вывода, требуемые для транзакции, одновременно обеспечивая возможность восстановления.

Во-первых, подлежащая изменению страница фиксируется и блокируется в монопольном режиме. В буфер журнала делается запись о том, как повторить и как отменить это изменение. Часть этого действия - получение

последовательного номера записи журнала (LSN) и помещение этого номера в заголовок подлежащей изменению страницы. После этого производится изменение страницы. Наконец, страница освобождается от блокировки и фиксации. Страница теперь считается “грязной”, поскольку в нее внесены изменения, еще не записанные на диск. Изменения были сделаны и в буфере журнала.

Как данные в буфере журнала, так и “грязная” страница данных нуждается в записи на диск. Из соображений производительности соответствующие операции ввода/вывода откладываются до удобного момента (например, во время перерыва в нагрузке системы) или до момента, когда это необходимо для обеспечения возможности восстановления или уменьшения времени восстановления. А именно, “грязная” страница записывается на диск:

- Когда другой агент выбирает ее для удаления из буфера.
- Когда чистильщик страниц выбирает эту страницу из-за того, что:
  - Другой агент выбрал ее для удаления из буфера.
  - Превышено процентное значение параметра *chngpgs_thresh* конфигурации базы данных. Превышение этого порога “активирует” чистильщики страниц, которые записывают измененные страницы на диск.
  - Превышено процентное значение параметра *softmax* конфигурации базы данных. Превышение этого порога “активирует” чистильщики страниц, которые записывают измененные страницы на диск.
- Когда страница изменена как часть таблицы, определенной с условием NOT LOGGED INITIALLY, и выполняется COMMIT. COMMIT записывает все измененные страницы на диск, чтобы обеспечить возможность восстановления.

Буфер журнала записывается на диск управляемой единицей ядра (EDU) программы журнала:

- Перед записью на диск соответствующих страниц данных. Это называется “опережающей записью журнала”.
- При выполнении принятия или после достижения заданного числа группируемых принятий (параметр *mincommit* конфигурации базы данных).
- Когда буфер журнала почти заполнен. Процесс, записывающий журнал, сохраняет данные журнала на диск, чтобы избежать условия “переполнения буфера журнала”.

Модель процесса

С одной и той же базой данных могут работать процессы локальных и удаленных прикладных программ. Удаленная прикладная программа - это программа, которая инициирует операции над базой данных с компьютера, удаленного по отношению к компьютеру базы данных. Локальные прикладные программы напрямую подсоединяются к базе данных на компьютере сервера.

На следующем рисунке каждый кружок представляет управляемую единицу ядра (EDU) - “процесс” на платформах UNIX и “поток” на платформах Windows NT и OS/2.

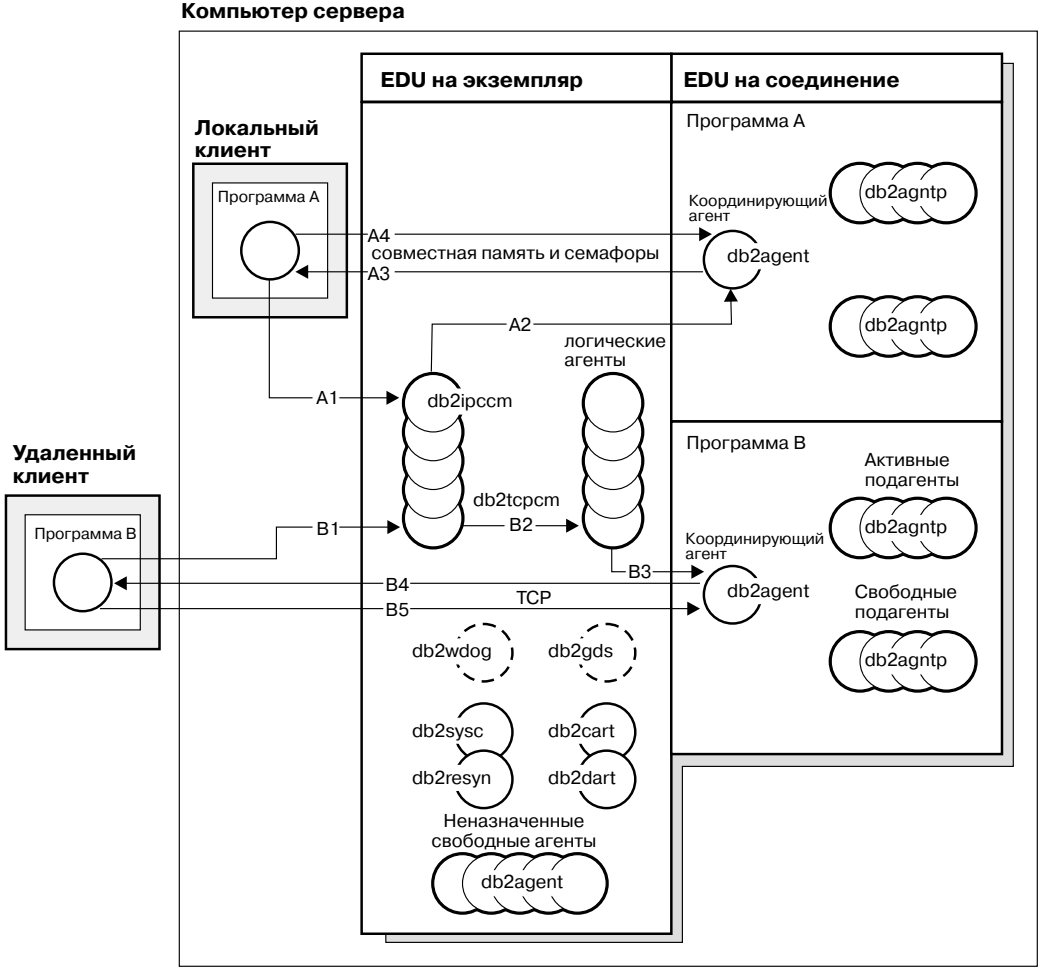


Рисунок 7. Общая схема моделей процесса

Прежде чем может быть выполнена работа на базе данных, нужная прикладной программе, должна быть установлена связь между прикладной программой и менеджером баз данных.

На приведенном выше рисунке стрелка A1 показывает первоначальное установление связи локального клиента с управляемой единицей ядра (EDU) db2ipccm. Эта EDU работает с EDU db2agent (стрелка A2), которая становится агентом-координатором для требований прикладной программы, поступающим от локального клиента. Агент-координатор связывается с клиентской

программой (стрелка A3) и устанавливает совместную память и семафоры для связи между клиентской программой и базой данных (стрелка A4). Прикладная программа на локальном клиенте соединяется с базой данных.

На том же рисунке удаленный клиент вначале устанавливает связь с EDU db2tcpcm (стрелка B1). Если был выбран другой протокол связи, будет использована соответствующая EDU. EDU db2tcpcm в B2 работает с логическим агентом (стрелка B2). Эта EDU работает с EDU db2agent (стрелка B3), которая становится агентом-координатором для требований прикладной программы, поступающим от удаленного клиента. Агент-координатор контактирует с клиентской программой (стрелка B4) и устанавливает связь TCP/IP между клиентской программой и базой данных (стрелка B5). Прикладная программа на удаленном клиенте соединяется с базой данных.

Обратите также внимание на то, что на этом рисунке:

- Есть два класса агентов: логические агенты и рабочие агенты. Логический агент представляет подключенную программу в менеджере баз данных. Рабочий агент выполняет запросы прикладной программы, но не приписан постоянно к конкретной программе.
- Существует четыре типа рабочих агентов: агенты активного координатора, подагенты, неактивные агенты и свободные агенты.
- Каждый процесс или поток клиентской программы, представленный логическим агентом, будет связан с агентом активного координатора.
- В среде многораздельных баз данных и в средах разрешенного внутрираздельного параллелизма агенты-координаторы распределяют требования базы данных по подагентам (db2agntp). Подагенты выполняют требования для прикладной программы.
- Существует пул агентов (db2agent), в котором свободные агенты ожидают новой работы.
- Есть и другие EDU, управляющие подключением клиентов, журналами, двухфазными принятиями, задачами резервного копирования и восстановления и другими задачами.

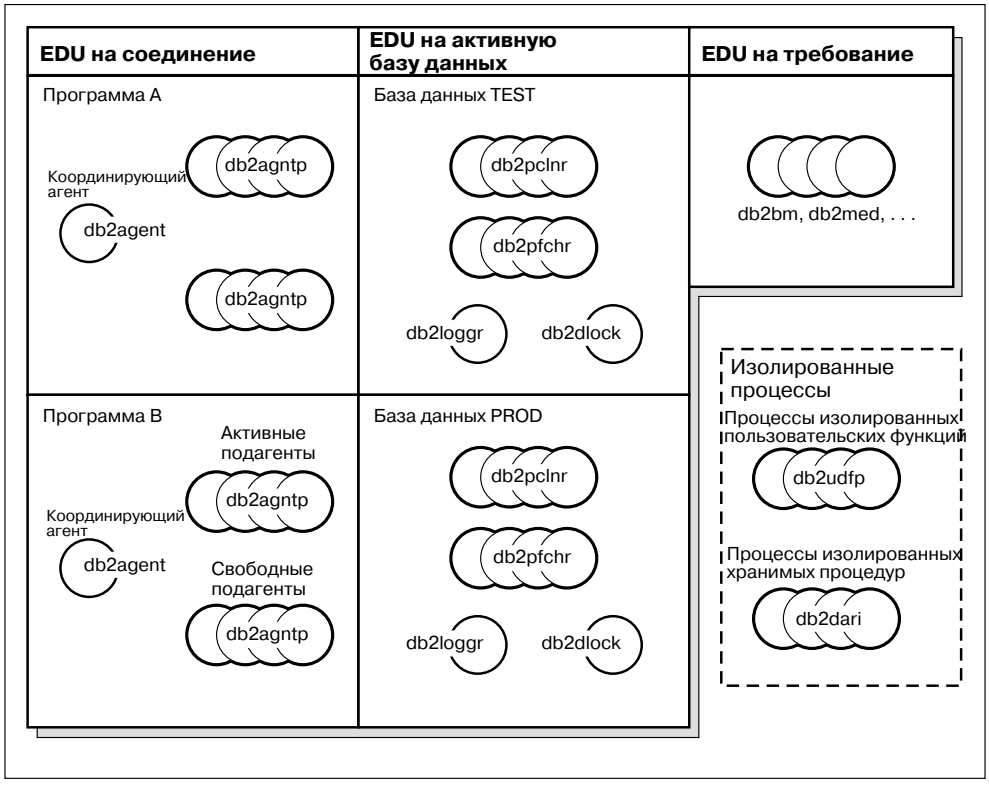


Рисунок 8. Модели процессов, часть 2

На этом рисунке показаны дополнительные управляемые единицы ядра (EDU), являющиеся частью среды компьютера сервера. Каждая активная база данных имеет собственный совместный пул программ предварительной выборки (db2pfchr), чистильщиков страниц (db2pclnr), а также собственную программу ведения журнала (db2loggr) и детектор тупиковых ситуаций (db2dlock).

Кружки с метками “db2udfp” и “db2dari” в нижней правой части рисунка изображают процессы, выполняемые в DB2 Universal Database как изолированные пользовательские функции и хранимые процедуры соответственно. Эти процессы управляются так, чтобы минимизировать стоимость их создания и уничтожения. По умолчанию параметр *keepdari* конфигурации менеджера баз данных имеет значение “YES”, что делает процесс хранимой процедуры доступным для повторного использования при следующем вызове хранимой процедуры.

Примечание: Бывают также неизолированные пользовательские функции хранимые процедуры, которые выполняются прямо в адресном пространстве агента. Этот способ работы повышает

производительность. С другой стороны, их неограниченный доступ к адресному пространству агента требует тщательной их проверки перед использованием.

Дополнительную информацию смотрите в главе о хранимых процедурах руководства *Application Development Guide*.

Модель обработки нескольких разделов - это логическое расширение модели обработки одного раздела. Собственно, оба режима работы поддерживаются одной общей базовой программой. На следующем рисунке показаны сходства и различия между одноразделной моделью обработки, которую можно видеть на двух предыдущих рисунках, и многоразделной моделью обработки.

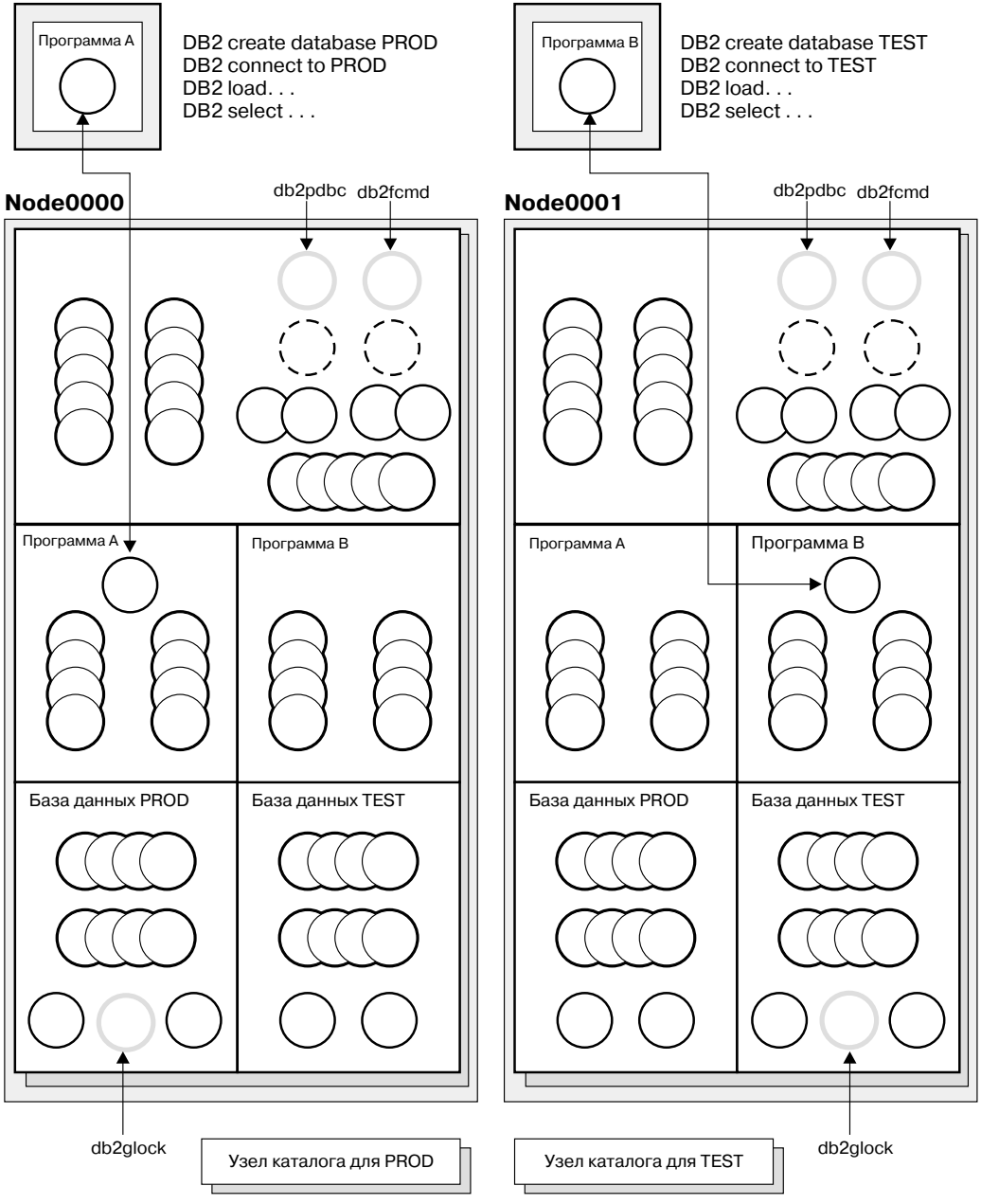


Рисунок 9. Модель обработки и многораздельность

Большинство управляемых единиц ядра (EDU) одинаковы для одноразделной модели обработки и многораздельной.

В среде с несколькими разделами (или узлами) один из разделов является узлом каталога. В каталоге хранится информация обо всех объектах в базе данных.

Как показано на приведенном выше рисунке, поскольку Программа A создает базу данных PROD на узле Node0000, на этом узле создается каталог для базы данных PROD. Аналогичным образом, поскольку Программа B создает базу данных TEST на узле Node0001, на этом узле создается каталог для базы данных TEST. Может оказаться желательным создавать базы данных на разных узлах, чтобы равномерно распределить по узлам среды системы дополнительную нагрузку, связанную с каталогами для каждой базы данных.

Есть дополнительные EDU (db2pdbc и db2fcmd), связанные с экземпляром, которые находятся на каждом узле в среде многораздельных баз данных. Эти EDU нужны для координации требований в разделах баз данных и обеспечения работы Менеджера быстрой связи (FCM).

Кроме того, еще одна EDU (db2glock) связана с узлом каталога базы данных. Эта EDU занимается урегулированием глобальных тупиковых ситуаций между узлами, на которых располагается активная база данных.

Каждое соединение от прикладной программы представлено логическим агентом на базе данных и ведет к созданию одного агента-координатора. Агент-координатор находится в разделе, с которым соединилась прикладная программа. Этот раздел становится “узлом-координатором” для этой прикладной программы. Узел-координатор может быть задан также командой *SET CLIENT CONNECT_NODE*. Часть запросов базы данных от прикладных программ посылаются узлом-координатором между подагентами в других разделах, и все результаты из других разделов объединяются на узле-координаторе перед возвращением прикладной программе.

Раздел базы данных, где подана команда CREATE DATABASE, называется “узлом каталога” базы данных. Именно в этом разделе базы данных хранятся таблицы каталога. В типичном случае все пользовательские таблицы распределяются по набору узлов.

Примечание: Любое число разделов может быть сконфигурировано для выполнения на одном компьютере. Такая конфигурация называется “логической многораздельностью” или “несколькими логическими узлами”. Это весьма полезно на больших компьютерах с симметрической многопроцессорной средой (SMP) и очень большой основной памятью. В этой среде связь между разделами может быть оптимизирована за счет использования совместной памяти и семафоров.

Модель памяти

Память играет важную роль, так как существенно влияет на выполнение работ в базе данных. Первая возможность повысить производительность базы данных связана с разделением доступной памяти между областями в пределах базы данных. Этим разделением памяти между разными кучами управляют параметры конфигурации. В этом разделе описываются ключевые параметры конфигурации и части памяти, которыми они управляют. Дополнительную информацию по этой теме смотрите в разделе “Как DB2 использует память” на стр. 253.

Все управляемые единицы ядра (EDU) в разделе базы данных подсоединены к совместной памяти экземпляра. Все EDU, работающие в базе данных, подсоединены к этой совместной памяти экземпляра базы данных. Все EDU, работающие для конкретной прикладной программы, подсоединены к региону совместной памяти прикладной программы, принадлежащему этой программе. Этот тип совместной памяти отводится только при разрешенном межраздельном и внутрираздельном параллелизме. Наконец, у каждой EDU есть своя собственная память.

Совместная память экземпляра (другое название - совместная память Менеджера баз данных) выделяется при запуске базы данных. Из этой совместной памяти экземпляра происходит подсоединение/выделение остальных видов памяти. Если используется Менеджер быстрой связи (FCM), из этой памяти берутся его буферы. FCM используется для внутренней связи - в первую очередь сообщений - между серверами и внутри серверов базы данных в конкретной среде баз данных. Когда к базе данных подключится или подсоединится первая прикладная программа, происходит выделение совместной памяти базы данных, совместной памяти прикладной программы и собственной памяти агента.

Совместная память базы данных (другое название - глобальная память базы данных) выделяется при запуске базы данных или первом подключении к ней. Эта память используется всеми прикладными программами, подключающимися к базе данных. Совместная память базы данных содержит много разных областей памяти, в том числе:

- Пулы буферов
- Список блокировок
- Куча базы данных – включающая буфер журнала и кэш каталога.
- Куча утилит
- Кэш пакета

Параметр *numdb* конфигурации менеджера баз данных задает число локальных баз данных, которые могут быть активны одновременно. В среде многораздельных баз данных этот параметр ограничивает число активных

разделов баз данных на сервере разделов базы данных. Значение параметра *numdb* может влиять на общий объем выделяемой памяти.

Совместная память прикладной программы (другое название - глобальная память прикладной программы) отводится при подключении программы к базе данных. Это выделение памяти производится только в среде многораздельных баз данных или при включенном параметре *intra_parallel* конфигурации менеджера баз данных. Эта память используется агентами прикладной программы для совместного доступа к данным и взаимной координации действий.

Параметр *maxappls* конфигурации базы данных задает верхний предел для числа программ, подключающихся к базе данных. Поскольку для каждой прикладной программы, подсоединяющейся к базе данных, надо выделить некоторую собственную память, много одновременно работающих программ могут потребовать большого объема памяти.

До некоторой степени максимальное число прикладных программ регулируется также параметром конфигурации менеджера баз данных *maxagents* (или *max_coordagents* для многораздельных сред). Параметр *maxagents* задает верхний предел общего числа агентов менеджера баз данных в разделе. В это число входят агенты активного координатора, подагенты, неактивные агенты и свободные агенты.

Собственная память агента выделяется агенту при назначении ему работы для конкретной прикладной программы. Собственная память агента отводится для этого агента и содержит области памяти, которые будут использоваться только этим конкретным агентом, например, куча сортировки и куча программы.

Есть несколько особых типов совместной памяти:

- Совместная память агента/локальной прикладной программы. Эта память используется для передачи запросов SQL и ответов между агентом и его клиентской программой.
- Совместная память UDF/агента. Эта память подсоединяется агентами, выполняющими изолированную пользовательскую функцию или хранимую процедуру. Она используется как область связи.
- Расширенная память. Как правило, очень большой (больше 4 Гбайт) регион совместной памяти, используемой как расширенный пул буферов. Агенты/программы предварительной выборки/чистильщики страниц не подсоединены к нему постоянно, а подсоединяются к отдельным его сегментам по мере надобности.

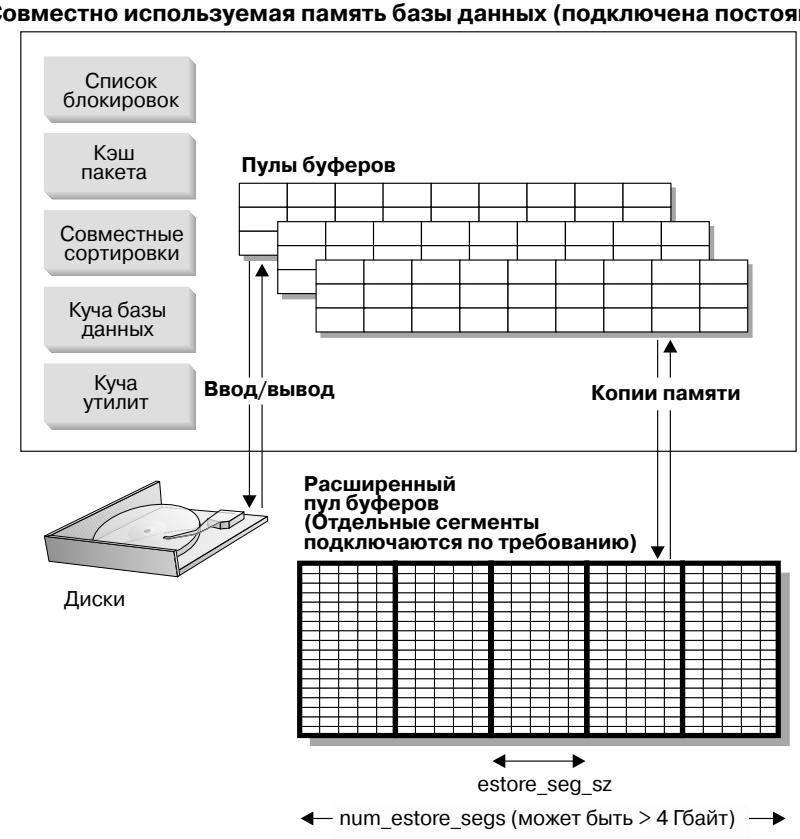


Рисунок 10. Пулы буферов и расширенная память

Расширенная память служит дополнительным буфером для одного или нескольких основных пулов буферов. Дополнительную информацию по этой теме смотрите в разделе “Расширение памяти” на стр. 296. Объем этой памяти может быть намного больше 4 Гбайт, что позволяет с успехом использовать компьютеры с очень большими объемами основной памяти. Кэш расширенной памяти определяется в терминах сегментов памяти.

Следует знать, что решение использовать некоторую часть реальной адресуемой памяти как кэш расширенной памяти исключает ее использование на компьютере для других целей, например, в качестве кэша журнальной файловой системы или собственного адресного пространства процесса. Передача дополнительной реальной адресуемой памяти под кэш расширенной памяти может увеличить системную подкачку страниц.

Следующие параметры конфигурации базы данных влияют на объем и размер памяти, доступной для расширения памяти:

- *num_estore_segs* задает число сегментов расширения памяти.
- *estore_seg_sz* задает размер каждого сегмента расширения памяти.

Каждому табличному пространству присваивается пул буферов. Кэш расширенной памяти всегда должен быть связан с одним или несколькими конкретными пулами буферов. Размер страницы кэша расширенной памяти должен совпадать с размером страницы связанного с ним пула буферов.

Дополнительную информацию о кэше расширенной памяти смотрите в разделе “Расширение памяти” на стр. 296.

Часть 2. Настройка производительности прикладных программ

Глава 3. Особенности прикладного программирования

Ряд факторов может повлиять на производительность выполнения вашей прикладной программы. В этой главе описаны следующие темы, которые надо учитывать при разработке и написании прикладной программы:

- Одновременность
- Блокировка
- Настройка класса оптимизации
- Ограничение наборов результатов для повышения производительности
- Блокирование строк
- Настройка запросов
- Составные операторы SQL
- Вопросы производительности и преобразование символов
- Хранимые процедуры
- Активация базы данных
- Параллельная обработка прикладных программ.

Кроме того, следует посмотреть руководства *Application Development Guide* и *CLI Guide and Reference*, в которых приводится дополнительная информация, связанная с производительностью прикладных программ:

- Написание программ, использующих встроенные статические SQL
- Написание программ, использующих встроенные динамические SQL
- Написание программ, использующих интерфейс уровня вызовов (CLI) DB2.

Одновременность

Целостность данных в реляционной базе данных должна поддерживаться при множестве обращений пользователей и изменений данных. *Одновременность* - это одновременное совместное использование ресурсов несколькими пользователями или прикладными программами. Менеджер баз данных управляет операциями доступа, чтобы предотвратить такие нежелательные эффекты, как:

- *Потеря изменений данных*. Две прикладные программы, A и B, могут прочитать из базы данных одну и ту же строку и на основе ее содержимого вычислить новые значения для одного из столбцов этой строки. Если программа A записывает в строку новое значение, а затем в эту же строку записывает новое значение программа B, выполненное процессом A изменение данных будет утеряно.

- *Обращение к непринятым данным*. Прикладная программа A может изменить значение в базе данных, а программа B может прочитать это значение до того, как оно будет принято. Если впоследствии для этого значения, записанного программой A, будет выполнено не принятие, а откат, вычисления процесса B будут основаны на непринятых (и, возможно, неверных) данных.
- *Получение измененных данных при повторном чтении.* В некоторых прикладных программах происходит следующая последовательность событий: Прикладная программа A читает из базы данных строку и затем выполняет обработку других требований SQL. Тем временем прикладная программа B изменяет или удаляет эту строку и выполняет принятие изменений. Если после этого прикладная программа A попытается вновь прочитать исходную строку, она получит измененную строку или обнаружит, что исходная строка была удалена.
- *Эффект чтения фантомных строк*. Эффект чтения фантомных строк возникает, когда:
  1. Прикладная программа выполняет запрос, читающий набор строк по заданному критерию поиска.
  2. Другая прикладная программа вставляет новые данные или изменяет существующие данные и эти новые данные удовлетворяют критерию поиска первой программы.
  3. Первая программа повторяет запрос из шага 1 (в той же единице работы).

  При повторении запроса (шаг 3) в наборе результатов возвращаются некоторые дополнительные строки (“фантомные”), которые не были возвращены при первом выполнении запроса (шаг 1).

Уровень изоляции определяет режим блокировки или изоляции данных от других обращающихся к ним процессов. Уровень изоляции действует на протяжении единицы работы. Для прикладных программ, использующих указатель, объявленный в операторе DECLARE CURSOR с условием WITH HOLD, выбранный уровень изоляции будет сохраняться в течение единицы работы, в которой был выполнен оператор OPEN CURSOR. (Дополнительную информацию смотрите в руководстве *SQL Reference*.) Информацию о задании уровня изоляции смотрите в разделе “Задание уровня изоляции” на стр. 50.

DB2 поддерживает следующие уровни изоляции:
- Многократное чтение
- Стабильность чтения
- Стабильность на уровне указателя
- Чтение непринятого.

(Обратите внимание на то, что некоторые серверы баз данных DRDA поддерживают уровень изоляции *без принятия*. В других базах данных этот

уровень изоляции совпадает с уровнем изоляции ″чтение непринятого″. Информацию об этом уровне изоляции смотрите в справочнике *SQL Reference*.)

Смотрите также:
- “Выбор уровня изоляции” на стр. 49
- “Задание уровня изоляции” на стр. 50.

Используемая *система баз данных объединения* может разрешать прикладным программам и пользователям выполнять операторы SQL, обращающиеся (в одном операторе) к нескольким СУБД или к нескольким базам данных. Система объединения DB2 обеспечивает *прозрачность положений* для объектов баз данных. Например, если данные в таблицах и производных таблицах перемещены, можно исправить ссылки (так называемые *псевдонимы*), не внося никаких изменений в сами программы, запрашивающие данные. Если прикладная программа использует псевдонимы для обращения к данным, для обеспечения требуемых уровней изоляции DB2 использует протоколы управления одновременностью менеджеров баз данных источников данных. (Под *источником данных* понимается СУБД и сами данные.) DB2 попытается подобрать для требуемого уровня изоляции его логический эквивалент на источнике данных, однако результат может зависеть от возможностей конкретного источника данных. Информацию о написании прикладных программ, использующих псевдонимы, смотрите в руководстве *Application Development Guide*.

Далее подробно описан каждый уровень изоляции; уровни изоляции перечислены в порядке уменьшения влияния на производительность (но увеличения требуемой степени внимания при обращении к данным и изменении данных).

Многократное чтение

При уровне изоляции *многократное чтение* (RR) блокируются все строки, к которым программа обращается в единице работы. При использовании многократного чтения если программа дважды выполняет оператор SELECT в той же единице работы, где был открыт указатель, результат оба раза будет одинаковым. Использование многократного чтения исключает потерю изменений, доступ к непринятым данным и фантомные строки.

Программа, использующая уровень изоляции многократное чтение, может до завершения единицы работы получать и обрабатывать строки необходимое число раз. Однако другие прикладные программы не могут до завершения этой единицы работы изменять, удалять или вставлять строки, которые могут повлиять на таблицу результатов. Программы, использующие уровень изоляции многократное чтение, не могут увидеть непринятые изменения, выполненные другими программами.

При использовании этого уровня изоляции блокируются все строки, на которые есть ссылки, а не только получаемые строки. Устанавливаются соответствующие блокировки, не позволяющие другим прикладным программам вставлять или изменять строки, которые могли бы быть добавлены в список строк, на которые ссылается запрос, при повторном выполнении запроса. Этим предотвращается возникновение фантомных строк. Это означает, что если просматриваются 10000 строк и к ним применяются предикаты, все эти 10000 строк блокируются, даже если предикатам удовлетворяют только 10 строк.

Примечание: Уровень изоляции многократное чтение обеспечивает неизменность всех возвращенных данных до их *использования* прикладной программой, даже если применяются временные таблицы или блокировка строк.

Поскольку при уровне изоляции многократное чтение может устанавливаться и удерживаться значительное число блокировок, их число может превысить число блокировок, разрешенных параметрами конфигурации *locklist* и *maxlocks*. (Смотрите разделы “Максимальный процент списка блокировок перед расширением (maxlocks)” на стр. 407 и “Максимальная память для списка блокировок (locklist)” на стр. 375.) Если оптимизатор предполагает, что с большой вероятностью возникнет расширение блокировок, он может выбрать использование для просмотра индекса одной блокировки уровня таблицы, чтобы избежать расширения блокировок. (Описание расширения блокировок смотрите в разделе “Расширение блокировок” на стр. 58.) Это выглядит так, как будто менеджер баз данных выполняет за вас оператор LOCK TABLE. Если вы не хотите, чтобы использовались блокировки уровня таблиц, надо разрешить для транзакции достаточно большое число блокировок или использовать уровень изоляции Стабильность чтения.

Стабильность чтения

При уровне изоляции *стабильность чтения* (RS) блокируются только строки, которые прикладная программа получает в единице работы. Таким образом, ни одна нужная строка, прочитанная в единице работы, не может быть изменена другими процессами программ до завершения этой единицы работы, и ни одна строка, измененная другой программой, не читается, пока изменение не будет принято этой программой. Таким образом **исключено** “получение измененных данных при повторном чтении”.

В отличие от уровня изоляции многократное чтение при уровне изоляции стабильность чтения при повторном выполнении одного запроса могут быть получены дополнительные *фантомные* строки (*явление чтения фантомных строк*). В описанном выше примере просмотра 10000 строк при уровне изоляции стабильность чтения будут блокироваться только строки, удовлетворяющие предикатам. Поэтому при этом уровне изоляции будут блокированы только получаемые строки (10 строк). В отличие от этого при уровне изоляции

многократное чтение будут блокированы все 10000 строк. Могут использоваться блокировки SHARE, NEXT SHARE, UPDATE или EXCLUSIVE. (Дополнительную информацию об атрибутах блокировок смотрите в разделе “Атрибуты блокировок” на стр. 53.)

Примечание: Уровень изоляции стабильность чтения обеспечивает неизменность всех возвращенных данных до их *использования* программой, даже если применяются временные таблицы или блокировка строк.

Одно из назначений уровня изоляции стабильность чтения - обеспечить высокую степень одновременности и стабильность получения данных. Поэтому оптимизатор не использует блокировки уровня таблиц, пока не возникнет расширение блокировок. (Дополнительную информацию о расширении блокировок смотрите в разделе “Расширение блокировок” на стр. 58.)

Уровень изоляции стабильность чтения лучше всего подходит для прикладных программ, которые:
- Работают в среде с одновременно работающими другими программами
- Требуют, чтобы выбранные строки оставались неизменными в течении единицы работы
- Не выполняет один запрос более одного раза в единице работы или не требует, чтобы при повторном выполнении запроса в той же единице работы был получен тот же результат.

Стабильность на уровне указателя

При уровне изоляции *стабильность на уровне указателя* (CS) блокируется строка, к которой обращается транзакция или прикладная программа, пока на этой строке находится указатель. Блокировка сохраняется до выбора следующей строки или до прекращения транзакции. Однако если данные в строке изменяются, блокировка должна сохраняться до принятия этого изменения.

Пока указатель с возможностью изменения находится на строке, полученной прикладной программой, использующей уровень изоляции стабильность на уровне указателя, другие прикладные программы не смогут изменить или удалить эту строку. Программы, использующие уровень изоляции стабильность на уровне указателя, не могут увидеть непринятые изменения, выполненные другими программами.

В описанном выше примере просмотра 10000 строк при уровне изоляции стабильность на уровне указателя будет блокироваться только строка, на которой в настоящее время находится указатель. Если эта строка не была изменена, ее блокировка освобождается после перемещения указателя с этой строки.

Уровень изоляции стабильность на уровне указателя допускает и разночтения, и явление чтения фантомных строк. Стабильность на уровне указателя - это уровень изоляции по умолчанию; его следует использовать, когда требуется максимальный параллелизм и доступ только к принятым строкам других программ.

Чтение непринятого

При уровне изоляции *чтение непринятого* (UR) прикладная программа может обращаться к непринятым данным других транзакций. Программа также не блокирует читаемую строку для других программ, кроме случаев, когда они пытаются удалить или изменить таблицу. Чтение непринятого работает по-разному для указателей только для чтения и изменяемых указателей.

Указатели только для чтения могут обращаться к большинству непринятых данных других транзакций. Однако таблицы, производные таблицы и индексы, создаваемые или отбрасываемые другими транзакциями, будут недоступны до завершения обработки этих транзакций. Все остальные изменения данных, выполненные другими транзакциями, могут быть прочитаны до того, как для них будет выполнено принятие или откат.

Примечание: Указатели с возможностью изменения, работающие с уровнем изоляции чтение непринятого, ведут себя так же, как с уровнем изоляции стабильность на уровне указателя.

При работе прикладной программы с уровнем изоляции UR может использоваться уровень изоляции CS. Это происходит, если в такой прикладной программе используются неоднозначные указатели. В зависимости от опции BLOCKING уровень изоляции для неоднозначных указателей может быть повышен до CS. По умолчанию опция BLOCKING имеет значение UNAMBIG. Это означает, что неоднозначные указатели считаются изменяемыми и происходит повышение уровня изоляции до CS. Есть два способа предотвратить повышение уровня изоляции. Во-первых, в прикладной программе можно сделать указатели однозначными, включив в оператор SELECT условие FOR READ ONLY. Во-вторых, можно оставить указатели в прикладной программе неоднозначными, но при прекомпиляции или связывании этой программы задать опцию BLOCKING ALL. Тогда во время выполнения программы все неоднозначные указатели будут считаться указателями только для чтения.

В описанном выше примере просмотра 10000 строк при уровне изоляции чтение непринятого блокировки вообще не будут использоваться.

Уровень изоляции чтение непринятого допускает и получение измененных данных при повторном чтении, и явление чтения фантомных строк.

Этот уровень изоляции чаще всего используется при запросах к таблицам только для чтения и при операциях выбора, когда обращение к непринятым данным других программ не влияет на результат.

Выбор уровня изоляции

В Табл. 1 показаны разные уровни изоляции и возможные нежелательные эффекты (описанные в руководстве *Application Development Guide*).

Таблица 1. Уровни изоляции

Уровень изоляции	**Обращение к непринятым данным**	**Получение измененных данных при повторном чтении**	**Явление чтения фантомных строк**
Многократное чтение (RR)	Невозможно	Невозможно	Невозможно
Стабильность чтения (RS)	Невозможно	Невозможно	Возможно
Стабильность на уровне указателя (CS)	Невозможно	Возможно	Возможно
Чтение непринятого (UR)	Возможно	Возможно	Возможно

В Табл. 2 представлены простые рекомендации, которые могут помочь при выборе исходного уровня изоляции для прикладных программ. Рассматривайте эту таблицу как исходную точку для рассмотрения и используйте представленное выше описание разных уровней изоляции, чтобы учесть факторы, которые могут повлиять на выбор подходящего уровня изоляции для конкретных требований.

Таблица 2. Рекомендации по выбору уровня изоляции

Тип прикладной программы	**Требуется высокая стабильность данных**	**Высокая стабильность данных не требуется**
Транзакции с чтением и записью	RS	CS
Транзакции только с чтением	RR или RS	UR

Очень важно выбрать для прикладной программы правильный уровень изоляции, чтобы избежать явлений, которые могут повлиять на работу этой прикладной программы. Уровень изоляции влияет не только на степень изоляции между прикладными программами, но также на характеристики производительности отдельных прикладных программ, поскольку требуемые для установления и освобождения блокировок ресурсы процессора и памяти зависят от уровня изоляции. Возможность тупиковых ситуаций также зависит от уровня изоляции.

Задание уровня изоляции

Уровень изоляции задается во время прекомпиляции или при связывании прикладной программы с базой данных. Для прикладной программы, написанной на поддерживаемом компилируемом языке, используйте опцию ISOLATION команд процессора командной строки PREP или BIND. Уровень изоляции можно также задать с помощью функций API PREP или BIND.

Если во время прекомпиляции создается файл связывания, информация об уровне изоляции сохраняется в файле связывания. Если во время связывания не задан уровень изоляции, по умолчанию используется уровень изоляции, использованный во время прекомпиляции.

Если уровень изоляции не задан, по умолчанию используется уровень изоляции стабильность на уровне указателя.

Уровень изоляции пакета можно узнать с помощью такого запроса:

```
SELECT ISOLATION FROM SYSCAT.PACKAGES
  WHERE PKGNAME = 'XXXXXXXX'
  AND PKGSCHEMA = 'YYYYYYYY'
```

где *XXXXXXXX* - имя пакета, а *YYYYYYYY* - имя схемы этого пакета. Оба эти имени должны задаваться в верхнем регистре.

При создании базы данных с ней связываются несколько файлов связывания, используемых для поддержки разных уровней изоляции для SQL в REXX (на серверах, поддерживающих REXX). При создании базы данных с ней также связываются другие пакеты процессора командной строки. Дополнительную информацию о файлах связывания смотрите в руководстве *Application Development Guide*.

Для соединений REXX и процессора командной строки с базой данных используется уровень изоляции по умолчанию - стабильность на уровне указателя. При изменения уровня изоляции состояние соединения не изменяется. Оно должно выполняться в состоянии CONNECTABLE AND UNCONNECTED или в состоянии IMPLICITLY CONNECTABLE. (Подробную информацию о состояниях соединения смотрите в описании оператора CONNECT TO в руководстве *SQL Reference*.)

Прикладная программа REXX может узнать используемый уровень изоляции, проверив значение переменной REXX SQLISL. Это значение изменяется при каждом выполнении команды CHANGE SQLISL.

Можно использовать переменную реестра профиля DB2_RR_TO_RS для минимизации блокировок следующего ключа с целью повышения одновременности и производительности. Однако это исключает применение уровня изоляции многократное чтение (RR). Рекомендуется задавать для этой

переменной значение ″Yes″, если вы не хотите применять уровень изоляции RR ни в какой из программ, работающих с базой данных. Чтобы это значение вступило в силу, необходимо остановить и запустить менеджер баз данных. После выполнения команды `db2start` это значение будет влиять на весь экземпляр. Если после того, как задано это значение, получен запрос на обращение к пользовательской таблице с использованием уровня изоляции RR, этот запрос будет изменен и для него будет использоваться уровень изоляции стабильность чтения (RS). Предупреждение при этом не выдается.

Помимо задания уровня изоляции на уровне пакетов при подготовке или связывании программы вы можете задать изоляцию на уровне операторов. Уровень изоляции для оператора задается при помощи условия WITH.

Изоляцию на уровне операторов поддерживают следующие операторы SQL:

- Оператор SELECT
- SELECT INTO
- DELETE с поиском
- INSERT
- UPDATE с поиском
- DECLARE CURSOR

При использовании изоляции уровня операторов существуют некоторые ограничения:

- Условие WITH нельзя использовать в подзапросах
- Опция WITH UR применяется только к операциям только для чтения. Если она используется в других ситуациях, уровень изоляции автоматически изменяется с “UR” на “CS”.
- Уровнем изоляции по умолчанию для оператора является уровень пакета, в котором этот оператор связан.
- Уровень изоляции оператора заменяет уровень изоляции для пакета, в котором появляется данный оператор.

Используя процессор командной строки, можно изменить уровень изоляции с помощью команды CHANGE ISOLATION LEVEL. Дополнительную информацию смотрите в руководстве *Command Reference*.

Для интерфейса уровня вызовов DB2 (DB2 CLI) можно изменить уровень изоляции, задав его в конфигурации DB2 CLI. Во время выполнения в CLI используется функция SQLSetConnectAttr с атрибутом SQL_ATTR_TXN_ISOLATION. Она задает уровень изоляции транзакции для текущего соединения, указываемого аргументом *ConnectionHandle*. В файле db2cli.ini можно также использовать ключевое слово TXNISOLATION.

Примечание: JDBC и SQLJ используют CLI DB2, поэтому заданные в файле db2cli.ini значения могут влиять на работу программ, использующих JDBC и SQLJ. Дополнительную информацию смотрите в руководстве *CLI Guide and Reference*.

Для задания уровня изоляции при работе с JDBC или SQLJ во время выполнения можно использовать метод setTransactionIsolation из java.sql. Дополнительную информацию смотрите в главе "Programming in Java" руководства *Application Development Guide*.

При работе с SQLJ при выполнении оптимизатора SQLJ db2profc создается пакет. Можно задать итоговые опции для этого пакета, указав требуемый уровень изоляции. Дополнительную информацию смотрите в главе "Programming in Java" руководства *Application Development Guide*.

Кроме этого, многие коммерческие прикладные программы позволяют пользователю выбирать уровень изоляции. Дополнительную информацию смотрите в руководстве *CLI Guide and Reference*.

Объявленные временные таблицы и одновременность

Для объявленных временных таблиц нет проблем одновременности, поскольку они доступны только для прикладной программы, в которой они объявлены. Этот тип таблицы существует только с момента объявления ее прикладной программой и до момента завершения или отключения программы.

Блокировка

Менеджер баз данных обеспечивает управление одновременным выполнением, предотвращая неконтролируемый доступ при помощи блокировок. *Блокировка* означает установление связи ресурса менеджера баз данных с некоторой прикладной программой, позволяющее управлять доступом других программ к этому же ресурсу. О программе, с которой связан ресурс, говорят, что она удерживает блокировку или владеет блокировкой.

Менеджер баз данных задает блокировки, чтобы запретить программам доступ к непринятым данным, записанным другими программами (если только используемый уровень изоляции не разрешает чтение непринятого). Этот принцип защищает целостность данных (то есть согласованность и безопасность данных). Блокировки также могут запрещать изменение строк (например, для программ с многократным чтением).

Для обеспечения целостности данных менеджер баз данных задает блокировки неявно, по собственной инициативе. Если только используемый уровень изоляции не разрешает чтение непринятого, прикладной программе нет необходимости явно требовать блокировку, чтобы скрыть непринятые данные от других процессов.

Основной принцип блокировки построен так, что в большинстве случаев вам не приходится предпринимать никаких действий для управления блокировками. Тем не менее есть несколько общих параметров, на основе которых программы получают блокировки. Зная свою локальную ситуацию, вы можете изменить эти параметры и улучшить использование системных ресурсов. Вам помогут следующие темы, в которых рассказывается о блокировке:

- Атрибуты блокировок
- Блокировки и производительность прикладных программ
- Факторы, влияющие на блокировку
- Оператор LOCK TABLE
- CLOSE CURSOR WITH RELEASE
- Сводка особенностей блокировки.

Атрибуты блокировок

У блокировок менеджера баз данных есть следующие основные атрибуты:

Режим Тип доступа, разрешаемого владельцу блокировки, а также тип доступа к заблокированному объекту, разрешаемый другим пользователям. Иногда этот атрибут называют *состоянием* блокировки.

Объект

Блокируемый ресурс. Единственный тип явно блокируемого объекта - это таблица. Менеджер баз данных устанавливает блокировки и на другие типы ресурсов, такие как строки, таблицы и табличные пространства. Блокируемый объект представляет собой *минимальный размер* блокировки.

Длительность

Длительность интервала времени, в течении которого удерживается блокировка. На длительность блокировки влияет уровень изоляции, как описано в разделе “Одновременность” на стр. 43.

В следующей таблице режимы и их влияния перечислены в порядке возрастания контроля над ресурсами.

Таблица 3. Сводка режимов блокировки

Режим блокировки	Допустимый тип объекта	Описание
IN (intent none - без предназначения)	Табличные пространства, таблицы	Владелец блокировки может читать любые данные в таблице, включая непринятые данные, но никакие из них не может изменять. Блокировок строк владелец блокировки не получает. Таблицу могут читать и изменять другие одновременно работающие программы.

Таблица 3. Сводка режимов блокировки (продолжение)

Режим блокировки	Допустимый тип объекта	Описание
IS (intent share - намерение совместного использования)	Табличные пространства, таблицы	Владелец блокировки может читать данные в заблокированной таблице, но не может их изменять. Если программа удерживает табличную блокировку типа IS, она получает блокировку типа S или NS на каждую прочитанную строку. В обоих случаях таблицу могут читать и изменять другие программы.
NS (next key share - следующий ключ совместный)	Строки	Владелец блокировки и все одновременно работающие программы могут читать, но не изменять заблокированную строку. Эту блокировку получают строки таблицы вместо блокировки S, когда для прикладной программы используется уровень изоляции RS или CS.
S (share - совместно используемый)	Строки, таблицы	Владелец блокировки и все одновременно работающие программы могут читать, но не изменять заблокированные данные. Возможна блокировка типа S отдельных строк таблицы. Если блокировкой типа S заблокирована таблица, блокировок строк не требуется.
IX (intent exclusive - намерение монопольного)	Табличные пространства, таблицы	Владелец блокировки и одновременно работающие программы могут читать и изменять данные в таблице. Когда данные читает владелец блокировки, каждая читаемая строка получает блокировку S, NS, X или U. Блокировку X получает также каждая строка, которую владелец блокировки изменяет. Таблицу могут читать и изменять другие одновременно работающие программы.
SIX (share with intent exclusive - совместно используемый с намерением монопольного)	Таблицы	Владелец блокировки может читать и изменять данные в таблице. Владелец блокировки получает блокировки типа X на каждую изменяемую строку, но не получает блокировок на читаемые строки. Таблицу могут читать другие одновременно работающие программы.
U (update - обновление)	Строки, таблицы	Владелец блокировки может изменять данные в заблокированной строке или таблице. Владелец блокировки получает блокировки типа X на строки до их обновления. Другие единицы работы могут читать данные в заблокированной строке или таблице, но не могут пытаться изменить их.
NX (next key exclusive - следующий ключ монопольный)	Строки	Владелец блокировки может читать, но не изменять блокированную строку. Этот режим подобен блокировке X, но совместим с блокировкой NS.

Таблица 3. Сводка режимов блокировки (продолжение)

Режим блокировки	Допустимый тип объекта	Описание
NW (next key weak exclusive - следующий ключ слабый монопольный)	Строки	Эту блокировку получает следующая строка при вставке некоторой строки в индекс таблицы, не являющейся таблицей каталога. Владелец блокировки может читать, но не изменять блокированную строку. Этот режим подобен блокировкам X и NX, но совместим с блокировками W и NS.
X (exclusive - монопольный)	Строки, таблицы	Владелец блокировки может читать и изменять данные в заблокированной строке или таблице. Возможна монопольная блокировка таблиц, и в таких таблицах блокировка строк не производится. Только прикладные программы с чтением непринятого могут обращаться к заблокированной таблице.
W (weak exclusive - слабый монопольный)	Строки	Эту блокировку получает строка при вставке некоторой строки в таблицу, не являющуюся таблицей каталога. Владелец блокировки может изменять блокированную строку. Эта блокировка подобна блокировке X, но совместима с блокировкой NW. Только прикладные программы с чтением непринятого могут обращаться к заблокированной строке.
Z (super exclusive - сверхмонопольный)	Табличные пространства, таблицы	Эту блокировку таблица получает при определенных условиях, например, когда таблица изменяется или отбрасывается, когда создается или отбрасывается индекс для таблицы или когда таблица реорганизуется. Таблицу не могут читать и изменять никакие другие одновременно работающие программы.

Примечание: Режимы блокировки с “намерением” используются только в отношении таблиц и табличных пространств. Для строк эти режимы не применяются.

Блокировки и производительность прикладных программ

Программистам следует учитывать несколько взаимосвязанных факторов, касающихся использования блокировок и их влияния на производительность программ. В число этих факторов входят:

- Степень параллельности и размер блокировки
- Совместимость блокировки
- Преобразование блокировки
- Расширение блокировок
- Ожидание блокировок и истечение сроков ожидания
- Тупиковые ситуации.

Степень параллельности и размер блокировки

Блокировка, удерживаемая прикладной программой, может лишить доступа другую прикладную программу. Поэтому для максимальной степени параллелизма блокировки на уровне строки предпочтительнее блокировок таблиц. С другой стороны, управление блокировками требует памяти и времени

процессора. Поэтому для уменьшения памяти и времени обработки единая блокировка таблицы предпочтительнее большого числа блокировок строк.

Можно определить минимальный размер блокировок на уровне строки или таблицы при помощи условия LOCKSIZE оператора ALTER TABLE. По умолчанию используются блокировки строк. Для постоянных блокировок таблиц, определенных оператором ALTER TABLE, используются только блокировки типов S и X. Производительность повышается, поскольку прикладной программе не требуется получать и освобождать весьма большое число блокировок строк. Получение постоянной блокировки таблицы при помощи оператора ALTER TABLE может оказаться предпочтительнее блокировки таблицы на одну транзакцию при помощи оператора LOCK TABLE в следующих случаях:

- Таблица - только для чтения, и всегда нужны блокировки типа S. Блокировка уровня таблицы повысит производительность, позволяя в то же время другим получать для этой таблицы блокировки типа S.
- К таблице будет обращаться один-единственный пользователь, который ее поддерживает, и это лицо требует блокировки типа X на ограниченное время. Изменение уровня блокировки таблицы при помощи ALTER TABLE в отношении таблиц обеспечит этому лицу блокировку типа X на уровне таблицы. После того, как это лицо завершит работу, можно при помощи ALTER TABLE вернуть таблицу к блокировке уровня строк.

Использование оператора ALTER TABLE не предотвратит случаев обычного расширения блокировки.

Кроме того, заметьте, что использование ALTER TABLE для принудительной установки блокировок уровня таблицы - это глобальный подход, влияющий на все прикладные программы и всех пользователей, обращающихся к этой таблице. Альтернативой этому подходу является использование оператора LOCK TABLE для отдельных программ. Это позволит переходить к блокировкам таблиц на уровне прикладных программ, а не базы данных (как было описано во втором пункте).

Совместимость блокировки

Табл. 4 указывает, будет ли удовлетворяться требование блокировки, если блокировку того же ресурса в данном состоянии удерживает или запрашивает другой процесс. Значение **no** указывает, что реквестер должен ожидать, пока другие процессы не освободят все несовместимые блокировки. Заметьте, что при ожидании блокировки может истечь срок ожидания. Значение **да** указывает, что блокировка будет получена (если только этого ресурса больше никто не ожидает).

Таблица 4. Совместимость типов блокировки

	Состояние удерживаемого ресурса												
Запрашивается состояние	**отсут-ствует**	**IN**	**IS**	**NS**	**S**	**IX**	**SIX**	**U**	**NX**	**X**	**Z**	**NW**	**W**
отсутствует	да	да	да	да	да	да	да	да	да	да	да	да	да
IN	да	да	да	да	да	да	да	да	да	да	нет	да	да
IS	да	да	да	да	да	да	да	да	нет	нет	нет	нет	нет
NS	да	да	да	да	да	нет	нет	да	да	нет	нет	да	нет
S	да	да	да	да	да	нет	нет	да	нет	нет	нет	нет	нет
IX	да	да	да	нет	нет	да	нет	нет	нет	нет	нет	нет	нет
SIX	да	да	да	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет
U	да	да	да	да	да	нет	нет	нет	нет	нет	нет	нет	нет
NX	да	да	нет	да	нет	нет	нет	нет	нет	нет	нет	нет	нет
X	да	да	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет
Z	да	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет	нет
NW	да	да	нет	да	нет	нет	нет	нет	нет	нет	нет	нет	да
W	да	да	нет	нет	нет	нет	нет	нет	нет	нет	нет	да	нет

Примечание:

I Намерение
N Нет
NS Следующий ключ совместный
S Совместно используемый
NX Следующий ключ монопольный
X Монопольная
U Изменение
Z Сверхмонопольная
NW Следующий ключ слабый монопольный
W Слабая монопольная

Подробности этих типов блокировки рассмотрены в разделе “Атрибуты блокировок” на стр. 53.

Примечание:

- да - **предоставлять** затребованную блокировку немедленно
- нет - **ожидать**, пока не освободится удерживаемая блокировка или истечет срок ожидания

Предположим, прикладная программа A сохраняет блокировку для таблицы, к которой также хочет обратиться программа B. Менеджер баз данных затребует для прикладной программы B блокировку в некотором конкретном режиме. Если режим блокировки, удерживаемой A, разрешает блокировку, затребованную B, эти две блокировки (или два режима) называются совместимыми.

Если режим блокировки, затребованный для прикладной программы B, не совместим с блокировкой, удерживаемой программой A, программа B не может продолжать работу. Напротив, она должна ждать, пока программа A не освободит свою блокировку, и не будут освобождены *все* другие существующие несовместимые блокировки.

Преобразование блокировки

Преобразование блокировки производится, когда процесс обращается к объекту данных, в отношении которого уже удерживается блокировка, и режим доступа требует более сильной блокировки, нежели та, что уже удерживается. В отношении одного объекта данных процесс может удерживать только одну блокировку в одно время, хотя он может (косвенно, посредством запроса) много раз затребовать блокировку в отношении одного и того же объекта. Операция изменения режима уже удерживаемой блокировки называется *преобразованием*.

Причина преобразования для строк проста: например, преобразование производится, если требуется X, а удерживается S или U.

Есть разные типы блокировок для таблиц и для строк. Однако блокировки типов IX (intent exclusive - намерение монопольного) и S (Shared - совместный) - особые случаи с точки зрения преобразования блокировок. Ни S, ни IX не считаются более сильным типом, чем другой, так что если удерживается один из них, а затребован другой, производится преобразование в блокировку SIX (share with intent exclusive - совместно используемый с намерением монопольного). Все другие преобразования приводят к тому, что затребованный режим блокировки превращается в режим удерживаемой блокировки, если затребован менее сильный режим.

Запрос на изменение строки также может привести к двойному преобразованию. Допустим, что строка была прочитана посредством обращения к индексу и была заблокирована в режиме S. У таблицы, содержащей эту строку, будет покрывающая блокировка с намерением. Допустим, это блокировка типа IS, а не IX. Тогда, если строка затем будет изменена, блокировка таблицы будет преобразована в IX, а строки - в X.

Напомним, что прикладная программа обычно получает блокировки неявно, при выполнении запроса. Знание видов блокировки, получаемых при различных запросах и сочетаниях таблиц и индексов, помогает в разработке и настройке прикладных программ. Дополнительную информацию по этой теме смотрите в разделе “Факторы, влияющие на блокировку” на стр. 63.

Расширение блокировок

Расширение блокировок - это внутренний механизм для сокращения числа удерживаемых блокировок. Расширение состоит в переходе от многих блокировок строк (в одной таблице) к одной блокировке таблицы.

Расширение блокировки производится, когда число удерживаемых сейчас блокировок (любого типа) становится слишком большим.

Расширение блокировки может производится для заданного агента базы данных, если агент превысит отведенный ему размер списка блокировок (смотрите раздел “Максимальный процент списка блокировок перед расширением (maxlocks)” на стр. 407).

Такое расширение обрабатывается внутренним образом; единственным обнаружимым внешне результатом может оказаться сокращение одновременного доступа к одной или нескольким таблицам. Обычно в правильно сконфигурированной базе данных расширение блокировки происходит не часто.

Например, расширение блокировки может производиться, когда разработчик прикладной программы для большей производительности и параллельности использовал индекс для большой таблицы, но программа стала обращаться к большому проценту записей таблицы. В этом случае менеджер баз данных не был в состоянии спрогнозировать блокировку столь большой части таблицы, и блокировал отдельно каждую запись, вместо того чтобы просто заблокировать таблицу в режиме S или X. Для решения этой проблемы разработчик базы данных может посоветоваться с разработчиком прикладной программы и рекомендовать для этой транзакции оператор LOCK TABLE.

Иногда процесс, получающий требование расширения (внутренним образом), удерживает лишь небольшое число блокировок записей или даже не удерживает ни одной блокировки ни на одной таблице. Причина этого расширения в том, что один или несколько процессов могут удерживать большое число блокировок (хотя это число и не достигает числа блокировок в одном процессе, заданного значением параметра конфигурации базы данных), которое еще не вызывает требования расширения. Этот процесс может не потребовать новых блокировок или обращений к базе данных, пока ему не понадобится завершить транзакцию. В это время другой процесс может потребовать одну или несколько блокировок, что вызовет требование расширения.

Если расширение блокировок снижает параллелизм до неприемлемого уровня, можно сделать следующее:

- Проверьте, какую информацию о расширениях содержит *db2diag.log*. Информация записывается отдельно для каждой таблицы, захваченной расширением. Записывается следующая информация:
  - Число текущих блокировок.
  - Число блокировок, необходимое для выполнения расширения.
  - Информация об идентификаторе и имени таблицы для каждой таблицы, захваченной расширением.
  - Число удерживаемых сейчас нетабличных блокировок.

– Новая блокировка уровня таблицы, получаемая в рамках расширения. В типичном случае это будет блокировка в режиме “S” (share - совместно используемый) или “X” (exclusive - монопольный).
– Внутренний код возврата в результате получения новой блокировки уровня таблицы.

Может также быть записан текущий динамический оператор SQL. В таком случае записываемая информация будет содержать текущий оператор SQL, предшествовавший расширению блокировок таблицы, **если** параметр DIAGLEVEL конфигурации менеджера баз данных равен 4. Если при попытке расширения блокировки произойдет сбой, записываемая информация будет содержать таблицу, для которой расширение не удалось, и текущий оператор SQL (если он доступен и не был записан ранее), **если** DIAGLEVEL больше или равен 2.

При помощи этой информации вы сможете принять необходимые меры, опираясь на другие пункты, изложенные ниже.

Чтобы начать запись этого типа информации, нужно задать параметр DIAGLEVEL конфигурации менеджера баз данных равным 3 (значение по умолчанию) или 4.

- Увеличьте допустимое число блокировок, увеличив значение параметров *maxlocks* и/или *locklist* в файле конфигурации базы данных. (Смотрите разделы “Максимальный процент списка блокировок перед расширением (maxlocks)” на стр. 407 и “Максимальная память для списка блокировок (locklist)” на стр. 375.) Этот метод возможен, когда основную роль играет одновременный доступ к таблице со стороны других процессов. Однако стоимость получения блокировок на уровне записей может привести росту задержек для других процессов, что сведет на нет экономию за счет параллельного доступа к таблице. (При изменении этих параметров в многораздельной базе данных убедитесь, что параметры изменяются во всех разделах).
- Найдите и отрегулируйте один или несколько причастных процессов (не обязательно тех, которые вызывают расширение или откат), и явно выполните операторы LOCK TABLE.
- Измените степень изоляции. Помните, что это может привести к снижению параллелизма.
- Повысьте частоту принятий. Это обычно уменьшает число одновременно существующих блокировок. Дополнительную информацию об уровнях изоляции и параллелизме смотрите в разделе “Одновременность” на стр. 43.

Ожидание блокировок и истечение сроков ожидания

Без истечения сроков блокировок в ненормальной ситуации прикладной программе пришлось бы ждать освобождения блокировки до бесконечности. Это может случиться, например, когда транзакция ожидает блокировки,

удерживаемой программой другого пользователя, а тот ушел с рабочей станции, не завершив диалога, который разрешил бы его программе принять транзакцию и освободить блокировку. Ясно, что это снижает производительность программ. Предотвратить это можно с помощью параметра конфигурации *locktimeout*, задающего максимальное время ожидания для получения блокировки какой-либо программой. (Смотрите раздел “Срок ожидания блокировки (locktimeout)” на стр. 409.)

Использование этого параметра помогает предотвращать глобальные тупиковые ситуации, особенно в программах с распределенными единицами работы. Если срок ожидания блокировки истечет, то есть длительность отложенного состояния требования блокировки превысит значение *locktimeout*, программа получит сообщение об ошибке, и будет выполнен откат транзакции. Например, если программа *program1* пытается получить блокировку, которая уже удерживается программой *program2*, в случае истечения срока ожидания *program1* вернет SQLCODE -911 (SQLSTATE 40001) с кодом причины 68.

Если параметр *diaglevel* конфигурации менеджера баз данных задан равным четырем, при истечении срока ожидания блокировки в db2diag.log помещается дополнительная информация. Эта информация содержит объект, режим блокировки и программу, удерживающую блокировку в отношении объекта. Там же может находиться текущий динамический оператор SQL или имя статического пакета.

Тупиковые ситуации

В менеджере баз данных блокировки, инициируемые процессами баз данных, могут привести к тупиковой ситуации. Пусть, например, процесс 1 блокирует таблицу A в режиме X (exclusive - монопольный), а процесс 2 блокирует в режиме X таблицу B; если затем процесс 1 попытается заблокировать в режиме X таблицу B, а процесс 2 попытается заблокировать в режиме X таблицу A, процессы попадут в тупиковую ситуацию. В тупиковой ситуации каждый из двух процессов остановлен до получения второй затребованной блокировки, но ни одно из этих требований не будет удовлетворено, пока один из процессов не выполнит принятия или отката. Эта ситуация сохраняется до бесконечности, пока внешний агент не активирует один из процессов, заставляя его выполнить откат.

Тупиковые ситуации в системе блокировок разрешаются в менеджере баз данных асинхронным системным фоновым процессом, называемым детектором тупиковых ситуаций. Детектор тупиковых ситуаций активируется периодически, как задано параметром конфигурации *dlchktime* (смотрите раздел “Интервал проверки тупиковых ситуаций (dlchktime)” на стр. 406). Активировавшись, детектор тупиковых ситуаций обследует систему на наличие тупиковых ситуаций. Если база данных многораздельная, каждый раздел посылает *схемы*

блокировок в тот раздел базы данных, в котором находятся производные таблицы системного каталога и где происходит глобальное обнаружение тупиковых ситуаций.

Если обнаружена тупиковая ситуация, детектор выбирает один из тупиковых процессов для отката. Выбранный процесс возобновляется и возвращает вызвавшей его прикладной программе код SQLCODE -911 (SQLSTATE 40001) к кодом причины 2. Менеджер баз данных автоматически выполняет откат выбранного процесса. По завершении отката принадлежащие выбранному процессу блокировки снимаются, и остальные процессы, участвовавшие в тупиковой ситуации, получают, наконец, возможность продолжить выполнение.

Выбор подходящего интервала для детектора тупиковых ситуаций необходим для обеспечения хорошей производительности. Слишком короткий интервал приведет к лишним служебным расходам, а слишком длинный позволит тупиковым ситуациям затянуть процесс на неприемлемо долгий срок. Например, задание интервала возобновления 30 минут может позволить тупиковой ситуации просуществовать почти 30 минут. Разработчик прикладной программы должен сбалансировать возможные задержки при решении тупиковых ситуаций и служебные расходы по их обнаружению.

В многораздельной базе данных интервал должен быть одинаков во всех разделах (при изменении параметра конфигурации *dlchktime* нужно задавать одно значение для всех разделов). Если значение в каталоге узла меньше, чем в других разделах, возможно обнаружение фантомных тупиковых ситуаций. Если значение в каталоге узла больше, чем в других разделах, может оказаться, что до обнаружения тупиковой ситуации пройдет больше двух интервалов. Если в многораздельной базе данных обнаруживается большое число тупиковых ситуаций, нужно увеличить значение параметра *dlchktime*, учтя ожидание блокировки и ожидание связи.

Другая проблема может возникать, когда программа с несколькими независимыми процессами, обращающимися к базе данных, имеет структуру, повышающую вероятность тупиковых ситуаций. Примером является прикладная программа, в которой несколько процессов обращаются к одной и той же таблице для чтения и затем для записи. Если процессы вначале выполняют запросы SQL только для чтения, а затем на той же таблице - изменения SQL, вероятность возникновения тупиковых ситуаций растет из-за потенциального конфликта между процессами по поводу общих данных. Например, если два процесса читают таблицу, а затем изменяют ее, наступает стадия, на которой процесс A пытается получить блокировку типа X строки, в отношении которой процесс B удерживает блокировку типа S, и наоборот. В результате может возникнуть тупиковая ситуация. Чтобы предотвратить такие тупиковые ситуации, прикладные программы, которые обращаются к данным с намерением изменить их, должны при выполнении выборки использовать

условие FOR UPDATE OF. Это условие позволяет убедиться, что, когда процесс A пытается прочитать данные, устанавливается блокировка типа U.

Примечание: Можно определить монитор, который будет записывать случаи тупиковых ситуаций. Для создания монитора используйте оператор CREATE EVENT, описанный в справочнике *SQL Reference*.

В среде системы объединения, когда прикладная программа обращается к псевдонимам, может случиться так, что данные, затребованные прикладной программой, недоступны из-за тупиковой ситуации на источнике данных. В таком случае для снятия блокировки DB2 использует функции обработки тупиковых ситуаций. В случае тупиковой ситуации, затрагивающей несколько источников данных, DB2 для снятия блокировки использует механизм истечения срока ожидания на источнике данных.

Если для параметра *diaglevel* конфигурации менеджера баз данных задано значение 4, и затребованная блокировка не получена из-за тупиковой ситуации, в db2diag.log записывается дополнительная информация. Эта информация содержит объект, режим блокировки и программу, удерживающую блокировку в отношении объекта. Там же может находиться текущий динамический оператор SQL или имя статического пакета.

Факторы, влияющие на блокировку

Режимы и размеры блокировок менеджера баз данных определяются сочетанием ряда факторов: типом обработки, выполняемой прикладной программой, ее методом доступа к данным и несколькими параметрами, которые вы можете задавать.

Обработка прикладной программы

Чтобы определить атрибуты блокировки, все виды обработки делят на четыре типа:

Только для чтения

Этот тип включает все операторы выборки, которые по своей природе предназначены только для чтения (информацию об указателях смотрите в руководстве *SQL Reference*), или имеют явное условие FOR READ ONLY, или не имеют четкого определения, но компилятор SQL предполагает, что эти операторы только для чтения из-за значения опции BLOCKING, заданного в командах PREP или BIND. Он требует только блокировок для совместного использования (S или IS).

Намерение изменения

Этот тип включает все операторы выборки с условием FOR UPDATE и те нечетко определенные операторы, которые компилятор SQL по результатам интерпретации предполагает предназначенными для

изменения. Он использует блокировки для совместного доступа и обновления (S, U и X для строк, IX, U, X для таблиц).

Изменение
Этот тип включает UPDATE, INSERT и DELETE, но не UPDATE WHERE CURRENT OF или DELETE WHERE CURRENT OF. Он требует монопольных блокировок (X или IX).

Управляемые указателем
Этот тип включает UPDATE WHERE CURRENT OF и DELETE WHERE CURRENT OF. Он также требует монопольных блокировок (X или IX).

Оператор, который производит в таблице назначения вставки, изменения и удаления, зависящие от результата подоператора выборки, выполняет обработку двух типов. Для таблиц, возвращаемых подвыборкой, блокировки определяются правилами обработки только для чтения; для таблиц назначения - правилами обработки с изменениями.

Пути доступа

Путь доступа - это метод получения данных, который программа оптимизации выбирает для конкретного обращения к таблице. (Смотрите раздел “Концепции доступа к данным и оптимизация” на стр. 169.) Путь доступа, выбираемый программой оптимизации, может существенно повлиять на режимы блокировок. Например, когда при помощи просмотра индекса производится поиск конкретной строки, программа оптимизации вполне может выбрать для таблицы блокировку уровня строк (IS). Этот тип доступа будет применен при выборке информации по одному сотруднику из таблицы сотрудников EMPLOYEE, имеющей индекс по номеру сотрудника (EMPNO), которая заказана таким оператором:

```
SELECT *
   FROM EMPLOYEE
  WHERE EMPNO = '000310';
```

Аналогичным образом, когда индекс не используется и, чтобы найти выбранные строки, приходится просматривать всю таблицу, она может получить одну блокировку уровня таблицы (S). Например, этот тип доступа может быть применен для выборки всех сотрудников мужского пола, когда для столбца пола SEX нет индекса, а выборка заказана таким оператором:

```
SELECT *
   FROM EMPLOYEE
  WHERE SEX = 'M';
```

В следующих таблицах дается обзор блокировок, получаемых при тех или иных видах плана доступа. Определения заголовков столбцов смотрите в разделе “Обработка прикладной программы” на стр. 63. Об определении метода доступа смотрите в разделе “Концепции доступа к данным и оптимизация” на стр. 169. Заметьте, что тип обработки *под управлением указателя* использует режим

блокировки базового указателя, пока прикладная программа не найдет строку, которую нужно изменить или удалить. Операция этого типа, независимо от режима блокировки указателя, всегда получает монопольную блокировку для выполнения изменения или удаления.

В следующих таблицах там, где приведен только один режим блокировки, имеется в виду режим блокировки уровня таблицы. Если приведены два режима блокировки, первый - режим блокировки уровня таблицы, а второй - режим блокировки уровня строки.

Таблица 5. Режимы блокировки при просмотре таблиц

Уровень изоляции	Только для чтения	Намерение изменения	Изменение
Метод доступа: ***Просмотр таблицы без предикатов***			
RR	S	U	X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
Метод доступа: ***Просмотр таблицы с предикатами***			
RR	S	U	U
RS	IS / NS	IX / U	IX / U
CS	IS / NS	IX / U	IX / U
UR	IN	IX / U	IX / U

Таблица 6. Режимы блокировки при просмотре индексов

Уровень изоляции	Только для чтения	Намерение изменения	Изменение
Метод доступа: ***Просмотр индекса без предикатов***			
RR	S	IX / U	X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
Метод доступа: ***Просмотр индекса без предикатов***			
RR	IS / S	IX / U	IX / X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
Метод доступа: ***Просмотр индекса только с предикатами запуска и остановки***			
RR	IS / S	IX / S	IX / X

Таблица 6. Режимы блокировки при просмотре индексов (продолжение)

Уровень изоляции	Только для чтения	Намерение изменения	Изменение
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
Метод доступа: ***Просмотр индекса с предикатами***			
RR	IS / S	IX / S	IX / U
RS	IS / NS	IX / U	IX / U
CS	IS / NS	IX / U	IX / U
UR	IN	IX / U	IX / U

В Табл. 7 даны режимы блокировки для случаев, в которых чтение страниц данных отложено, чтобы список строк можно было:

- Уточнить при помощи нескольких индексов. Дополнительную информацию смотрите в разделе “Доступ к нескольким индексам” на стр. 176.
- Отсортировать для ускорения предварительной выборки. Дополнительную информацию смотрите в разделе “Как работает предварительная выборка списка” на стр. 272.

Отложенный доступ к страницам данных подразумевает, что доступ к строке производится в два этапа и, следовательно, с более сложными сценариями блокировки. Есть две главных категории, которые зависят от уровня изоляции. Уровень изоляции многократного чтения сохраняет все полученные блокировки до завершения транзакции, поэтому блокировки, полученные на первом этапе, удерживаются, и на втором этапе требовать блокировок не приходится. При уровнях изоляции стабильность чтения и стабильность на уровне указателя на втором этапе получать блокировки нужно. Для максимального параллелизма блокировки не устанавливаются на первом этапе, вместо этого все предикаты применяют снова, чтобы обеспечить возврат лишь подходящих строк.

Таблица 7. Режимы блокировки при просмотре индексов, используемые для отложенного доступа к страницам данных

Уровень изоляции	Только для чтения	Намерение изменения	Изменение
Метод доступа: ***Просмотр индекса без предикатов***			
RR	IS / S	IX / S	X
RS	IN	IN	IN
CS	IN	IN	IN
UR	IN	IN	IN

Метод доступа: ***Отложенный доступ к страницам данных после просмотра индекса без предикатов***

Таблица 7. Режимы блокировки при просмотре индексов, используемые для отложенного доступа к страницам данных (продолжение)

Уровень изоляции	Только для чтения	Намерение изменения	Изменение
RR	IN	IX / S	X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
Метод доступа: ***Просмотр индекса с предикатами***			
RR	IS / S	IX / S	IX / S
RS	IN	IN	IN
CS	IN	IN	IN
UR	IN	IN	IN
Метод доступа: ***Просмотр индекса только с предикатами запуска и остановки***			
RR	IS / S	IX / S	IX / X
RS	IN	IN	IN
CS	IN	IN	IN
UR	IN	IN	IN
Метод доступа: ***Отложенный доступ к страницам данных после просмотра индекса с предикатами***			
RR	IN	IX / S	IX / S
RS	IS / NS	IX / U	IX / U
CS	IS / NS	IX / U	IX / U
UR	IN	IX / U	IX / U

Путь доступа не контролируется пользователем; он выбирается программой оптимизации.

Используемый путь доступа может влиять на режим и размер блокировки. Например, в программе, использующей уровень изоляции RR (repeatable read - многократное чтение), запрос UPDATE, который использует просмотр таблицы без предикатов, применит блокировку X в отношении таблицы. Если строки найдены при помощи индекса, менеджер баз данных может выбрать блокировку отдельных строк таблицы.

Объявленные временные таблицы и блокировка

Объявленные временные таблицы не блокируются, поскольку доступны только той прикладной программе, которая их объявила. Этот тип таблицы существует только с момента объявления ее прикладной программой и до момента завершения или отключения программы.

Оператор LOCK TABLE

Правила получения первоначальных режимов блокировки можно переопределить в прикладной программе оператором LOCK TABLE.

Этот оператор блокирует всю таблицу. Блокируется только таблица, указанная в операторе LOCK TABLE. Таблицы, родительские или зависимые от заданной, не блокируются. Нужно с учетом параллелизма и производительности решить, следует ли блокировать доступ к другим таблицам. Блокировка не освобождается, пока для единицы работы не будет выполнено принятие или откат.

Если с некоторой таблицей обычно работают несколько пользователей, ее блокировка может быть желательна по следующим причинам:

LOCK TABLE IN SHARE MODE

Желательно иметь доступ к данным, *согласованным на данный момент*; другими словами, данным таблицы, актуальным на данный момент времени. Если с таблицей ведется интенсивная работа, единственный способ обеспечить поддержание устойчивости всей таблицы - блокировать ее. Например, прикладная программа хочет сделать снимок таблицы. Но когда эта программа обрабатывает одни строки таблицы, другие программы изменяют еще не обработанные строки. Это разрешается в режиме многократного чтения, но в данном случае нежелательно.

Альтернативой служит включение в программу оператора LOCK TABLE IN SHARE MODE: изменение строк станет невозможным, независимо от того, извлечены они или нет. Тогда можно извлечь сколь угодно большое число строк, не опасаясь, что эти строки будут изменены перед самым извлечением.

При режиме LOCK TABLE IN SHARE MODE другие пользователи могут извлекать данные из таблицы, но не могут изменять, удалять или вставлять строки в таблицу.

LOCK TABLE IN EXCLUSIVE MODE

Допустим, вы хотите изменить значительную часть таблицы. Дешевле и быстрее запретить всем остальным пользователям доступ к таблице, чем блокировать каждую изменяемую строку, а затем снимать блокировку строки, после того как все изменения будут приняты.

При режиме LOCK TABLE IN EXCLUSIVE MODE все остальные пользователи блокируются; все остальные прикладные программы лишаются доступа к таблице, за исключением прикладных программ с чтением непринятого.

Дополнительные подробности об операторе LOCK TABLE смотрите в руководстве *SQL Reference*.

Альтернативой оператору LOCK TABLE служит оператор ALTER TABLE с параметром LOCKSIZE. Параметр LOCKSIZE позволяет выбрать блокировку строк (ROW) или таблиц (TABLE). Указанный размер определит выбор блокировки при следующем обращении к таблице. Блокировки строк ничем не отличаются от блокировок с размером, принимаемым по умолчанию при создании таблицы. Использование блокировки таблиц может повысить производительность запросов благодаря уменьшению числа получаемых блокировок. С другой стороны, при этом может снизиться параллелизм, поскольку все блокировки будут удерживаться в отношении всей таблицы. Ни тот, ни другой размер не препятствует обычному расширению блокировок. Дополнительные подробности об операторе ALTER TABLE смотрите в справочнике *SQL Reference*.

CLOSE CURSOR WITH RELEASE

Когда вы закрываете указатель оператором CLOSE CURSOR, содержащим условие WITH RELEASE, менеджер баз данных пытается освободить все блокировки чтения (если такие есть), удерживавшиеся для этого указателя. Блокировки чтения - это блокировки таблицы в режиме IS, S и U, а также блокировки строки в режиме S, NS и U. Дополнительную информацию о режимах блокировки смотрите в разделе “Атрибуты блокировок” на стр. 53.

Условие WITH RELEASE никак не влияет на указатели, работающие с уровнями изоляции CS или UR. Если условие WITH RELEASE задается для указателей, работающих с уровнями изоляции RS или RR, оно аннулирует некоторые гарантии этих уровней изоляции. А именно, для указателя RS можно получить *невоспроизводимое чтение*, а для указателя RR, кроме того, *чтение фантомных строк*.

Если указатель, открытый как RR или RS, повторно открывается после закрытия с условием WITH RELEASE, происходит получение новых блокировок чтения.

Это и другое первичное условие для оператора CLOSE CURSOR сопоставляются в разделе “Оператор DECLARE CURSOR WITH HOLD” на стр. 82.

В DB2 CLI прикладные программы CLI могут при помощи атрибута соединения SQL_ATTR_CLOSE_BEHAVIOR достигать тех же результатов, что и CLOSE CURSOR WITH RELEASE. Дополнительную информацию смотрите в разделе о `SQLSetConnectAttr()` руководства *CLI Guide and Reference*.

Сводка особенностей блокировки

О блокировке следует помнить следующее:

- Малые единицы работы (частые операторы COMMIT) способствуют одновременному доступу к данным большого числа пользователей. Применяйте операторы COMMIT, когда ваша прикладная программа логически достигает точки согласованности, то есть согласованы измененные

вами данные. После оператора COMMIT блокировки освобождаются (кроме блокировок таблиц, связанных с указателем, который объявлен с опцией WITH HOLD).

- Получение блокировок происходит даже тогда, когда прикладная программа всего лишь читает строки, так что принятие имеет смысл и в отношении единиц работы только для чтения. Это объясняется тем, что в программах только для чтения на уровнях изоляции многократного чтения, стабильности чтения и стабильности на указателе происходит получение блокировок совместного доступа. На уровне многократного чтения и стабильности чтения все блокировки удерживаются до выполнения COMMIT, что запрещает другим программам изменять заблокированные данные, если только вы не закроете указатель при помощи условия WITH RELEASE. Кроме того, получение блокировок каталога происходит даже в программах с чтением непринятого, использующих динамические SQL.
- Менеджер баз данных следит за тем, чтобы ваша прикладная программа не извлекала непринятых данных (строк, измененных другими программами, но еще не принятых), если вы не пользуетесь уровнем изоляции, который разрешает чтение непринятого.
- Вы можете заблокировать всю таблицу, которую хотите защитить, при помощи оператора LOCK TABLE:
  - Чтобы разрешить другим прикладным программам получать, но не обновлять, удалять или вставлять строки
  - Чтобы запретить другим программам (кроме программ с уровнем изоляции с чтением непринятого) доступ к строкам таблицы.
- Когда вы закрываете указатель оператором CLOSE CURSOR, содержащим условие WITH RELEASE, менеджер баз данных пытается освободить все блокировки чтения (если такие есть), удерживавшиеся для этого указателя.
- Изменяя параметры конфигурации, влияющие на блокировку в многораздельной базе данных, убедитесь, что изменения производятся во всех разделах базы данных.

Настройка класса оптимизации

Есть ряд методов оптимизации, при помощи которых во время компиляции запроса SQL можно построить эффективный план доступа для этого запроса. Использование большего числа методов оптимизации приводит к:

1. повышению производительности выполнения
2. большему времени компиляции запроса
3. большему использованию ресурсов системы.

По этим соображениям может оказаться желательным ограничить число применяемых методов оптимизации запроса, для чего и служит задание класса оптимизации. Это может оказаться особенно полезным при:

- Очень маленьких базах данных или очень простых динамических запросах

- Ограниченном объеме доступной памяти во время компиляции на сервере баз данных
- Желании уменьшить время компиляции запроса (например, время выполнения PREPARE).

Можно выбрать любой из описанных ниже классов оптимизации, хотя класс 0 и класс 9 следует использовать только в особых обстоятельствах. Класс 5 принимается по умолчанию. Классы 0, 1 и 2 используют алгоритм ″жадного″ перебора вариантов объединения; для сложных запросов этот алгоритм рассматривает намного меньше вариантов плана и, следовательно, существенно уменьшает время компиляции по сравнению с классами 3 и выше. Классы 3 и выше используют алгоритм динамического программирования перебора методов объединения; этот алгоритм рассматривает намного больше вариантов плана и, следовательно, с ростом числа таблиц существенно увеличивает время компиляции по сравнению с классами 0, 1 и 2.

0 - Этот класс указывает программе оптимизации использовать минимальный объем оптимизации при генерации плана доступа. Например:
- Программа оптимизации не рассматривает статистику неоднородности распределения.
- Применяются лишь основные правила перезаписи запроса (дополнительная информация о перезаписи запроса приведена в разделе “Перезапись запросов компилятором SQL” на стр. 159).
- Выполняется ″жадный″ перебор методов объединения (смотрите раздел “Стратегии поиска для выбора оптимального объединения” на стр. 188).
- Из методов доступа разрешены только объединение с вложенным циклом и просмотр индекса (смотрите разделы “Концепции объединения” на стр. 182 и “Концепции просмотра индекса” на стр. 170).
- Предварительная выборка списка и операция AND над индексами запрещены и не используются в генерируемых методах доступа.
- Стратегия объединения типа ″звезда″ не рассматривается.

Этот класс следует использовать только в особых обстоятельствах, когда требуется минимизировать затраты на компиляцию запроса. Хорошим примером, когда уместен класс оптимизации запроса 0, служит прикладная программа, целиком состоящая из очень простых динамических операторов SQL, которые обращаются к хорошо индексированным таблицам.

1 - Этот класс указывает программе оптимизации использовать степень оптимизации, которая примерно соответствует DB2/6000 Версии 1 плюс несколько функций с низкой стоимостью, которых не было в Версии 1. В частности:

- Программа оптимизации не рассматривает статистику неоднородности распределения.
- Применяется только подмножество набора правил для перезаписи запросов, включающее те правила, которые поддерживались в DB2/6000 Версии 1.
- Используется объединение с ″жадным″ перебором методов (смотрите раздел “Стратегии поиска для выбора оптимального объединения” на стр. 188.)
- Предварительная выборка списка и операция AND над индексами запрещены и не используются в генерируемых методах доступа.

Примечание: Операция AND над индексами все же используется при работе с полу-объединениями, связанными с объединениями типа ″звезда″.

Класс оптимизации 1 во всем аналогичен классу 0, за исключением того, что становятся доступны также объединения просмотром со слиянием и просмотр таблиц.

2 - Этот класс указывает программе оптимизации использовать степень оптимизации, которая существенно превосходит оптимизацию класса 1, но в то же время сохраняет стоимость компилирования сложных запросов на существенно более низком уровне, чем в классах 3 и старше. В частности:

- Используется вся доступная статистика, включая статистику частот и квантилей неравномерного распределения.
- Применяются все правила перезаписи запроса, включая маршрутизацию запросов к сводным таблицам, кроме вычислительно трудоемких правил, применимых в весьма редких случаях.
- Используется объединение с ″жадным″ перебором методов.
- Рассматривается широкий диапазон методов доступа, включая предварительную выборку списка и маршрутизацию сводных таблиц.
- Рассматривается, в допустимых случаях, стратегия объединения типа ″звезда″.

Класс оптимизации 2 во всем аналогичен классу 5, за исключением того, что использует ″жадный″ перебор методов объединения вместо динамического программирования. В этом классе достигается максимальная оптимизация среди всех классов оптимизации, использующих алгоритм ″жадного″ перебора методов объединения, при котором рассматривается меньше вариантов для сложных запросов и, следовательно, тратится меньше времени на компиляцию, чем в классе 3 и выше. Поэтому он рекомендуется для очень сложных запросов в среде поддержки решений или среде OLAP (online analytic processing, оперативный анализ данных). В этих случаях велика вероятность, что

один и тот же запрос будет повторяться нечасто, и его план доступа вряд ли останется в кэше до очередного повторения запроса.

3 - Этот класс требует среднего объема оптимизации при генерации плана доступа. Этот класс ближе всех по своим характеристикам к оптимизации запроса в DB2 for MVS/ESA или OS/390. Характерные особенности этого класса оптимизации:

- Используется, в допустимых случаях, статистика неравномерности распределения, которая отслеживает часто используемые значения.
- Применяется большинство правил перезаписи запроса, включая преобразования подзапроса в объединение.
- Используется перебор методов объединения с динамическим программированием (смотрите раздел “Стратегии поиска для выбора оптимального объединения” на стр. 188):
  - Ограниченно используются составные внутренние таблицы (смотрите раздел “Составные таблицы” на стр. 190)
  - Ограниченно используются декартовы произведения для схем типа ″звезда″, включая таблицы “просмотра” (смотрите раздел “Стратегии поиска для объединения типа ″звезда″” на стр. 188)
- Рассматривается широкий диапазон методов доступа, включая предварительную выборку списка, операцию AND над индексами и объединения типа ″звезда″.

Этот класс подходит для широкого диапазона прикладных программ. Использование этого класса повышает вероятность того, что программа оптимизации выберет отличный план доступа для запросов с четырьмя и более объединениями. Однако программа оптимизации, возможно, не сможет рассмотреть еще лучшие планы, которые могли бы быть выбраны для класса оптимизации по умолчанию.

5 - Этот класс указывает программе оптимизации использовать значительный объем оптимизации при генерации плана доступа. В частности, для класса 5 применяются:

- Вся доступная статистика, включая статистику частот и квантилей для неравномерного распределения.
- Все правила перезаписи запроса, включая маршрутизацию запросов к сводным таблицам, кроме вычислительно трудоемких правил, применимых в весьма редких случаях.
- Используется перебор методов объединения с динамическим программированием (смотрите раздел “Стратегии поиска для выбора оптимального объединения” на стр. 188):
  - Ограниченно используются составные внутренние таблицы (смотрите раздел “Составные таблицы” на стр. 190)

- Ограниченно используются декартовы произведения для схем типа ″звезда″, включая таблицы “просмотра” (смотрите раздел “Стратегии поиска для объединения типа ″звезда″” на стр. 188)
- Рассматривается широкий диапазон методов доступа, включая предварительную выборку списка, операцию AND над индексами и маршрутизацию сводных таблиц.

Когда программа оптимизации обнаруживает, что дополнительные ресурсы и время обработки для сложных динамических запросов SQL не гарантированы, объем оптимизации сокращается. Степень уменьшения зависит от размера компьютера и числа предикатов.

Когда программа оптимизации запроса уменьшает объем выполняемой оптимизации запроса, она продолжает применять все правила переписывания запроса, которые применялись бы в норме. Но она применяет метод ″жадного″ перебора методов объединения и сокращает число рассматриваемых сочетаний плана доступа.

Класс оптимизации запроса 5 - отличный выбор для смешанной среды, содержащей и транзакции, и сложные запросы. Этот класс оптимизации предназначен для эффективного применения наиболее дорогостоящих преобразований запроса и других методов оптимизации запроса.

7 - Этот класс указывает программе оптимизации использовать значительный объем оптимизации при генерации плана доступа. Он во всем аналогичен классу оптимизации запроса 5, за исключением того, что не уменьшает объем оптимизации запроса для сложных динамических запросов SQL.

9 - Этот класс указывает программе оптимизации использовать все доступные методы оптимизации. В их число входят:

- Все доступные статистические показатели
- Все правила перезаписи запроса
- Все возможности перебора методов объединения, включая декартовы произведения и неограниченные составные внутренние таблицы
- Все методы доступа.

Этот класс может серьезно увеличить число рассматриваемых программой оптимизации вариантов плана доступа. Этот класс следует использовать, чтобы выяснить, может ли более тщательная оптимизация сгенерировать лучший план доступа для очень сложных запросов и запросов с очень большим временем выполнения, использующих большие таблицы. Чтобы проверить, действительно ли найденный план лучше, следует использовать показ объяснений и измерения производительности.

Как задать класс оптимизации?

Способ задания определенного класса оптимизации запроса отличается для статических и динамических SQL.

- *Статические операторы SQL* используют класс оптимизации, заданный в командах PREP и BIND. Класс оптимизации при связывании пакетов записан в столбце QUERYOPT в таблице каталога SYSCAT.PACKAGES. Если пакет подвергается повторному связыванию неявно или при помощи команды REBIND PACKAGE, для статических операторов SQL будет использован тот же класс оптимизации. Если вы хотите изменить класс оптимизации, используемый для этих статических операторов SQL, нужно использовать команду BIND. Если не задать класс оптимизации, DB2 будет использовать оптимизацию по умолчанию, заданную параметром конфигурации базы данных *dft_queryopt*.
- *Динамические операторы SQL* используют класс оптимизации, заданный в специальном регистре CURRENT QUERY OPTIMIZATION при помощи оператора SQL SET. Например, следующий оператор задает класс оптимизации 1:

  ```
  SET CURRENT QUERY OPTIMIZATION = 1
  ```

  Чтобы гарантировать, что динамический оператор SQL будет всегда использовать один и тот же класс оптимизации, можно включить этот оператор SET в текст прикладной программы. Дополнительную информацию смотрите в справочнике *SQL Reference*.

  Если специальный регистр CURRENT QUERY OPTIMIZATION не задан, динамические операторы будут связываться при помощи класса оптимизации запроса по умолчанию. Значение по умолчанию и для динамических, и для статических операторов SQL определяется значением параметра конфигурации базы данных *dft_queryopt*. Класс оптимизации запроса 5 принимается по умолчанию, если вы не задали другого. (Более подробную информацию об этом параметре смотрите в разделе “Класс оптимизации запросов по умолчанию (dft_queryopt)” на стр. 470.) Значения по умолчанию для опции связывания и этого специального регистра берутся из параметра конфигурации *dft_queryopt*.

Необходимый объем оптимизации

Большинство операторов достаточно хорошо оптимизируются с разумным расходом ресурсов при классе оптимизации запроса по умолчанию. Время компиляции запроса и расходование ресурсов при заданном классе оптимизации в первую очередь зависит от сложности запроса, особенно числа объединений и подзапросов. Но время компиляции и использование ресурсов также зависят от объема оптимизации, выполняемого при разных классах оптимизации. При том или ином классе оптимизации различия во времени компиляции запроса и расходовании ресурсов скорее можно ожидать для очень сложного запроса, чем для простого.

Выбирая класс оптимизации, можно:

- Начать с использования класса оптимизации запроса по умолчанию.
- Если желательно использовать класс, отличный от класса по умолчанию, попробуйте вначале классы 1, 2 или 3.
- Низшие классы оптимизации (0 и 1) используйте для запросов, имеющих очень малое время выполнения, то есть запросов, занимающих менее одной секунды. (В дальнейшем описании приводятся дополнительные критерии для выбора низкого класса оптимизации.)
- Классы оптимизации 1 и 2 используйте в ситуации большого числа таблиц с большим числом предикатов объединения в одном и том же столбце, если при этом необходимо учитывать и время компиляции.
- Высшие классы оптимизации (3, 5 и 7) используйте для запросов с долгим выполнением, то есть запросов, занимающих более 30 секунд.
- При нормальных обстоятельствах не используйте класс оптимизации 9.
- Для запросов, которые выполняются долго, выполните запрос при помощи db2batch, чтобы определить, какая часть времени потрачена на компиляцию, а какая - на выполнение.
  - Если большая часть времени тратится на компиляцию, понизьте класс оптимизации.
  - Если большая часть времени тратится на выполнение, попробуйте более высокий класс оптимизации.

Заметим, что классы оптимизации запроса 1, 2, 3, 5 и 7 подходят для общего применения.

Класс 0 имеет смысл пробовать, только если вам требуется дальнейшее уменьшение времени компиляции запроса и известен тип SQL (например, предельно простые операторы). Обычно такие SQL характеризуются тем, что:
- Обращаются к одной или небольшому числу таблиц
- Производят выборку одной или небольшого числа строк
- Используют полностью заданные индексы уникальности.

Хорошим примером SQL этого типа служат транзакции OLTP (Online transaction processing - оперативная обработка транзакций).

Сложные запросы могут потребовать выбора других объемов оптимизации для получения наилучшего плана доступа. Возможно, вы захотите опробовать высшие классы оптимизации для запросов, для которых характерны:
- Обращение к большим таблицам
- Большое число предикатов
- Многочисленные подзапросы
- Многочисленные объединения
- Многочисленные операции над множествами, такие как UNION и INTERSECT
- Большое число отобранных строк

- Операции GROUP BY и HAVING
- Вложенные табличные выражения
- Большое число производных таблиц

Запросы для поддержки решений и запросы для месячных отчетов в отношении полностью нормализованных баз данных служат хорошими примерами сложных запросов, для которых следует использовать класс оптимизации запроса по умолчанию и выше.

Другой причиной для использования высоких классов оптимизации запроса служат SQL, порожденные генератором запросов. Многие генераторы запросов создают неэффективные SQL. Плохо написанные запросы, включая порожденные генератором запросов, могут потребовать дополнительной оптимизации для выбора хорошего плана доступа. Использование класса оптимизации запросов 2 и выше может улучшить плохо написанные запросы SQL.

Важно учитывать также различие между статическими и динамическими SQL, а также многократное повторение одного и того же динамического SQL. В случае статического SQL время и расход ресурсов на компиляцию запроса увеличиваются лишь один раз, после чего полученный план доступа может использоваться много раз. Как правило, для статических SQL не следует снижать класс оптимизации запроса по умолчанию. Динамические операторы связываются и выполняются после запуска; поэтому есть смысл проверить, действительно ли расходы на дополнительную оптимизацию динамических операторов повышают общую производительность. Но если один и тот же динамический оператор SQL выполняется многократно, выбранный план доступа попадет в кэш. С точки зрения выбора класса оптимизации запроса такой оператор можно приравнять к статическому оператору SQL.

(Посмотрите в руководстве *Application Development Guide*, когда использовать статический и динамический SQL.)

Если вы предполагаете, что имеете дело с запросом, который может улучшиться от дополнительной оптимизации, но не уверены в этом, или если вам приходится учитывать время компиляции и использование ресурсов, можно провести тестовые измерения. Тестирование может помочь сделать количественную оценку выгод, получаемых при различных классах оптимизации. В разделе “Глава 12. Измерение производительности” на стр. 335 описаны общие методы и особенности применения `db2batch`. При разработке и проведении тестовых измерений учитывайте разницу между статическими и динамическими операторами SQL в прикладной программе:

- Для **динамических** операторов SQL тестирование должно сравнивать среднее время выполнения оператора. Для вычисления среднего времени выполнения можно использовать следующую формулу:

```
время компиляции + суммарное время выполнения во всех повторах
---------------------------------------------------------------
                      число повторов
```

где число повторов означает ожидаемое число выполнений оператора SQL после одной компиляции.

Примечание: После первой компиляции динамические операторы SQL перекомпилируются, если этого требуют изменения в среде. После помещения в кэш оператор SQL не требует повторной компиляции, поскольку последующие операторы PREPARE будут повторно использовать оператор из кэша при условии, что среда не изменилась. (Информация о том, как кэш может повысить производительность при работе с динамическими операторами SQL, приведена в разделах “Размер кэша каталога (catalogcache_sz)” на стр. 370 и “Размер кэша пакета (pckcachesz)” на стр. 378.)

- Для **статических** операторов SQL тестирование должно сравнивать время выполнения оператора.

  Примечание: Хотя можно поинтересоваться и временем компиляции статического SQL, трудно представить ситуацию, в которой имело бы смысл общее время (компиляции и выполнения) оператора. Сравнение общего времени не показывает того факта, что статический оператор SQL можно запускать много раз после одного связывания, и что обычно он не подвергается связыванию при выполнении.

Ограничение наборов результатов для повышения производительности

Оператор SELECT задает набор строк, удовлетворяющих критериям поиска. Программа оптимизации DB2 предполагает, что программа получит все отобранные строки. Это предположение вполне оправдано в среде OLTP и пакетных средах. Однако в программах просмотра (“браузерах”) запросы обычно определяют очень большой потенциальный набор ответов, из которых программа реально получает только первые несколько строк, в типичном случае - столько строк, сколько требуется для заполнения экрана.

Предполагаемое по умолчанию программой оптимизации извлечение всех отобранных строк - не лучший вариант для прикладных программ, которые не изменяют и не удаляют информацию из хранимых данных.

Есть пять способов модификации оператора SELECT, позволяющих ограничить или изменить таблицу результатов, чтобы повысить производительность. Это:

- Условие FOR UPDATE
- Условие FOR READ/FETCH ONLY

- Условие OPTIMIZE FOR n ROWS
- Условие FETCH FIRST n ROWS ONLY
- Оператор DECLARE CURSOR WITH HOLD.

Условие FOR UPDATE

Условие FOR UPDATE задает столбцы, которые может изменять последующий оператор UPDATE. Если условие FOR UPDATE задано без имен столбцов, оно распространяется на все изменяемые столбцы таблицы или производной таблицы. Если имена столбцов заданы, каждое имя (без спецификатора) должно задавать столбец таблицы или производной таблицы.

Условие FOR UPDATE нельзя использовать, когда верно хотя бы одно из следующих утверждений:

- Указатель, связанный с оператором SELECT, не может быть удален.
- Хотя бы один из указанных столбцов является столбцом, который не может изменяться в таблице каталога и не был исключен условием FOR UPDATE.

Прикладные программы CLI могут достигать тех же результатов при помощи атрибута соединения DB2 CLI SQL_ATTR_ACCESS_MODE. Дополнительную информацию смотрите в разделе о `SQLSetConnectAttr()` руководства *CLI Guide and Reference*.

Условия FOR READ и FETCH ONLY

Условие FOR READ ONLY гарантирует, что таблица результатов будет использоваться только для чтения. Условие FOR FETCH ONLY означает то же самое.

Некоторые таблицы результатов являются таблицами только для чтения по определению. Например, таблица результатов оператора SELECT для производной таблицы, заданной как таблица только для чтения. В таких случаях условие FOR READ ONLY задать можно, но оно ни на что не повлияет.

Для таблиц результатов, в которых разрешены изменения и удаления, задание условия FOR READ ONLY может повысить производительность операций выборки FETCH. Это повышение производительности достигается тогда, когда менеджер баз данных может использовать блокирование данных вместо монопольных блокировок. Следует использовать условие FOR READ ONLY для повышения производительности во всех случаях, где запросы используются не для позиционных операторов UPDATE или DELETE.

Прикладные программы CLI могут достигать тех же результатов при помощи атрибута соединения DB2 CLI SQL_ATTR_ACCESS_MODE. Дополнительную информацию смотрите в разделе о `SQLSetConnectAttr()` руководства *CLI Guide and Reference*.

Условие OPTIMIZE FOR n ROWS

Условие OPTIMIZE FOR обеспечивает механизм, при помощи которого прикладная программа объявляет о намерении получить лишь подмножество набора результатов или отдать приоритет получению первых нескольких строк. Когда об этом известно, программа оптимизации может отдать предпочтение планам доступа, минимизирующим время ответа для получения первых нескольких строк. Кроме того, число строк, посылаемых клиенту в одном блоке (смотрите раздел “Блокирование строк” на стр. 83), связано со значением “n” в условии OPTIMIZE FOR. Таким образом, условие OPTIMIZE FOR влияет и на то, как отобранные строки извлекаются из базы данных сервером, и на то, как отобранные строки возвращаются клиенту.

Предположим, например, что вы обращаетесь к таблице сотрудников в поисках сотрудников с наивысшей основной зарплатой.

```
SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
FROM EMPLOYEE
ORDER BY SALARY DESC
```

Вы задали индекс, убывающий по столбцу SALARY (зарплата). Однако поскольку сотрудники упорядочены по номерам, индекс по зарплатам, скорее всего, кластеризован очень плохо. Программа оптимизации, пытаясь предотвратить большое число произвольных синхронных операций ввода/вывода, скорее всего выберет метод доступа с предварительной выборкой списка (смотрите раздел “Как работает предварительная выборка списка” на стр. 272), который потребует сортировки идентификаторов строк для всех отобранных строк. При этом может возникнуть задержка перед тем, как первые отобранные строки будут возвращены прикладной программе. Если добавить к оператору условие OPTIMIZE FOR:

```
SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
FROM EMPLOYEE
ORDER BY SALARY DESC
OPTIMIZE FOR 20 ROWS
```

программа оптимизации скорее всего использует индекс SALARY прямо, зная, что по всей вероятности будут извлечены только двадцать сотрудников с самыми высокими зарплатами. Независимо от того, сколько строк можно будет объединить в блок, клиенту строки будут возвращаться блоками по двадцать строк.

Использование условия OPTIMIZE FOR заставляет программу оптимизации отдавать предпочтение планам доступа, которые избегают операций с большими объемами данных и операций, которые прерывают поток строк, например, сортировок. С наибольшей вероятностью на план доступа повлияет условие OPTIMIZE FOR 1 ROW. Результатом использования этого условия могут быть следующие эффекты:

- Менее вероятны станут последовательности объединения с составными внутренними конструкциями, поскольку они требуют создания временной таблицы.
- Может измениться способ объединения. Наиболее вероятный выбор - объединение с помощью вложенных циклов, потому что при этом невысоки затраты и этот способ обычно более эффективен, если планируется получение лишь нескольких строк.
- Скорее всего, будет использован индекс, согласованный с условием ORDER BY. Это происходит потому, что для условия ORDER BY не требуется сортировка.
- Предварительная выборка списка будет использоваться с меньшей вероятностью, поскольку этот метод доступа требует сортировки.
- Последовательная предварительная выборка будет затребована DB2 с меньшей вероятностью, так как DB2 считает, что планируется получить только небольшое число строк.
- В запросе объединения таблица со столбцами из условия ORDER BY, скорее всего, будет выбрана в качестве внешней таблицы, если в этой таблице есть индекс, упорядоченный в соответствии с условием ORDER BY.

Хотя условие OPTIMIZE FOR применимо ко всем классам оптимизации (смотрите “Настройка класса оптимизации” на стр. 70), лучше всего оно работает с классами оптимизации 3 и выше. Использование метода перечисления ″жадного″ объединения (смотрите раздел “Стратегии поиска для выбора оптимального объединения” на стр. 188) в классах оптимизации ниже 3 иногда приводит к планам доступа для многотабличных объединений, которые не позволяют быстро извлекать из себя первые несколько строк.

Даже если указано условие OPTIMIZE FOR, можно получить все отобранные строки. Однако общее время получения всех отобранных строк может быть значительно выше, чем если бы программе оптимизации было позволено выполнять оптимизацию в расчете на получение всего набора результатов.

Если есть пакетная прикладная программа, использующая интерфейс уровня вызовов (DB2 CLI или ODBC), можно заставить DB2 CLI автоматически присоединить условие OPTIMIZE FOR к концу каждого оператора запроса с помощью ключевого слова OPTIMIZEFORNROWS в файле конфигурации `db2cli.ini`. Дополнительную информацию смотрите в руководстве *CLI Guide and Reference*.

При выборе данных из псевдонимов результаты могут существенно зависеть от поддержки источника данных. Если источник данных, заданный псевдонимом, поддерживает условие OPTIMIZE FOR, и программа оптимизации DB2 переводит весь запрос, содержащий условие, на уровень источника, это условие включается в операторы SQL, посылаемые источнику данных. Если источник данных не поддерживает это условие, или если программа оптимизации решит

выполнить условие локально (план наименьшей стоимости), условие OPTIMIZE FOR применяется в DB2 локально. В этом случае программа оптимизации DB2 продолжит отдавать предпочтение планам доступа, которые минимизирует время ответа при извлечении первых нескольких строк запроса, но возможности, доступные программе оптимизации для генерации планов, будут несколько ограничены, и выигрыш в производительности от условия OPTIMIZE FOR может оказаться незначительным.

Если заданы условия и FETCH FIRST, и OPTIMIZE FOR, на размер буфера связи повлияет меньшее из двух значений. Эти два значения для целей оптимизации считаются независимыми друг от друга. Дополнительную информацию о взаимодействии этих двух условий смотрите в разделе “Использование оператора SELECT” на стр. 85.

Условие FETCH FIRST n ROWS ONLY

Даже если указано условие OPTIMIZE FOR *n* ROWS, можно получить все отобранные строки. (Общее время получения всех отобранных строк может быть значительно выше, чем если бы программе оптимизации было позволено выполнять оптимизацию в расчете на получение всего набора результатов.)

Условие FETCH FIRST *n* ROWS ONLY задает максимальное число строк, которые можно извлечь в рамках оператора SELECT. Ограничение таблицы результатов первыми несколькими строками может повысить производительность. Извлекается только n строк, независимо от числа строк, которые могли бы попасть в таблицу результатов SELECT без этого условия.

Если заданы условия и FETCH FIRST, и OPTIMIZE FOR, на размер буфера связи повлияет меньшее из двух значений. Эти два значения для целей оптимизации считаются независимыми друг от друга. Дополнительную информацию о взаимодействии этих двух условий смотрите в разделе “Использование оператора SELECT” на стр. 85.

Оператор DECLARE CURSOR WITH HOLD

Когда указатель объявляется оператором DECLARE CURSOR с условием WITH HOLD, все открытые указатели остаются открытыми по завершении транзакции. Позднее снимаются все блокировки, кроме блокировок, защищающих текущую позицию указателя для указателей, открытых с условием WITH HOLD.

Когда указатель объявляется оператором DECLARE CURSOR с условием WITH HOLD, все открытые указатели закрываются, когда транзакция завершается операцией отката (ROLLBACK). Позднее все блокировки снимаются, и локаторы больших объектов освобождаются.

Это и другое первичное условие для оператора CLOSE CURSOR сопоставляются в разделе “CLOSE CURSOR WITH RELEASE” на стр. 69.

Прикладные программы CLI могут достигать тех же результатов при помощи атрибута соединения DB2 CLI SQL_ATTR_CURSOR_HOLD. Дополнительную информацию смотрите в разделе “SQLSetStmtAttr - Set Options Related to a Statement” руководства *CLI Guide and Reference*.

Если есть пакетная прикладная программа, использующая интерфейс уровня вызовов (DB2 CLI или ODBC), можно заставить DB2 CLI автоматически добавить условие WITH HOLD для каждого объявляемого указателя с помощью ключевого слова CURSORHOLD в файле конфигурации `db2cli.ini`. Дополнительную информацию смотрите в разделе о ключевых словах конфигурации транзакции в руководстве *CLI Guide and Reference*.

Блокирование строк

Блокирование строк - это метод, сокращающий затраты менеджера базы данных за счет получения *блока* строк в ходе одной операции. Эти строки хранятся в кэше, и каждое требование FETCH в прикладной программе извлекает из кэша очередную строку. Когда все строки в блоке обработаны, менеджер базы данных получает следующий блок строк.

Кэш выделяется, когда прикладная программа выдает требование OPEN CURSOR, и освобождается при закрытии указателя. Размер кэша определяется параметром конфигурации, который используется при выделении памяти для блока ввода/вывода. Используемый параметр зависит от того, идет работа с локальным или же с удаленным клиентом:

- Для *локальных прикладных программ* при выделении кэша для блокирования строк используется параметр *aslheapsz*. (Информацию об этом параметре смотрите в разделе “Размер кучи слоя поддержки программ (aslheapsz)” на стр. 395.)
- Для *удаленных прикладных программ* при выделении *кэша* для блокирования строк используется параметр *rqrioblk* на рабочей станции клиента. Этот кэш размещается на клиенте базы данных. (Информацию об этом параметре смотрите в разделе “Размер блока ввода-вывода клиента (rqrioblk)” на стр. 398.)

Для *локальных* прикладных программ для оценки числа возвращаемых строк на один блок можно использовать приведенную ниже формулу, где:
- *aslheapsz* выражается в страницах памяти
- 4096 - число байтов на страницу
- *orl* - длина выходной строки в байтах:

```
Число строк на блок = aslheapsz * 4096 / orl
```

В случае *удаленных* прикладных программ для оценки числа возвращаемых строк на один блок можно использовать приведенную ниже формулу, где:
- *rqrioblk* выражается в байтах памяти

- *orl* - длина выходной строки в байтах:

```
Число строк на блок = rqrioblk / orl
```

Обратите внимание на то, что если в операторе SELECT используется условие FETCH FIRST *n* ROWS ONLY или OPTIMIZE FOR *n* ROWS, число строк на блок будет равно наименьшей из следующих величин:
- Значение, рассчитанное по приведенной выше формуле
- Значение *n* в условии FETCH FIRST
- Значение *n* в условии OPTIMIZE FOR

Используйте опцию BLOCKING в командах PREP и BIND, чтобы задать один из следующих типов блокирования строк:

UNAMBIG
: Блокирование происходит только для указателей только для чтения и указателях, не заданных как “FOR UPDATE OF”. Неоднозначные указатели рассматриваются как указатели с возможностью изменения.

ALL
: Блокирование происходит только для указателей только для чтения и указателях, не заданных как “FOR UPDATE OF”. Неоднозначные указатели рассматриваются как указатели только для чтения.

NO
: Блокирование не происходит ни для каких указателей. Неоднозначные указатели рассматриваются как указатели только для чтения.

Подробности об этих типах блокирования строк смотрите в описаниях команд PREP и BIND в руководстве *Command Reference*.

Если в командах PREP и BIND не задано опций, по умолчанию используется тип UNAMBIG. Для процессора командной строки и интерфейса уровня вызовов по умолчанию используется тип блокировки ALL.

Дополнительную информацию об указателях смотрите в справочнике *SQL Reference*.

Настройка запросов

Этот раздел содержит конкретные советы и рекомендации, которые помогут вам выполнить точную настройку операторов SQL в прикладной программе. В качестве общих правил эти рекомендации могут помочь при разработке программы для минимизации использования системных ресурсов и затрат времени, необходимого для доступа к данным в очень большой таблице. В зависимости от степени оптимизации при компиляции операторов SQL точная настройка этих операторов может и не потребоваться. Компилятор SQL может сам переписать ваши операторы SQL в более эффективном виде. Смотрите разделы “Перезапись запросов компилятором SQL” на стр. 159 и “Настройка класса оптимизации” на стр. 70.

Кроме того, важно отметить, что план доступа, выбранный оптимизатором, зависит и от других факторов, включая особенности среды и статистику системного каталога. Проведя для программ тесты измерения производительности, можно внести изменения, которые улучшат этот план доступа.

Использование оператора SELECT

Язык SQL - это язык высокого уровня с большими возможностями. Поэтому для получения одних и тех же данных можно написать разные *операторы SELECT*. Однако производительность для разных форм и различных классов оптимизации может отличаться.

Компилятор SQL (включающий в себя фазу перезаписи запросов и фазу оптимизации) выберет план доступа, чтобы сгенерировать набор результатов для вашего запроса. Поэтому, как отмечено во многих последующих указаниях, следует кодировать запрос так, чтобы получать только необходимые данные.

Рекомендации по использованию оператора SELECT

При использовании *оператора SELECT* руководствуйтесь следующими рекомендациями:

- Задавайте в списке выбора только те столбцы, которые необходимы. Хотя может показаться проще задать все столбцы при помощи звездочки (*), это может привести к ненужной обработке и возврату нежелательных столбцов.
- Ограничивайте число выбираемых строк, используя предикаты, чтобы получить в наборе ответов только те строки, которые нужны. (Дополнительную информацию о различных типах предикатов и их относительном влиянии на производительность смотрите в разделе “Терминология предикатов” на стр. 180.)
- Если число нужных значительно меньше общего числа строк, которое могут быть возвращены, задайте для *оператора SELECT* условие OPTIMIZE FOR. Это условие влияет и на выбор планов доступа, и на число строк, которые блокируются в буфере связи. (Дополнительную информацию смотрите в разделе “Блокирование строк” на стр. 83.)
- Если число возвращаемых строк мало, в дополнение к условию FETCH FIRST n ROWS ONLY не надо задавать условие OPTIMIZE FOR k ROWS. Но если n велико, и вы хотите выполнить оптимизация, чтобы получать первые k строк быстро, с возможной задержкой для последующих k строк, задайте оба эти параметра. Размеры для буферов связи задаются на основе выбора меньшего из n и k.

```
SELECT EMPNAME, SALARY FROM EMPLOYEE
   ORDER BY SALARY DESC
      FETCH FIRST 100 ROWS ONLY
   OPTIMIZE FOR 20 ROWS
```

- Задание условия FOR READ ONLY (или FOR FETCH ONLY) позволяет использовать преимущества блокировки строк в запросе, а при этом повышается производительность. Кроме того, если это условие задано, может

улучшиться параллелизм данных, поскольку монопольные блокировки никогда не будут удерживаться для строк, получаемых запросом. Это условие допускает также использование перезаписи запросов. Таким же образом, задав условие FOR READ ONLY (или FOR FETCH ONLY) вместе с условием BLOCKING ALL BIND, можно повысить производительность запросов с псевдонимами в системе объединения.

- Задание условия FOR UPDATE OF может также улучшить производительность для указателей, которые будут изменены, что позволит менеджеру баз данных изначально выбрать более подходящие уровни блокировок, избежав, таким образом, потенциально возможных тупиковых ситуаций (смотрите в разделе “Тупиковые ситуации” на стр. 61) и преобразований блокировок (смотрите в разделе “Преобразование блокировки” на стр. 58).
- Избегайте, где это возможно, преобразований числовых типов данных. При сравнении значений обычно эффективнее использовать элементы с одним типом данных. Если преобразования необходимы, это может привести к погрешностям, связанным с ограничениями точности, и затратами производительности на преобразования во время выполнения.

  Где возможно, используйте следующие типы данных:
  – Для коротких столбцов лучше использовать CHARACTER, а не VARCHAR
  – Целые вместо плавающих или десятичных
  – Тип DATETIME вместо символьного.
  – Числовой вместо символьного.
- Операторам SQL, содержащим условия или операции, например DISTINCT или ORDER BY, требуются данные, которые должны быть упорядочены для выполнения данной операции. Если нужно уменьшить вероятность операции сортировки, не задавайте эти условия, если без них можно обойтись.
- Чтобы проверить существование строк в таблице, не используйте:

  ```
  SELECT COUNT(*) FROM  TABLENAME
  ```

  и не проверяйте полученное значение на неравенство нулю (если вы не знаете заведомо, что таблица будет очень маленькой). При больших размерах таблицы подсчет всех строк снизит производительность. Вместо этого лучше попытаться выбрать одну строку. Это можно сделать, либо открыв указатель и получив одну строку, либо выполнив операцию выбора одной строки (SELECT INTO). (Не забудьте проверить ошибку с кодом SQLCODE -811, если *оператор SELECT* возвращает не одну, а несколько строк.)
- Если изменения не слишком интенсивны, а ваши таблицы великие, определите индексы по тем столбцам, которые часто используются в предикатах.
- Если в нескольких условиях предикатов появляется один и тот же столбец, попробуйте воспользоваться списком IN.
- Для больших списков IN, используемых вместе с переменными хоста, цикл по подмножеству переменных хоста может улучшить производительность.

Следующие указания относятся только к *операторам SELECT*, обращающимся к нескольким таблицам.

- При объединении таблиц используйте предикаты объединения. (Предикат объединения - это сравнение столбцов из различных таблиц в объединении.)
- Для возможности более эффективной обработки объединения определите в предикате объединения индексы по столбцам. Это относится и к операторам UPDATE и DELETE, которые обращаются к нескольким таблицам.
- По возможности избегайте использовать с предикатами объединения выражения или условия OR. В этом случае менеджер баз данных не сможет использовать несколько методов объединения, и в результате нельзя будет выбрать наиболее эффективный метод объединения.
- По возможности обеспечьте, чтобы в среде многораздельных баз данных обе таблицы объединения разбивались на разделы по столбцам объединения.

Дополнительную информацию смотрите в разделе “Концепции объединения” на стр. 182.

Дополнительную информацию о кодировании операторов SQL с объединениями и подзапросами смотрите в руководстве *Application Development Guide*.

Составные операторы SQL

Составные операторы SQL позволяют сгруппировать несколько операторов SQL в один выполняемый блок. Операторы SQL, содержащиеся в блоке (*подоператоры*) могут выполняться по отдельности; однако создавая и выполняя блок операторов, можно уменьшить расходы на служебные операции менеджера баз данных. Для удаленных клиентов составные операторы SQL уменьшают также число запросов, которые нужно передавать по сети.

Существует два типа составных SQL:

- **Элементарные**

  Прикладная программа получает ответ от менеджера баз данных, когда все подоператоры успешно завершатся или когда один из подоператоров закончится с ошибкой. Если один из подоператоров закончится с ошибкой, весь блок считается закончившимся с ошибкой, и все изменения, сделанные в рамках блока, будут отменены.

- **Неэлементарные**

  Прикладная программа получает ответ от менеджера баз данных, когда все подоператоры завершатся. Все подоператоры в рамках блока выполняются независимо от того, успешно ли завершился предыдущий подоператор. Откат для операторов группы может выполняться только в том случае, если отменяется единица работы, содержащая составной оператор SQL типа NOT ATOMIC.

- Элементарные составные SQL не поддерживаются DB2 Connect

- Составные SQL поддерживаются в хранимых процедурах (подпрограммах DARI)
- Составные SQL поддерживаются через:
  - Встроенные статические SQL (смотрите справочник *SQL Reference*)
  - Интерфейс уровня вызовов DB2 (смотрите руководство *CLI Guide and Reference*)
  - JDBC (смотрите руководство *Application Development Guide*).

Динамические составные операторы

Динамический составной оператор группирует другие операторы SQL в выполняемый блок. В динамическом составном операторе вы можете объявлять переменные SQL, объявлять условия, относящиеся к соответствующим SQLSTATE; он может содержать один или несколько процедурных операторов SQL. Если в динамическом составном операторе происходит ошибка, выполняется откат всех предшествующих операторов SQL, а оставшиеся операторы SQL в динамическом составном операторе не обрабатываются.

Динамический составной оператор можно встроить в триггер, функцию SQL или выполнить как динамический оператор SQL. Этот выполняемый оператор может быть подготовлен динамически. Для вызова этого оператора не требуется никаких привилегий, но относящийся к нему ID авторизации должен иметь необходимые привилегии для вызова встроенных в этот составной оператор операторов SQL.

Переменные вводятся в подоператорах в объявлении переменной. Условия вводятся в подоператорах на основе значений SQLSTATE объявления условия. Динамические составные операторы компилируются DB2 как простые операторы. Такой оператор может эффективно использоваться для коротких сценариев с простыми алгоритмами управления потоками, но значительным объемом потоков данных. Для более сложных конструкций со встроенным потоком управления лучше подходят процедуры SQL.

В динамическом составном операторе можно использовать некоторые операторы управления потоком. К ним относятся: оператор FOR, оператор IF, оператор ITERATE и оператор WHILE. Подробные сведения об этих операторах смотрите в справочнике *SQL Reference*.

Вопросы производительности и преобразование символов

Когда прикладная программа и база данных используются разные кодовые страницей, производится, если возможно, преобразование данных из одной кодовой страницы в другую. Для правильного отображения данных между кодовыми страницами прикладной программы и базы данных требуется некоторое преобразование данных.

Это отображение и преобразование данных сопряжено с некоторыми дополнительными расходами времени обработки для прикладной программы, выполняемой в кодовой странице, отличной от кодовой страницы базы данных. Производительность прикладной программы можно повысить, если программа и база данных будут использовать одну и ту же кодовую страницу или идентичную последовательность сортировки.

Преобразование кодовой страницы

Преобразование символов может произойти в следующих ситуациях:

- Когда клиент или прикладная программа, обращающиеся к базе данных, выполняются в кодовой странице, отличной от кодовой страницы базы данных.

  Оба преобразования данных - из кодовой страницы прикладной программы в кодовую страницу базы данных и из кодовой страницы базы данных в кодовую страницу прикладной программы - будут производиться на компьютере сервера базы данных.
- Когда клиент или прикладная программа, импортирующие (или загружающие) файл, выполняются в кодовой странице, отличной от кодовой страницы импортируемого (или загружаемого) файла.
- Когда для доступа к данным на сервере DRDA используется DB2 Connect.

Преобразование символов **не** применятся для:

- Имен файлов.
- Данных, направляемые в столбец или поступающие из столбца с атрибутом FOR BIT DATA, и данных, используемые в операциях SQL, результаты которых - данные FOR BIT или BLOB.
- Продуктов DB2 и платформ, где не поддерживается функция преобразования в или из EUC и UCS-2. При запуске прикладной программы будет получен код SQLCODE -332 (код состояния SQLSTATE 57017).

Дополнительную информацию о вопросах поддержки кодовой страницы EUC и поддержке национальных языков (NLS) смотрите в книге *Administration Guide: Planning*.

В зависимости от среды операционной системы менеджеры баз данных DB2 используют ту или иную функцию преобразования и таблицы преобразования или (при преобразовании многобайтных кодовых страниц) интерфейсы API преобразования DBCS.

Примечание: Преобразования символьных строк между многобайтными кодовыми страницами, например, DBCS с EUC, может привести как к удлинению, так и к сокращению строки.

Кодовые символы, присвоенные тем или иным символам в кодовых наборах PC DBCS, EUC и UCS-2 разных стран, могут приводить к разным результатам при сортировке одних и тех же символов. Если требуется сортировка нескольких кодовых наборов для разных стран, прочитайте книгу *Administration Guide: Planning*.

Поддержка кодовой страницы Extended UNIX Code (EUC)

Использование переменных хоста, содержащих графические данные, в прикладных программах на C или C++ требует рассмотрения таких вопросов, как особый прекомпилятор, производительность прикладной программы и вопросов разработки прикладной программы.

Если прикладные программы разработаны под использование кодовых наборов EUC, следует обратиться к книге *Administrative API Reference*.

Поддержка графических данных (то есть двухбайтных символов) в базе данных и прикладной программе клиента должна преодолевать ограничение двухбайтной ширины при работе со многими символами кодовых страниц Japanese (для японского языка) и Traditional Chinese (для традиционного китайского языка) EUC. Графические данные из этих кодовых страниц EUC хранятся и обрабатываются при помощи кодового набора UCS-2.

Хранимые процедуры

В среде прикладных программ баз данных многие ситуации часто повторяются; например, получение фиксированного набора данных, выполнение одних и тех же нескольких запросов к базе данных или возвращение фиксированного набора данных. Хранимые процедуры позволяют одним обращением к удаленной базе данных выполнить ранее запрограммированную процедуру. Один вызов заменяет несколько обращений к базе данных.

Обработка отдельного оператора SQL для удаленной базы данных требует двух операций передачи: отправки запроса и приема ответа. Однако прикладная программа может содержать много операторов SQL. Без хранимых процедур потребуется много операций передачи, прежде чем прикладная программа завершит свою работу.

Когда клиент баз данных использует хранимую процедуру, она расходует только две операции передачи на целый процесс, уменьшая тем самым число передач по сети. Чтобы вызвать хранимую процедуру, прикладная программа должна сначала соединиться с базой данных, содержащей эту процедуру.

Обычно эти хранимые процедуры выполняются в отдельных процессах, а не агентами базы данных. Использование отдельных процессов требует связи между хранимой процедурой и процессами агентов через маршрутизатор. Для достижения максимальной производительности хранимой процедуры можно

объявить хранимую процедуру “trusted” (доверенной) или “not fenced” (неизолированной), и в результате выполнять процедуру прямо в процессе агента базы данных. Что здесь значит ″доверенная″ и ″неизолированная″?

- *Неизолированная* означает, что ничто не отделяет хранимую процедуру от структур управления базы данных, которые используются агентом базы данных.
- *Доверенная* указывает, что вы, как администратор, уверены, что хранимая процедура не повредит, случайно или умышленно, управляющие структуры базы данных. Это значит, что вы доверяете им работу в режиме, который потенциально угрожает целостности базы данных.

Оба этих термина реально означают одно и то же - другими словами, если хранимая процедура объявлена “неизолированной”, она считается “доверенной”. Учитывая риск повреждения базы данных, использовать неизолированные хранимые процедуры **СЛЕДУЕТ ТОЛЬКО** в том случае, когда требуется достичь максимального выигрыша в производительности. Кроме того, прежде чем разрешать хранимой процедуре выполняться в неизолированном режиме, нужно убедиться, что она хорошо запрограммирована и тщательно протестирована. Если при выполнении одной из таких неизолированных хранимых процедур произойдет неисправимая ошибка, менеджер баз данных определит, произошла ли ошибка в прикладной программе или в программе менеджера баз данных, и выполнит соответствующее восстановление.

Может случиться и так, что неизолированная хранимая процедура разрушит менеджер баз данных до невосстановимого состояния, что может привести к потере данных и повреждению базы данных. При выполнении доверенных хранимых процедур должна соблюдаться крайняя осторожность. Почти во всех случаях надлежащий анализ производительности прикладной программы позволит достичь желаемой производительности без использования этой опции. Например, производительность можно повысить путем использования триггеров.

Есть два способа создать неизолированную хранимую процедуру:

- Использовать команду CREATE PROCEDURE с условием NOT FENCED.
- Поместите процедуру в особый каталог, как описано в руководстве *Быстрый старт* для вашей платформы. (Этот метод не работает для хранимых процедур Java.)

Чтобы запустить хранимую процедуру, конечный пользователь, выполняющий прикладную программу, которая вызывает процедуру, должен во время выполнения иметь одну из следующих привилегий:

- Привилегию EXECUTE или CONTROL для пакета, связанного с хранимой процедурой
- Полномочия SYSADM или DBADM

Информацию о написании программ, использующих хранимые процедуры, смотрите в руководстве *Application Development Guide*.

Активация базы данных

При запуске базы данных кэшируется несколько типов данных. Например, буферы данных кэшируются в пуле буферов, а пакеты и динамические операторы SQL кэшируются в кэше пакетов.

Если имеют место частые и короткие периоды, когда ни один пользователь не соединен с базой данных, и эти периоды чередуются с периодами, когда с базой данных соединяются несколько пользователей, преимущества кэширования теряются, поскольку кэш часто разрушается. Чтобы избежать такой ситуации, рассмотрите вариант активации базы данных следующей командой:

```
DB2 ACTIVATE DATABASE база_данных
```

Эта команда активирует указанную базу данных и запускает все требуемые службы, так что база данных становится доступной для соединений и использования любой прикладной программой. Базы данных, инициализированные командой ACTIVATE DATABASE, может быть закрыты командой DEACTIVATE DATABASE или *db2stop*. Дополнительную информацию об этих командах смотрите в руководстве *Command Reference*.

Параллельная обработка прикладных программ

Параллельная среда, поддерживаемая DB2, требует компьютеров SMP (symmetric multi-processor - симметричных мультипроцессорных). В этой среде к базе данных совместно обращаются несколько процессоров. Это делает возможным параллельное выполнение сложных запросов SQL, которые можно разделить между процессорами.

Степень реализуемого параллелизма можно задать во время компиляции прикладной программы при помощи специального регистра CURRENT DEGREE или опции связывания DEGREE. ″Степень″ - это просто число одновременно выполняемых частей запроса. Между числом процессоров и выбранным значением степени параллелизма нет прямой связи. Не обязательно при запуске прикладных программ запрашивать общее число процессоров, доступных для использования на платформе; можно выбрать значение как больше, так и меньше этого числа.

Всякое увеличение степени параллелизма увеличивает расход памяти системы и ресурсов процессора.

Применяя параллелизм, учтите, что для оптимизации производительности потребуется модификация некоторых параметров конфигурации. В среде с высокой степенью параллелизма следует пересмотреть и, при необходимости,

модифицировать параметры конфигурации, управляющие объемом совместной памяти и предварительной выборкой. Список параметров, связанных с параллельными операциями и средами многораздельных баз данных, смотрите в разделе “Многораздельная база данных” на стр. 487.

Есть три параметра конфигурации, с помощью которых можно управлять внутриразделным параллелизмом. Первый параметр - *intra_parallel* конфигурации менеджера баз данных, он включает или выключает поддержку параллелизма. Второй параметр - *max_querydegree* конфигурации базы данных, он задает верхний предел степени параллелизма при обработке любого запроса в базе данных. Это значение переопределяет значения, заданные специальным регистром CURRENT DEGREE и опцией связывания DEGREE. Третий параметр - это параметр *dft_degree*. Он задает значение по умолчанию специального регистра CURRENT DEGREE и опции связывания DEGREE.

Дополнительную информацию об использовании прикладной программы и последствиях использования степени параллелизма выше единицы смотрите в руководстве *Application Development Guide*.

Если запрос выполняется при DEGREE = ANY, менеджер баз данных выбирает степень внутриразделного параллелизма с учетом ряда факторов, включая число процессоров и характеристики запроса. Под влиянием этих факторов фактически используемая при выполнении степень может быть ниже числа процессоров.

Степень параллелизма определяется программой оптимизации SQL при компиляции оператора и может уточняться перед выполнением запроса с учетом активности базы данных. Степень параллелизма может быть ниже выбранной программой оптимизации SQL, если система интенсивно используется. Это делается в связи с тем, что внутриразделный параллелизм агрессивно использует системные ресурсы для уменьшения времени обработки запроса, а это может снизить производительность других пользователей базы данных.

Степень параллелизма, выбранную программой оптимизации SQL, можно выяснить при помощи возможности объяснения SQL, которая выводит план доступа. Степень параллелизма, используемую при выполнении, можно выяснить при помощи системного монитора базы данных. Дополнительную информацию о возможности объяснения SQL и связанных с ней инструментах смотрите в разделах “Глава 7. Возможность объяснения SQL” на стр. 225 и “Приложение C. Инструменты объяснения SQL” на стр. 591. Дополнительную информацию о мониторе смотрите в руководстве *System Monitor Guide and Reference*.

Примечание: ″Степень″ параллелизма можно задать независимо от аппаратной среды. Это значит, что параллелизм можно использовать на компьютере без SMP. Например, запросы, где ограничение задают

операции ввода/вывода, на однопроцессорных машинах могут выполняться быстрее при объявлении степени параллелизма ″2″ или больше. В этом случае единственному процессору не придется ожидать завершения задач ввода или вывода до начала обработки нового запроса. Объявление степени параллелизма ″2″ или больше не влияет прямо на параллелизм ввода/вывода на однопроцессорной машине. Такие утилиты, как Load, могут управлять параллелизмом ввода/вывода независимо от этого объявления. Ключевое слово ANY также можно использовать при изменении *dft_degree*. Использование ANY означает, что степень внутриразделного параллелизма будет определять программа оптимизации.

Во многих случаях параллельное выполнение координируют *агенты базы данных*. Дополнительную информацию и список различных параметров конфигурации менеджера баз данных, которые могут влиять на агентов базы данных, смотрите в разделе “Агенты базы данных” на стр. 287.

Глава 4. Факторы среды

Кроме факторов, которые нужно учесть при разработке и кодировании прикладной программы (они описаны в разделе “Глава 3. Особенности прикладного программирования” на стр. 43), существуют факторы среды, которые могут влиять на выбор плана доступа для прикладной программы:

- Параметры конфигурации, влияющие на оптимизацию запросов
- Влияние группы узлов на оптимизацию запросов
- Влияние табличного пространства на оптимизацию запросов
- Влияние индексов на оптимизацию запросов
- Опции сервера, влияющие на запросы базы данных объединения.

Дополнительную информацию о факторах, влияющих на оптимизатор SQL, смотрите в разделе “Глава 5. Статистика системного каталога” на стр. 117.

При настройке прикладных программ и среды выполните повторное связывание прикладных программ после внесения изменений в какую-либо из указанных выше областей. Это гарантирует, что будет использоваться наилучший план доступа.

Параметры конфигурации, влияющие на оптимизацию запросов

Некоторые параметры конфигурации влияют на выбор компилятором SQL плана доступа. Многие из них используются для одноразделных баз данных, а некоторые - только для многораздельных баз данных. Для многораздельной базы данных рекомендуется на каждом разделе базы данных использовать одни и те же значения параметров.

Если при работе в среде базы данных объединения большая часть запросов использует псевдонимы, прежде чем изменять среду, рассмотрите тип используемого запроса. Например, если пул буферов не кэширует страницы из источников данных, само по себе увеличение значения параметра *buffpage* не гарантирует, что оптимизатор будет рассматривать дополнительные альтернативы при создании плана доступа для запросов, использующих псевдонимы. (Источники данных - это СУБД и данные в системе объединения.) Оптимизатор может также выбрать локальную материализацию таблиц источника данных, как метод, минимизирующий затраты или необходимый шаг для сортировки. В этом случае увеличение объема ресурсов, доступных для DB2 Universal Database, может улучшить производительность. Дополнительную информацию смотрите в разделах “Опции сервера, влияющие на запросы базы данных объединения” на стр. 111 и “Совместная память баз данных” на стр. 365.

Ниже приводится список параметров конфигурации, влияющих на выбор компилятором SQL плана доступа:

- “Размер пула буферов (buffpage)” на стр. 366.

  При выборе плана доступа оптимизатор рассматривает затраты ввода-вывода на выборку страниц с диска в пул буферов. В процессе своей работы оптимизатор оценивает число операций ввода-вывода, требуемых для выполнения запроса. Вычисление этой оценки включает предсказание использования пула буферов, поскольку для чтения строк со страницы, которая уже находится в пуле буферов, не требуются дополнительные физические операции ввода-вывода. Для оценки того, будет ли страница находиться в пуле буферов, оптимизатор использует значение столбца *npages* таблиц системного каталога BUFFERPOOLS.

  Затраты на операции ввода-вывода для чтения таблиц могут влиять:
  – На объединение двух таблиц (как описано в разделе “Выбор внешней и внутренней таблицы” на стр. 186).
  – На использование для чтения данных некластеризованного индекса (смотрите раздел “Кластеризованные индексы” на стр. 178).

  У базы данных может быть несколько пулов буферов. Это верно и для многораздельной базы данных. Новый пул буферов можно добавить для конкретных разделов или для всех разделов базы данных. Для оценки использования пулов буферов в многораздельной базе данных оптимизатор использует столбец *npages* в таблицах системного каталога BUFFERPOOLS и BUFFERPOOLSNODE.

- “Степень параллелизма по умолчанию (dft_degree)” на стр. 469.

  Параметр конфигурации *dft_degree* задает значение по умолчанию для специального регистра CURRENT DEGREE и опции связывания DEGREE. Значение 1 задает отсутствие внутриразделного параллелизма. Значение -1 означает, что степень внутриразделного параллелизма определяется оптимизатором на основе числа процессоров и типа запроса.

- “Класс оптимизации запросов по умолчанию (dft_queryopt)” на стр. 470.

  Класс оптимизации запроса можно использовать для задания оптимизатору различных уровней оптимизации при компиляции запросов SQL. Дополнительную информацию о выборе подходящего класса оптимизации запросов смотрите в разделе “Настройка класса оптимизации” на стр. 70.

- “Среднее число активных программ (avg_appls)” на стр. 420.

  Параметр *avg_appls* используется оптимизатором SQL для оценки степени доступности пула буферов для выбранного плана доступа во время выполнения. При более высоком значении этого параметра оптимизатор выбирает для запросов план доступа с меньшим использованием пула буферов. Если для этот параметр имеет значение 1, оптимизатор считает, что весь пул буферов будет доступен для этой прикладной программы.

- “Размер кучи сортировки (sortheap)” на стр. 381.

Если для хранения полученного в результате сортировки списка данных не требуется временная таблица, эта операция сортировки может быть "конвейерной". Это значит, что результаты сортировки можно прочитать с помощью одной последовательной операции доступа. Конвейерные сортировки дают лучшую производительность, чем неконвейерные, и по возможности будет использоваться именно такой тип сортировки. (Сравнение неконвейерных сортировок с конвейерными смотрите в разделе "Влияние сортировки на оптимизатор" на стр. 201.)

При выборе плана доступа оптимизатор оценивает затраты на операции сортировки и определяет, может ли сортировка быть конвейерной; для этого он:

- Оценивает объем сортируемых данных
- При помощи параметра *sortheap* определяет, достаточно ли места для сортировки в конвейерном режиме.

- "Максимальная память для списка блокировок (locklist)" на стр. 375 и "Максимальный процент списка блокировок перед расширением (maxlocks)" на стр. 407.

  Если используется уровень изоляции (смотрите раздел "Одновременность" на стр. 43) **многократное чтение (RR)**, оптимизатор SQL по значениям параметров *locklist* и *maxlocks* определяет вероятность расширения блокировок уровня строки до блокировки уровня таблицы. Если оптимизатор решит, что при обращении к таблице возникнет такое расширение блокировок, он выберет для плана доступа блокировку уровня таблицы, чтобы избежать затрат на расширение блокировок при выполнении запроса.

- "Скорость процессора (cpuspeed)" на стр. 501.

  Скорость процессора используется оптимизатором SQL для оценки затрат на выполнение конкретных операций. Оптимизатор использует оценки затрат процессорного времени вместе с оценками различных затрат на ввод-вывод, чтобы выбрать для запроса наилучший план доступа.

  Скорость процессора может существенно влиять на выбор плана доступа. Значение этого параметра конфигурации автоматически задается при установке или перенастройке базы данных. Менять значение этого параметра нужно **только** при моделировании среды производства в системе тестирования или для учета влияния изменения аппаратного обеспечения. Использование этого параметра для моделирования другой аппаратной среды позволяет узнать, какой план доступа будет выбран для такой среды.

- "Размер кучи операторов (stmtheap)" на стр. 385.

  Размер динамической памяти операторов не влияет на выбор оптимизатором пути доступа; однако он может влиять на объем оптимизации, выполняемой для сложных операторов SQL.

  Если для параметра *stmtheap* задано недостаточно большое значение, вы можете получить предупреждение SQL, указывающее на нехватку памяти для обработки оператора. Например, SQLCODE +437 (SQLSTATE 01602) может указывать, что использованный при компиляции оператора объем

оптимизации меньше, чем запрошенный при задании класса оптимизации запросов. (Дополнительную информацию смотрите в разделе “Настройка класса оптимизации” на стр. 70.)

- “Максимальная степень параллелизма запросов (max_querydegree)” на стр. 493.

  Если этот параметр имеет значение ″ANY″, используемая степень параллелизма выбирается оптимизатором. Если он имеет другое значение, степень параллелизма для этой прикладной программы определяется значением, заданным пользователем.

- “Пропускная способность связи (comm_bandwidth)” на стр. 500.

  Для определения путей доступа оптимизатор использует значение пропускной способности связи. Оптимизатор использует значение этого параметра для оценки затрат на выполнение конкретных операций между серверами разделов многораздельной базы данных.

Более подробную информацию смотрите в разделе “Настройка параметров конфигурации” на стр. 350.

Влияние группы узлов на оптимизацию запросов

В многораздельных базах данных оптимизатор узнает, когда таблицы размещены совместно, и использует эту информацию для определения наилучшего плана доступа для запроса. Если таблицы, для которых часто выполняются запросы объединения, размещены на нескольких разделах многораздельной базы данных, желательно, чтобы объединяемые строки этих таблиц находились в одном и том же разделе базы данных. Если данные таблиц, входящих в объединение, размещены совместно, во время операции объединения не нужно будет перемещать данные с одного раздела на другой. Чтобы данные таблиц располагались совместно, поместите эти таблицы в одну группу узлов.

Дополнительную информацию о совместном размещении таблиц смотрите в разделе *Administration Guide: Planning*.

Кроме этого, распределение данных по большему числу разделов в многораздельной базе данных (в зависимости от размера таблицы) уменьшает предполагаемое время (затраты) на выполнение запроса. Время выполнения запроса зависит от числа таблиц, их размера, положения данных этих таблиц и типа запроса (требуется ли объединение, как указано выше).

Влияние табличного пространства на оптимизацию запросов

Некоторые характеристики табличных пространств могут влиять на выбор компилятором SQL плана доступа:

- Характеристики контейнеров

При выполнении запроса характеристики контейнеров могут сильно влиять на затраты ввода-вывода. При выборе плана доступа оптимизатор SQL рассматривает эти затраты ввода-вывода и учитывает все различия в затратах для доступа к данным из разных табличных пространств. Для оценки затрат ввода-вывода для обращения к данным из табличного пространства оптимизатор использует два столбца таблицы системного каталога SYSCAT.TABLESPACES:

– OVERHEAD, который содержит оценку (в миллисекундах) времени до завершения чтения из контейнера в память каких-либо данных. Оно включает затраты на передачу контроллера ввода/вывода контейнера, а также время задержки для диска, в которое входит время поиска данных на диске.

  Для оценки этих дополнительных затрат можно использовать следующую формулу:

```
OVERHEAD = среднее время поиска в миллисекундах
           + (0,5 * задержка вращения)
```

  где:
  - 0,5 соответствует средней задержке на пол-оборота
  - Задержка вращения в миллисекундах определяется скоростью вращения:

```
(1 / скорость в об/мин) * 60 * 1000
```

    где:
    - По скорости вычисляется время одного оборота в минутах
    - Умножается на 60 (секунд в минуте)
    - Умножается на 1000 (миллисекунд в секунде).

  Например, пусть скорость вращения диска - 7200 оборотов в минуту. Тогда по формуле задержки вращения получим значение

```
(1 / 7200) * 60 * 1000 = 8,328 миллисекунд
```

  которое можно затем использовать для оценки значения OVERHEAD, предполагая, что среднее время позиционирования - 11 миллисекунд:

```
OVERHEAD = 11 + (0,5 * 8,328)
         = 15,164
```

  что дает оценку значения OVERHEAD около 15 миллисекунд.

– TRANSFERRATE, который содержит оценку (в миллисекундах) времени, необходимого для чтения в память одной страницы данных.

  Если каждый контейнер табличного пространства представляет собой один физический диск, для оценки (в миллисекундах на страницу) затрат на передачу данных можно использовать следующую формулу:

```
TRANSFERRATE = (1 / паспортная_скорость) * 1000 / 1024000 * размер_страницы
```

где:
- Паспортная_скорость - это паспортная характеристика скорости передачи данных для этого диска в Мбайтах в секунду
- 1 / паспортная_скорость - это время в секундах на передачу 1 Мбайта
- Умножается на 1000 (миллисекунд в секунде)
- Делится на 1024000 (байт в Мбайте)
- Умножается на размер страницы в байтах (например, на 4096 байт для 4-Кбайтных страниц)

Например, пусть паспортная скорость передачи диска - 3 Мбайта в секунду. Вычисление

```
TRANSFERRATE = (1 / 3) * 1000 / 1024000 * 4096
             = 1,333248
```

дает оценку значения TRANSFERRATE около 1,3 миллисекунд на страницу.

Если контейнеры табличного пространства представляют собой не одиночные диски, а дисковые массивы (типа RAID), для определения значения TRANSFERRATE нужно учитывать дополнительные обстоятельства. Если используются дисковый массив относительно небольшого размера, можно умножить паспортную_скорость на число дисков, предполагая, что узким местом будут диски. Однако если контейнер состоит из большого числа дисков, узким местом могут быть не сами диски, а другой компонент подсистемы ввода-вывода (например, контроллеры дисков, шины ввода-вывода или системная шина). В этом случае нельзя считать, что пропускная способность ввода-вывода будет определяться паспортной_скоростью и числом дисков. Вместо этого необходимо измерить действительную скорость ввода-вывода (в Мбайтах) во время последовательного просмотра. Например, можно выполнить последовательный просмотр `select count(*) from big_table` для таблицы объемом несколько Мбайт. Результат надо разделить на количество контейнеров в табличном пространстве, в котором находится таблица big_table. Полученный результат подставьте вместо значения паспортная_скорость в приведенную выше формулу. Например, если при просмотре таблицы в табличном пространстве с четырьмя контейнерами получена скорость ввода-вывода 100 Мбайт в секунду (то есть 25 Мбайт на контейнер), значение TRANSFERRATE будет равно (1/25) * 1000 / 1024000 * 4096 = 0,16 миллисекунд на страницу.

Каждый из контейнеров, назначенных табличному пространству, может располагаться на отдельном физическом диске. Для получения лучших результатов все используемые для конкретного табличного пространства физические диски должны иметь одни и те же характеристики OVERHEAD и TRANSFERRATE. Если диски имеют разные характеристики, задайте для OVERHEAD и TRANSFERRATE средние значения.

Значения для этих столбцов для конкретного носителя можно получить, используя спецификации на используемое аппаратное обеспечение, или определить экспериментально. Значения можно указать в операторах CREATE TABLESPACE и ALTER TABLESPACE.

Экспериментальное определение этих значений особенно важно в упомянутых выше средах (в которых контейнеры представляют собой дисковые массивы). Следует создать простой запрос, перемещающий данные, и выполнить его вместе с подходящей для вашей платформы программой определения скорости. Затем можно повторно выполнить этот запрос с разными конфигурациями контейнеров в этом дисковом пространстве. Характеристики передачи данных в вашей среде можно изменять с помощью операторов CREATE и ALTER TABLESPACE.

Сведения о затратах на ввод/вывод, содержащиеся в этих двух значениях, влияют на оптимизатор, в частности, на то, используется ли индекс для доступа к данным и какая таблица выбирается на роль внешней и внутренней при объединении.

- Предварительная выборка

  Рассматривая затраты ввода-вывода на доступ к данным в табличном пространстве, оптимизатор будет также учитывать возможное влияние на производительность выполнения запроса предварительной выборки с диска страниц данных и индексов. Предварительная выборка страниц данных и индексов может уменьшить затраты времени и время ожидания, связанные с чтением данных в пул буферов. Дополнительную информацию смотрите в разделе “Предварительная выборка данных в пул буферов” на стр. 269.

  Для оценки объема предварительной выборки для табличного пространства оптимизатор использует значения столбцов PREFETCHSIZE и EXTENTSIZE в SYSCAT.TABLESPACES.

  – Значение EXTENTSIZE можно задать только при создании табличного пространства (например с помощью оператора CREATE TABLESPACE). Размер экстента по умолчанию - 32 страницы (каждая размером 4 Кбайта); обычно этого достаточно.

  – Значение PREFETCHSIZE можно задать при создании табличного пространства, а также с помощью оператора ALTER TABLESPACE. Размер предварительной выборки по умолчанию определяется параметром конфигурации базы данных DFT_PREFETCH_SZ, значение которого зависит от конкретной операционной системы. Ознакомьтесь с рекомендациями в разделе “Размер предварительной выборки по умолчанию (dft_prefetch_sz)” на стр. 415 по заданию значения этого параметра и внесите необходимые изменения для повышения скорости перемещения данных.

Ниже показан пример синтаксиса, используемого для изменения характеристик табличного пространства RESOURCE:

```
ALTER TABLESPACE RESOURCE
  PREFETCHSIZE 64
  OVERHEAD     19.3
  TRANSFERRATE 0.9
```

После изменения табличных пространств заново выполните связывание прикладных программ и используйте утилиту RUNSTATS для сбора самой свежей статистики об индексах, чтобы обеспечить выбор наилучших планов доступа.

Влияние индексов на оптимизацию запросов

Важно помнить, что не вы принимаете решение о том, будет ли использоваться индекс - это решение принимается оптимизатором на основе доступной информации о таблицах и индексах. Однако вы играете важную роль в этом процессе, создавая необходимые индексы, которые могут улучшить производительность. Важно также собирать статистику для этих индексов (с помощью утилиты RUNSTATS): после создания индексов или после изменения размера предварительной выборки, а также регулярно, чтобы статистика не была устаревшей. Поэтому надо знать, какие индексы можно создать и как это сделать.

Преимущества и недостатки индексирования

Для каждой заданной в запросе таблицы, для которой нет индексов, должен быть выполнено просмотр. Чем больше размер таблицы, тем больше времени занимает ее просмотр. При *просмотре таблицы* менеджер баз данных последовательно обращается к каждой строке таблицы. В отличие от этого, при *просмотре индекса* менеджер баз данных обращается к данным, используя индекс. (Смотрите раздел “Концепции просмотра индекса” на стр. 170.)

Если в операторе SELECT упоминаются столбцы индекса и оптимизатор считает, что просмотр индекса будет быстрее просмотра таблицы, он выбирает использование индекса. Файлы индексов обычно имеют меньший размер и на их чтение нужно меньше времени, чем на чтение всей таблицы (это тем более верно, чем больше размер таблицы). Кроме того, просмотр всего индекса может и не потребоваться. Предикаты применяются к данным индекса, что уменьшает число строк, которые нужно прочитать со страниц данных.

Каждая запись индекса состоит из значения ключа поиска и указателя на строку, содержащую это значение. Поиск значений может выполняться в обратном направлении, только если в операторе CREATE INDEX задан параметр ALLOW REVERSE SCANS. Таким образом, задав правильный предикат, можно ограничить объем поиска. Индекс может также использоваться для получения упорядоченной последовательности строк; в этом случае менеджеру баз данных не нужно сортировать прочитанные из таблицы строки. Если задан параметр ALLOW REVERSE SCANS, индекс можно использовать, чтобы сразу получить

строки по порядку (в прямом или обратном порядке). Более подробную информацию смотрите в руководстве *SQL Reference*.

Индекс уникальности, кроме значения ключа поиска и указателя на строку, может содержать также включенные столбцы.

Примечание: Вы не можете управлять тем, будет ли оптимизатор выбирать, а менеджер баз данных - использовать индекс. Например, если для таблицы существует индекс, это не гарантирует, что результат запроса для этой таблицы будет упорядоченным. Менеджер баз данных может при помощи оптимизатора выбрать использование этого индекса при обработке запроса, но не обязательно будет это делать. Упорядоченность набора результатов “гарантируется” только условием ORDER BY.

Индексы могут значительно уменьшить время доступа; однако индексы могут также ухудшить производительность. Перед созданием индексов учтите влияние большого числа индексов на используемый объем дискового пространства и время обработки:

- Каждый индекс занимает определенный объем дискового пространства. Этот объем зависит от размера таблицы и размера и числа столбцов, входящих в индекс.
- Для каждой операции INSERT или DELETE для таблицы выполняются дополнительные операции изменения для каждого индекса этой таблицы. Это верно также для операций UPDATE, изменяющих ключ индекса.
- Утилита LOAD заново строит все существующие индексы или добавляет к ним данные индексов.
- Чтобы переопределить значение PCTFREE, заданное при создании индекса, можно использовать параметр `indexfreespace` MODIFIED BY команды LOAD.
- Каждый индекс может увеличивать число возможных путей доступа для запроса, которые будет рассматривать оптимизатор, увеличивая тем самым время компиляции.

Будьте внимательны при выборе индексов, чтобы они соответствовали потребностям прикладной программы.

Чтобы узнать, используется ли индекс в конкретном пакете, можно использовать возможность объяснения SQL, описанную в разделе “Глава 7. Возможность объяснения SQL” на стр. 225.

Использование советчика по индексам

Советчик по индексам DB2 - это средство, помогающее выбрать оптимальный набор индексов для данных таблицы. Его можно вызвать разными способами:

- Из Центра управления, выбрав папку Индексы, щелкнув правой кнопкой мыши и выбрав **Создать —> При помощи мастера**.

- Из командной строки, введя команду `db2advis`.

Дополнительную информацию о советчике по индексам DB2 смотрите в разделе “Советчики по SQL” на стр. 245.

Использование индексных ключей большего размера

Можно разрешить задавать в качестве части ключа индекса столбцы длиной более 255 байт. Переменная реестра DB2_INDEX_2BYTEVARLEN допускает использование для хранения длины ключа индекса 2 байта вместо 1.

Изменения переменной реестра влияют на несколько операторов SQL. Это:

- CREATE TABLE. Части первичного, внешнего ключа и ключа уникальности могут иметь размер более 255 байтов.
- CREATE INDEX. Размер больше 255 байт могут иметь все индексы, включая индексы уникальности и включая столбцы с переменными частями ключей.
- ALTER TABLE. Части первичного, внешнего ключа и ключа уникальности могут иметь размер более 255 байтов. Размер больше 255 байт могут иметь все индексы, включая индексы уникальности и включая столбцы с переменными частями ключей.

  Для внешнего ключа ограничение 255 байтов снимается независимо от значения переменной реестра. Соответствие первичного ключа внешнему ключу снимает все ограничения и пределы.

Чтобы пользоваться ключами индексов больших размеров, преобразуйте существующие индексы: отбросьте индексы, задайте для переменной реестра DB2_INDEX_2BYTEVARLEN значение ON, и создайте индексы вновь (со столбцами больших размеров).

Дополнительную информацию об операторах SQL и описание их синтаксиса смотрите в справочнике *SQL Reference*.

Рекомендации по индексированию

Какие индексы нужно создать, зависит от данных и их предполагаемого использования. Следующие рекомендации помогут вам определить, какие индексы лучше всего использовать:

- Если требуются первичные ключи и ключи уникальности, определите их с помощью оператора CREATE UNIQUE INDEX. (Дополнительную информацию смотрите в руководстве *SQL Reference*.) Индексы уникальности могут помочь оптимизатору избежать выполнения определенных операций, таких как сортировки.
- Определите индексы уникальности со включенными (INCLUDE) столбцами, чтобы улучшить производительность получения данных. Хорошие кандидаты для включения в индекс уникальности в качестве включенных столбцов:
  - Столбцы, к которым часто выполняются обращения, и поэтому выгодно использовать обращения только к индексу

- Столбцы, которые не требуются для ограничения диапазона просмотра индекса
- Столбцы, не влияющие на упорядоченность или уникальность ключа индекса.

Дополнительную информацию о включенных столбцах смотрите в главе "Создание индекса или спецификации индекса" в руководстве *Administration Guide: Planning*.

- Используйте индексы для оптимизации частых запросов для таблиц достаточно большого размера (размер таблицы записан в столбце NPAGES производной таблицы каталога SYSCAT.TABLES). Советы:
  - Создайте индексы для всех столбцов, которые будут использоваться при объединении таблиц.
  - Создайте индексы для всех столбцов, в которых регулярно будет выполняться поиск конкретных значений.
- Выберите для ключей восходящий или нисходящий порядок (в зависимости от того, какой порядок будет чаще всего использоваться в запросах). Поиск значений может выполняться в обратном направлении, только если в операторе CREATE INDEX задан параметр ALLOW REVERSE SCANS. Хотя индексы можно просматривать как в прямом, так и в обратном направлении, прямой просмотр индекса (то есть просмотр в порядке, заданном при создании индекса) выполняется несколько быстрее, чем обратный. Более подробную информацию смотрите в руководстве *SQL Reference*.
- Избегайте создания индексов, ключи которых являются частью ключей другого индекса. Например, если есть индекс для столбцов a, b и c, второй индекс для столбцов a и b обычно не нужен.
- Используйте индексы для внешних ключей, чтобы улучшить производительность операций удаления и изменения для родительской таблицы.
- Используйте индексы для столбцов, которые будут часто использоваться для сортировки данных.
- Если при создании многостолбцовых индексов есть несколько вариантов выбора столбца первого ключа, задайте в качестве первого ключа столбец, который чаще используется в предикате "=" (equijoin), или столбец, содержащий наибольшее число различных значений.
- При создании индексов для всех столбцов не только расходуется большой объем дискового пространства, но уходит также много времени на подготовку индексов. Это особенно верно для сложных запросов, для которых используется класс оптимизации с динамическим программным перечислением объединения. (Смотрите раздел "Настройка класса оптимизации" на стр. 70.)
- Следующие *практические* рекомендации описывают типичное число индексов для таблицы. Это число зависит от того, как будет использоваться таблица:
  - Для сред оперативной обработки транзакций (OLTP) следует создать не более двух индексов

  - Для сред запросов (только для чтения) можно создать более пяти индексов
  - Для смешанных сред запросов/OLTP можно создать от двух до пяти индексов.
- Можно создать индекс кластеризации, способствующий размещению новых вставляемых строк в соответствии с этим индексом. Индекс кластеризации может значительно уменьшить потребность в реорганизации таблицы.

  Примечание: Если определен индекс кластеризации, таблицу следует загрузить, оставляя зарезервированное свободное пространство на каждой странице данных (пространство для вставки новых строк на эти страницы). (Для задания объема зарезервированного пространства используется ключевое слово PCTFREE оператора ALTER TABLE или условие `pagefreespace` MODIFIED BY команды LOAD.)
- При создании индексов можно использовать ключевое слово PCTFREE. PCTFREE задает объем пространства на страницах индекса, зарезервированного для будущего обновления индекса. Это позволяет уменьшить частоту разбиений страниц и улучшить производительность.
- При создании индексов можно использовать опцию MINPCTUSED. MINPCTUSED задает порог минимального количества используемого пространства на конечных страницах индекса и разрешает реорганизацию индекса без отключения. Это может уменьшить потребность в реорганизации данных и индекса в автономном режиме.

Примечание: Для объявленных временных таблиц индексы не поддерживаются.

Ниже описаны типичные обстоятельства, в которых создание индекса может улучшить производительность:

- Можно создать индекс для столбцов, используемых в условиях WHERE в наиболее часто выполняемых запросах и транзакциях.

  Для условия WHERE:

  ```
  WHERE WORKDEPT='A01' OR WORKDEPT='E21'
  ```

  обычно выгодно использовать индекс для столбца `WORKDEPT` (если только эти значения не встречаются часто в этом столбце).
- Можно создать индекс для столбца или столбцов, чтобы упорядочить строки в соответствии с последовательностью слияния. Упорядочение может запрашиваться не только условием ORDER BY, но также и другими средствами (например, условиями DISTINCT и GROUP BY).

  В следующем примере используется условие DISTINCT:

  ```
  SELECT DISTINCT WORKDEPT
       FROM EMPLOYEE
  ```

  Менеджер баз данных может использовать индекс, определенный для столбца `WORKDEPT` с восходящим или нисходящим порядком, чтобы исключить

повторяющиеся значения. Тот же индекс может использоваться для группировки значений в следующем примере с условием GROUP BY:

```
SELECT WORKDEPT, AVERAGE(SALARY)
    FROM EMPLOYEE
  GROUP BY WORKDEPT
```

- Можно создать индекс, задав для него все столбцы, используемые в операторе. Если индекс определен таким образом, для получения данных будет использоваться только обращение к индексу и не будет использоваться обращение к таблице, что улучшает производительность.

  Например, предположим, что используется следующий оператор SQL:

```
     SELECT LASTNAME
    FROM EMPLOYEE
  WHERE WORKDEPT IN ('A00','D11','D21')
```

  Если определен индекс для столбцов WORKDEPT и LASTNAME таблицы EMPLOYEE, для выполнения этого оператора вместо просмотра всей таблицы может использовать более эффективный просмотр индекса. Обратите внимание на то, что поскольку в этом предикате используется столбец WORKDEPT, именно этот столбец должен быть первым столбцом индекса.

- Еще один способ улучшения использования индексов таблицы - применение включенных (INCLUDE) столбцов. Для предыдущего примера можно определить индекс уникальности так:

```
CREATE UNIQUE INDEX x ON employee (workdept) INCLUDE (lastname)
```

  Когда столбец lastname задан в качестве включенного столбца, а не в качестве ключа индекса, значения этого столбца хранятся только на конечных страницах индекса.

Управление индексами: советы по улучшению производительности

Представленная ниже информация поможет понять, как влияет на производительность правильное использование индексов и управление ими:

1. **Создание индексов**

   При создании индексов для больших таблиц на компьютере SMP можно задать для параметра *intra_parallel* значение YES (1) или SYSTEM (-1), чтобы использовать преимущества параллельной работы.

   Для просмотра и сортировки данных можно использовать несколько процессоров. Наличие многих процессоров не дает преимуществ при создании индекса, только когда параметр конфигурации базы данных *indexsort* имеет значение NO. (По умолчанию его значение - YES). Этот параметр определяет, будет ли при создании индекса выполняться сортировка ключей индекса.

2. **Индексное табличное пространство**

   Индексы могут храниться в не в табличном пространстве, используемом для хранения данных таблицы, а в отдельном табличном пространстве. Это

может повысить производительность работы дисков, уменьшая число перемещений головок дисков. Кроме этого, индексные табличные пространства можно создать на более быстрых физических устройствах.

Табличному пространству можно также назначить отдельный пул буферов, чтобы страницы индекса не удалялись из буферов при чтении страниц данных.

Если индексы размещаются в том же индексном пространстве, что и данные, для страниц данных и страниц индексов используются одинаковые размер экстента и объем предварительной выборки. Если для индексов используется отдельное табличное пространство, для всех характеристик этого табличного пространства можно задать другие значения. Поскольку индексы обычно имеют меньший размер, чем таблицы, и распределены по меньшему числу контейнеров, для них обычно используются меньшие размеры экстентов (например, 8 и 16). Дополнительную информацию смотрите в разделе “Предварительная выборка страниц индекса” на стр. 179. Если для табличного пространства используются более быстрые устройства, оптимизатор SQL учтет это (как описано в разделе “Влияние табличного пространства на оптимизацию запросов” на стр. 98). Дополнительную информацию о табличных пространствах смотрите в руководстве *Administration Guide: Planning*.

3. **Степень кластеризации**

   Если оператор SQL требует упорядочения данных (например, при использовании условий ORDER BY, GROUP BY или DISTINCT) и существует подходящий индекс, удовлетворяющий условиям упорядочения, менеджер баз данным может в некоторых случаях *не* использовать этот индекс. Это может произойти, если:
   - У этого индекса низкая степень кластеризации (смотрите столбцы CLUSTERRATIO и CLUSTERFACTOR в SYSCAT.INDEXES)
   - Размер таблицы невелик, и поэтому проще просмотреть таблицу и отсортировать набор результатов в памяти
   - Для обращения к этой таблице можно использовать несколько индексов.

   После создания индекса кластеризации рекомендуется выполнить реорганизацию (REORG) или сортировку и загрузку (LOAD). Как правило, таблица может быть кластеризована только по одному индексу. Индекс кластеризации для таблицы определяет порядок, в котором будут построены таблица и индексы. Индекс кластеризации пытается поддерживать частичный порядок данных, улучшая значения статистик CLUSTERRATIO или CLUSTERFACTOR, собираемых утилитой RUNSTATS.

   Можно также использовать параметр PCTFREE при изменении таблицы перед ее загрузкой или реорганизацией. Для поддержки кластеризации таблицы требуется доступное пространство на каждой странице данных для вставки дополнительных строк. Когда такое пространство доступно, вставляемые дополнительные данные могут быть кластеризованы с

существующими. Поэтому можно загрузить данные в таблицу, оставив свободным некоторый процент пространства на каждой странице данных для обеспечения кластеризации дополнительных данных. Для этого можно сначала создать таблицу, а затем изменить ее, задав параметр PCTFREE. Можно также изменить таблицу перед реорганизацией данных, используя параметр PCTFREE. В противном случае (если не задан параметр PCTFREE) реорганизация не оставит дополнительного свободного пространства.

В настоящее время кластеризация при изменениях данных не поддерживается. Это значит, что если после изменения записи будет изменено ее значение ключа в индексе кластеризации, эта запись может не быть перемещена на новую страницу для поддержки порядка кластеризации. Для сохранения кластеризации используйте вместо операции UPDATE операции DELETE и затем INSERT.

4. **Утилита RUNSTATS**

   После создания нового индекса следует использовать утилиту RUNSTATS для сбора статистики индекса. Эта статистика позволяют оптимизатору определить, можно ли улучшить производительность, используя этот индекс. Дополнительную информацию по этой теме смотрите в разделе “Сбор статистики с помощью утилиты RUNSTATS” на стр. 119.

5. **Реорганизация индекса**

   Чтобы получить наибольший выигрыш в производительности от использования индексов, следует регулярно выполнять реорганизацию индексов. Обновление данных в таблице может сделать предварительную выборку страниц индексов менее эффективной. Для сохранения эффективности предварительной выборки страниц индекса требуется реорганизация индекса.

   Для реорганизации индекса можно или отбросить и создать его заново, или использовать утилиту REORG. Дополнительную информацию смотрите в разделе “Реорганизация каталогов и пользовательских таблиц” на стр. 281.

   Чтобы избежать необходимости часто выполнять реорганизацию индекса, можно задать при создании индекса параметр PCTFREE. Если при создании индекса задан параметр PCTFREE, на каждой созданной конечной странице индекса будет оставлено свободное пространство. В результате при последующих операциях, затрагивающих этот индекс, при вставке записей в индекс реже будет требоваться разбиение страниц индекса. Разбиение страниц индекса нарушает упорядоченность страниц индекса. В результате ухудшается возможность предварительной выборки страниц индекса. Задав для индекса подходящее значение PCTFREE, можно избежать реорганизаций индекса или уменьшить их частоту.

   Примечание: Заданное при создании индекса значение PCTFREE используется при повторном создании индекса при реорганизации.

При отбрасывании и повторном создании индекса для него используется новый набор страниц, которые в большей степени идут подряд и упорядочены. Это улучшает возможность предварительной выборки.

Хотя утилита REORG требует для своего выполнения большего времени, она также обеспечивает кластеризацию страниц данных. Эта кластеризация дает особенный выигрыш при просмотре индексов, обращающихся к большому числу страниц данных.

При работе в симметричной многопроцессорной (SMP) среде утилита REORG будет использовать несколько процессоров, если параметр *intra_parallel* имеет значение YES или ANY.

6. **Использование EXPLAIN**

   Регулярно выполняйте EXPLAIN для самых часто используемых запросов и проверяйте, чтобы каждый из индексов использовался хотя бы один раз. Если индекс не используется ни для одного запроса, его, возможно, следует отбросить.

   Кроме того, при помощи EXPLAIN проверяйте, не выполняются ли при объединении с вложенным циклом операции просмотра больших таблиц в качестве внутренних. Это может означать, что отсутствует индекс для столбца предиката объединения или что использование этого индекса для предиката объединения считается неэффективным. Возможно также, что отсутствует предикат объединения.

7. **Таблицы нестабильного объема**

   Таблица *нестабильного объема* - это таблица, объем содержимого которой может во время работы меняться от нулевого до очень большого. При создании плана доступа, использующего таблицу нестабильного объема, оптимизатор может выбрать использование для доступа к ней просмотра таблицы, а не просмотра индекса.

   Объявив таблицу таблицей “нестабильного объема” с помощью оператора ALTER TABLE...VOLATILE, можно разрешить оптимизатору использовать для нее просмотр индексов. Оптимизатор будет использовать просмотр индекса (а не просмотр таблицы) независимо от статистики:

   - Если в индекс включены все столбцы, или
   - Если к индексу при просмотре можно применить предикат.

   Для типизированной таблицы использование оператора ALTER TABLE...VOLATILE поддерживается только для корневой таблицы в иерархии типизированных таблиц. Дополнительную информацию смотрите в руководстве *Administration Guide: Planning* или *SQL Reference*.

Опции сервера, влияющие на запросы базы данных объединения

Система объединения состоит из DB2 DBMS (базы данных объединения) и одного или нескольких источников данных. Источники данных определяются для базы данных объединения с помощью операторов CREATE SERVER. В этих операторах также задаются опции сервера, определяющие характеристики операций системы объединения, в которых участвуют DB2 и конкретный источник данных. Впоследствии опции сервера можно изменить с помощью оператора ALTER SERVER. Дополнительную информацию об операторах CREATE SERVER и ALTER SERVER смотрите в руководстве *SQL Reference*.

Примечание: Перед созданием серверов и заданием их опций необходимо установить вариант системы для распределенного объединения и задать для параметра менеджера баз данных *federated* значение YES.

Опции сервера и их значения облегчают анализ места выполнения запроса, глобальную оптимизацию и другие аспекты работы базы данных объединения. Например, в операторе CREATE SERVER в качестве значений опции сервера можно задать некоторые статистики производительности. Скажем, для опции *cpu_ratio* можно задать значение, указывающее отношение скоростей процессоров источника данных и сервера объединения. Для опции *io_ratio* можно задать значение, указывающее отношение скоростей устройств ввода-вывода источника данных и сервера объединения. При выполнении оператора CREATE SERVER эти значения записываются в производную таблицу каталога SYSCAT.SERVEROPTIONS, затем оптимизатор использует их при определении плана доступа для этого источника данных. Если значение показателя изменилось (это может произойти, например, при изменении процессора источника данных), можно при помощи оператора ALTER SERVER внести новое значение в SYSCAT.SERVEROPTIONS. При разработке нового плана доступа для этого источника данных оптимизатор будет использовать новое значение.

Таблица 8. Опции сервера и их значения

Опция	Допустимые значения	Значение по умолчанию
collating_sequence	Задает, использует ли источник данных ту же последовательность упорядочения по умолчанию, что и база данных объединения (на основе кодового набора и информации о стране). Если на источнике данных последовательность упорядочения отличается от последовательности упорядочения DB2, большинство операций, зависящих от последовательности упорядочения DB2, нельзя выполнять удаленно на источнике данных. Пример - выполнение функций столбцов MAX для символьного столбца псевдонима на источнике данных с отличающейся последовательностью упорядочения. Поскольку при выполнении функции MAX на удаленном источнике данных могут быть получены другие результаты, DB2 будет выполнять операцию столбца и функцию MAX локально. Если в вашем запросе есть знак равенства, есть возможность выполнять часть запроса на источнике, даже если последовательности упорядочения отличаются (задано значение 'N'). Например, предикат C1 = 'A' может быть передан источнику данных. Естественно, такие запросы нельзя переносить на источник, если последовательность упорядочения на источнике данных регистронезависима. На таком источнике данных предикаты C1= 'A' и C1 = 'a' дадут одни и те же результаты, что неприемлемо в регистрозависимой среде (DB2). Администраторы могут создавать базы данных объединения с определенной последовательностью упорядочения, которая совпадает с последовательностью упорядочения источника данных. Такой подход может повысить производительность, если на всех источниках данных используется одна и та же последовательность упорядочения, или если большая часть функций столбцов применяется к источникам данных, использующих одинаковую последовательность упорядочения. 'Y' Последовательность упорядочения источника данных совпадает с последовательностью упорядочения базы данных объединения. 'N' Последовательность упорядочения источника данных не совпадает с последовательностью упорядочения базы данных объединения. 'I' Последовательность упорядочения источника данных отличается от последовательности упорядочения базы данных объединения и не зависит от регистра (например, 'TOLLESON' и 'TolLESon' считаются одинаковыми).	'N'

Таблица 8. Опции сервера и их значения (продолжение)

Опция	Допустимые значения	Значение по умолчанию
comm_rate	Задает скорость связи между сервером объединения и связанными с ним источниками данных. Выражается в мегабайтах в секунду. Допустимые значения - от нуля до 2147483648. Значения могут быть выражены только целыми числами, например, 12.	'2'
connectstring	Задает свойства инициализации, необходимые для соединения с провайдером OLE DB. Полный синтаксис и семантику строки соединения смотрите в теме ″Data Link API of the OLE DB Core Components″ в справочнике *Microsoft OLE DB 2.0 Programmer's Reference and Data Access SDK, Microsoft Press, 1998*.	Нет
cpu_ratio	Указывает, насколько быстрее или медленнее работает процессор источника данных по сравнению с процессором сервера объединения. Допустимы значения от 0 до $1x10^{23}$. Значения могут быть выражены в любом допустимом двойном формате, например 123E10, 123 или 1.21E4.	'1.0'
dbname	Имя базы данных источника данных, к которой будет обращаться сервер объединения. Требуется для источников данных семейства DB2; не используется в источниках данных Oracle**, так как экземпляры Oracle содержат только одну базу данных. Для DB2 это значение соответствует определенной базе данных в экземпляре или (для DB2 for OS/390) значению LOCATION базы данных.	Отсутствует.

Таблица 8. Опции сервера и их значения (продолжение)

Опция	Допустимые значения	Значение по умолчанию
fold_id (смотрите примечания 1 и 4 в конце этой таблицы.)	Применяется для ID пользователей, посылаемых сервером объединения на источники данных для аутентификации. Допустимые значения: 'U' Перед тем, как посылать ID пользователя на источник данных, сервер объединения переводит его в верхний регистр. Это логичный выбор для источников данных семейства DB2 и Oracle** (Смотрите в конце таблицы примечание 2.) 'N' Сервер объединения не изменяет ID пользователя перед передачей его на источник данных. (Смотрите примечание 2 в конце этой таблицы.) 'L' Перед тем, как посылать ID пользователя на источник данных, сервер объединения переводит его в нижний регистр. Если не задана ни одна из этих опций, сервер объединения пытается послать ID пользователя, переведя его в верхний регистр. В случае неудачи сервер пытается послать ID пользователя в нижнем регистре.	Отсутствует.
fold_pw (смотрите примечания 1, 3 и 4 в конце этой таблицы.)	Применяется для паролей, посылаемых сервером объединения на источники данных для аутентификации. Допустимые значения: 'U' Перед тем, как посылать пароль на источник данных, сервер объединения переводит его в верхний регистр. Это логичный выбор для источников данных семейства DB2 и Oracle**. 'N' Сервер объединения не изменяет пароль перед передачей его на источник данных. 'L' Перед тем, как посылать пароль на источник данных, сервер объединения переводит его в нижний регистр. Если не задана ни одна из этих опций, сервер объединения пытается послать пароль, переведя его в верхний регистр. В случае неудачи сервер пытается послать пароль в нижнем регистре.	Отсутствует.
io_ratio	Указывает, насколько быстрее или медленнее работает система ввода-вывода источника данных по сравнению с системой ввода-вывода сервера объединения. Допустимы значения от 0 до $1x10^{23}$. Значения могут быть выражены в любом допустимом двойном формате, например 123E10, 123 или 1.21E4.	'1.0'

Таблица 8. Опции сервера и их значения (продолжение)

Опция	Допустимые значения	Значение по умолчанию
node	Задает имя, под которым источник данных определен как экземпляр для своей реляционной СУБД. Требуется для всех источников данных. Для источника данных семейства DB2 это имя узла, заданное в каталоге `узлов DB2` базы данных объединения. Чтобы увидеть содержимое этого каталога, используйте команду **db2 list node directory**. Для источника данных Oracle** это имя сервера, заданное в файле Oracle** `tnsnames.ora`. Чтобы узнать это имя на платформе Windows NT, задайте опцию **View Configuration Information** инструмента Oracle** SQL Net Easy Configuration.	Отсутствует.
password	Задает, нужно ли посылать источнику данных пароль. 'Y' Пароли всегда посылаются источнику данных и проверяются. Это значение по умолчанию. 'N' Пароли не посылаются источнику данных (независимо от заданных отображений пользователей) и не проверяются. 'ENCRYPTION' Пароли всегда посылаются источнику данных в зашифрованном виде и проверяются. Допустимо только для источников данных семейства DB2, которые поддерживают шифрованные пароли.	'Y'
plan_hints	Задает включение *советов по плану*. Советы по плану - это фрагменты оператора, передающие дополнительную информацию оптимизаторам источников данных. Эта информация может улучшить производительность для определенных типов запросов. Советы по плану помогают оптимизатору источника данных решить, использовать индекс или нет, какой индекс использовать или какую использовать последовательность объединения таблиц. 'Y' Советы по плану нужно включать для источника данных, если он их поддерживает. 'N' Советы по плану не нужно включать для источника данных.	'N'
pushdown	'Y' DB2 рассматривает возможность передачи части операций на источник данных. 'N' DB2 будет получать от удаленного источника данных только столбцы и не будет передавать на источник данных другие операции, например, объединения.	'Y'

Таблица 8. Опции сервера и их значения (продолжение)

Опция	Допустимые значения	Значение по умолчанию
varchar_no_trailing_blanks	Задает, что источник данных использует семантику сравнения varchar без дополнения пробелами. В некоторых СУБД методы сравнения символьных строк переменной длины без хвостовых пробелов возвращают те же результаты, что и методы сравнения DB2. Если точно известно, что все столбцы типа VARCHAR таблиц или производных таблиц этого источника данных не содержат хвостовых пробелов, возможно, имеет смысл задать значение 'Y' для этой опции сервера этого источника данных. Эта опция часто используется с источниками данных Oracle**. Убедитесь, что учтены все объекты (включая производные таблицы), которые потенциально могут иметь псевдонимы. 'Y' Для такого источника данных используется семантика сравнения без дополнения пробелами, как в DB2. 'N' Для такого источника данных не используется семантика сравнения без дополнения пробелами, как в DB2.	'N'

Примечания к Табл. 8 на стр. 112:

1. Это значение применяется независимо от значения, заданного для аутентификации.
2. Поскольку DB2 хранит ID пользователей в верхнем регистре, значения 'N' и 'U' равнозначны.
3. Значение опции fold_pw не действует, если для опции password задано значение 'N'. Так как пароли не посылаются, регистр значения не имеет.
4. Лучше не задавать для любой из этих опций пустых значений. Пустое значение может показаться привлекательным, так как DB2 будет выполнять несколько попыток отправки ID пользователей и паролей, но это может снизить производительность (DB2 может посылать ID пользователя и пароль четыре раза перед тем, как будет успешно проведена аутентификация источника данных).

Глава 5. Статистика системного каталога

На решения, принимаемые компилятором SQL при оптимизации запросов SQL, сильно влияет используемая оптимизатором модель содержимого базы данных. Эта модель данных используется оптимизатором для оценки затрат на альтернативные пути доступа, которые могут использоваться для выполнения конкретного запроса.

Ключевой элемент модели данных - набор статистических показателей, собранных для данных в базе данных и хранящихся в таблицах системного каталога. В них входят статистические показатели для таблиц, псевдонимов, индексов, столбцов и пользовательских функций. При изменении данных статистики может измениться выбираемый план доступа, выбранный в качестве самого эффективного метода доступа к требуемым данным.

Примеры статистических показателей, которые используются при построении модели данных оптимизатором:
- Число страниц в таблице и число непустых страниц
- Процент строк, перемещенных с их исходных страниц на другие страницы (страницы переполнения)
- Число строк в таблице
- Число различных значений в столбце
- Степень кластеризации индекса, то есть степень, в которой физическая последовательность строк таблицы соответствует индексу.
- Число уровней индекса и число конечных страниц каждого индекса
- Число повторений часто используемых значений столбца (смотрите раздел “Сбор и использование статистики распределения” на стр. 127)
- Распределение значений столбца в диапазоне значений, присутствующих в этом столбце (смотрите раздел “Сбор и использование статистики распределения” на стр. 127)
- Оценки затрат для пользовательских функций.

Статистика для объектов обновляется в таблицах системного каталога, только когда это явно запрошено. Некоторые или все статистические показатели могут быть обновлены:
- С помощью утилиты сбора статистики RUNSTATS (смотрите раздел “Сбор статистики с помощью утилиты RUNSTATS” на стр. 119)
- С помощью утилиты LOAD, если для нее заданы опции сбора статистики
- Операторами SQL UPDATE, которые действуют на набор предопределенных производных таблиц каталога (смотрите раздел “Статистика каталога, изменяемая пользователем” на стр. 139). Обратите внимание на то, что для обновления статистики для пользовательских функций необходимо использовать именно этот метод (смотрите раздел “Изменение статистики

для пользовательских функций" на стр. 145). За исключением пользовательских функций, обновляйте каталоги вручную только для моделирования производственной среды в системе тестирования или для "анализа возможных последствий". Не следует обновлять статистику таким способом в производственной среде.

В системе базы данных объединения для сбора новой статистики для псевдонима конкретного источника данных необходимо отбросить этот псевдоним, выполнить эквивалент утилиты RUNSTATS на источнике данных и затем заново создать этот псевдоним. При создании псевдонима статистика для базовой таблицы берется из системного каталога источника данных.

Если информация определения данных в базовой таблице изменена, необходимо отбросить и заново создать псевдонимы. Например, это нужно сделать, если в определение таблицы добавлен столбец.

Кроме этого, возможно, следует пересоздать псевдоним, если упала производительность выполнения запросов. Другой способ - вручную обновить статистику в SYSSTAT.TABLES.

Будьте осторожны при создании псевдонима для производной таблицы. Статистическая информация (например, число строк для этого псевдонима) может не отражать реальных затрат на работу с этой производной таблицей. Если производная таблица определена на основе одной базовой таблицы и без применения в списке выборки (SELECT) функций столбцов, доступная оптимизатору статистическая информация может быть точной. Для сложной производной таблицы можно создать на основе псевдонимов базовых таблиц этой производной таблицы новую производную таблицу на сервере DB2 Universal Database в системе базы данных объединения, чтобы оптимизатор смог выбрать эффективный план доступа к этим данным.

Дополнительная информация:

Каталоги SYSCAT и SYSSTAT содержат информацию о собранной статистике. Смотрите в руководстве *SQL Reference* информацию о:

- Всех производных таблицах каталога и входящих в них столбцах.
- Всех обновляемых производных таблицах каталога и входящих в них столбцах. Если вас интересуют только столбцы статистики таблиц каталога, можно использовать этот раздел.
- Статистике таблиц.
- Статистике столбцов.
- Статистике распределения столбцов.
- Статистике индексов.
- Статистике пользовательских функций.

Сбор статистики с помощью утилиты RUNSTATS

Утилита RUNSTATS обновляет в таблицах системного каталога статистику, используемую при оптимизации запросов. Без этой статистики менеджер баз данных может принять решения, неблагоприятно влияющие на производительность выполнения операторов SQL. Утилита RUNSTATS позволяет собрать статистику для данных в таблицах, данных в индексах или данных в таблицах и индексах.

Используйте утилиту RUNSTATS для сбора статистики для данных таблиц и индексов, чтобы обеспечить точную информацию для процесса выбора плана доступа, в следующих ситуациях:

- Когда в таблицу загружены данные и созданы соответствующие индексы.
- Когда таблица реорганизована с помощью утилиты REORG.
- После выполнения большого числа операций изменения, удаления и вставки, влияющих на таблицу и ее индексы. (“Большое число” в данном случае может означать, что изменены 10 или 20 процентов данных таблиц и индексов.)
- Перед связыванием прикладных программ, производительность которых критична
- Если нужно сравнить новые статистические показатели с предыдущими. Регулярный сбор статистики позволяет обнаружить проблемы производительности на ранних стадиях.
- Когда изменен объем предварительной выборки.
- После использования утилиты REDISTRIBUTE NODEGROUP.

При работе с многораздельной базой данных для сбора статистики для таблицы и ее индексов выполните операцию RUNSTATS на одном узле. (Узел, на котором выполняется эта утилита, зависит от того, содержит ли данные таблицы узел, на котором запущена эта команда, или нет. Смотрите раздел “Раздел базы данных, на котором выполняется утилита RUNSTATS” на стр. 120.) Поскольку в каталоге хранится статистическая информация уровня таблицы, собранные менеджером баз данных статистические показатели уровня узла при необходимости умножаются на число узлов, содержащих эту таблицу. При этом получается приближение к действительным значениям показателей, которые можно было бы получить, выполнив утилиту RUNSTATS на каждом узле и сложив полученные значения.

Примечание: Оптимизатор запросов DB2 предполагает, что значения атрибутов (данные) равномерно распределены по всем разделам базы данных в системе. Если данные размещены не равномерно, нужно выполнить эту команду для раздела базы данных, который лучше отражает распределение данных таблицы.

Раздел базы данных, на котором выполняется утилита RUNSTATS

При запуске утилиты RUNSTATS для таблицы должно быть установлено соединение с базой данных, содержащей эту таблицу, но раздел базы данных, с которого запускается эта команда, не обязательно должен содержать раздел этой таблицы:

- Если утилита RUNSTATS запущена с раздела базы данных, содержащего раздел этой таблицы, утилита выполняется на этом разделе базы данных.
- Если утилита RUNSTATS запущена с раздела базы данных, не содержащего раздел этой таблицы, посылается запрос первому разделу базы данных в этой группе узлов, содержащему раздел этой таблицы. Затем эта утилита выполняется на том разделе базы данных.

Анализ статистики

Анализируя статистику, можно определить, когда необходима реорганизация. Обратите внимание на следующие характеристики:

- Кластеризация индексов

  Если собираются статистические показатели отношения кластеризации, их значения могут быть от 0 до 100. Если собираются статистические показатели коэффициента кластеризации, их значения будут числами между 0 и 1. В каталог SYSCAT.INDEXES будет записано значение только одного из этих двух показателей. В общем случае только один из индексов в таблице может иметь высокую степень кластеризации. Значение -1 используется для обозначения того, что показатель недоступен.

  Если надо сравнить отношения кластеризации, умножьте коэффициент кластеризации на 100, чтобы получить значение кластеризации в процентах.

  Операции просмотра индексов, при которых обращения производятся **не** только к индексу, могут выполняться лучше при более высоких отношениях кластеризации. При низком отношении кластеризации для просмотра этого типа требуется большее число операций ввода-вывода, так как уменьшается вероятность того, что после первого обращения к каждой странице данных при следующем обращении к этой странице она еще будет в пуле буферов. Увеличение размера пула буферов может улучшить производительность для некластеризованных индексов. (Информацию о том, как менеджер баз данных может улучшить производительность просмотра индексов для индексов с низким отношением кластеризации, смотрите в разделе “Как работает предварительная выборка списка” на стр. 272; информацию о том, как оптимизатор использует статистику индексов, смотрите в разделе “Кластеризованные индексы” на стр. 178.)

  Если данные таблицы изначально были кластеризованы для конкретного индекса, но теперь информация кластеризации указывает, что данные плохо кластеризованы в отношении этого индекса, можно реорганизовать таблицу, чтобы заново кластеризовать данные в отношении этого индекса.

- Число строк переполнения

Число переполнения указывает число строк, которые не умещаются на своих исходных страницах. Это может произойти, когда в столбцы VARCHAR записываются более длинные значения. В таких случаях указатель продолжает указывать исходное положение строки. Это может ухудшить производительность, поскольку менеджер баз данных использует указатель для поиска содержимого строки и это увеличивает время обработки, а также, возможно, число операций ввода-вывода.

С ростом числа строк переполнения растут также и потенциальные выгоды от реорганизации данных таблицы. Реорганизация данных таблицы устраняет переполнения строк.

- Сравнение страниц файла

  Можно сравнить число страниц, содержащих строки, с общим числом страниц в таблице. Пустые страницы будут читаться при просмотре таблицы. Пустые страницы могут возникнуть при удалении целых блоков строк.

  С ростом числа пустых страниц растет потребность в реорганизации таблицы. При реорганизации таблицы уменьшается используемый таблицей объем пространства и освобождаются пустые страницы. Кроме более эффективного использования дискового пространства, освобождение пустых страниц может также улучшить производительность просмотра таблицы, так как в пул буферов нужно будет помещать меньшее число страниц.

- Число конечных страниц

  Число конечных страниц позволяет определить число операций чтения страниц индекса, необходимое для просмотра индекса.

  Изменение данных может вызывать разбиение страниц, в результате которого размер индекса увеличивается и становится больше минимального возможного. При реорганизации таблицы индексы строятся заново и каждый из них занимает минимальный возможный объем. Дополнительную информацию о минимальном требуемом для индекса объеме смотрите в разделе “Влияние индексов на оптимизацию запросов” на стр. 102 или в разделе “Создание индекса или спецификации индекса” книги *Administration Guide: Planning*.

  Примечание: По умолчанию при повторном построении индекса на каждой странице индекса оставляется свободным десять процентов пространства. Объем оставляемого свободного пространства можно увеличить, задав при первом создании индекса параметр PCTFREE. При последующих реорганизациях индекса будет использоваться это значение PCTFREE. Оставляя более десяти процентов свободного пространства, можно уменьшить число необходимых реорганизаций индекса. Это свободное пространство используется для вставки дополнительных данных индекса.

Утилита RUNSTATS может также помочь определить, как влияют на производительность изменения в базе данных. Статистика показывает распределение данных в таблице. При регулярном применении утилиты RUNSTATS можно получить информацию о таблицах и индексах за определенный промежуток времени, что позволяет обнаружить изменение производительности в соответствии с изменением со временем модели данных.

В идеале после сбора статистики следует заново выполнить связывание прикладных программ, поскольку оптимизатор запросов может на основании новых данных статистики выбрать другой план доступа.

Если достаточного времени для одновременного сбора всех статистик нет, можно периодически выполнять RUNSTATS для обновления только части возможных показателей. Если при работе с таблицей между запусками RUNSTATS для частичного обновления статистики будут обнаружены несовместимости статистики, будет выдано предупреждение (SQL0437W, код причины 6). Например, сначала RUNSTATS используется для сбора статистики распределения таблицы. Затем RUNSTATS используется для сбора статистики индексов. Если при работе с таблицей будут обнаружены несовместимости, статистика распределения таблицы будут отброшена и будет выдано предупреждение. Если это произошло, рекомендуется выполнить RUNSTATS для сбора статистики распределения таблицы.

Следует регулярно использовать RUNSTATS для сбора и статистики таблицы, и статистики индексов, чтобы статистика индексов были синхронизированы со статистикой таблицы. Статистика индексов включают в себя большинство статистических показателей таблицы и столбцов, собранных во время последнего выполнения RUNSTATS. Если с момента последнего сбора статистики данные таблицы были сильно изменены, сбор для этой таблицы только статистики индексов приведет к тому, что эти два набора статистики останутся несинхронизированными.

Статистику только для данных индекса можно собирать в следующих ситуациях:

- После выполнения утилиты RUNSTAT был создан новый индекс и вы не хотите заново собирать статистику для данных таблицы.
- После значительных изменений данных, влияющих на первый столбец индекса.

Утилита RUNSTATS позволяет собирать различные уровни статистики. Для таблиц можно собрать только статистику базового уровня или же статистику распределения для значений столбцов таблицы (смотрите раздел “Сбор и использование статистики распределения” на стр. 127). Для индексов можно собрать только статистику базового уровня или же подробную статистику, помогающие оптимизатору получить более точную оценку затрат ввода-вывода

для операций просмотра индекса. (Информацию о “подробной” статистике смотрите в разделе “Кластеризованные индексы” на стр. 178).

Примечание: Статистика не собирается для столбцов с типами данных LONG, столбцов больших объектов или структурированных типов. Для типов строк не собираются статистика уровня таблицы NPAGES, FPAGES и OVERFLOW для подтаблицы. Статистика не собираются для расширенных индексов, а также для объявленных временных таблиц.

В следующих таблицах показаны статистические показатели каталога, обновляемые утилитой RUNSTATS:

Таблица 9. Статистические показатели таблиц (SYSCAT.TABLES и SYSSTAT.TABLES)

Статистический показатель	**Описание**	**Опция RUNSTATS**	
		Таблица	**Индексы**
FPAGES	число страниц, используемых таблицей	Да	Да
NPAGES	число страниц, содержащих строки	Да	Да
OVERFLOW	число строк переполнения	Да	Нет
CARD	число строк в таблице (объем таблицы)	Да	Да (примечание 2)
Примечание: 1. Для многораздельной базы данных значение каждого статистического показателя определяются на основе ее значения для раздела базы данных, умноженного на число разделов. 2. Если у таблицы нет индексов и запрошен сбор статистики для индексов, значение показателя CARD не будет обновлено. Сохранится прежнее значение CARD.			

Таблица 10. Статистические показатели столбцов (SYSCAT.COLUMNS и SYSSTAT.COLUMNS)

Статистический показатель	**Описание**	**Опция RUNSTATS**	
		Таблица	**Индексы**
COLCARD	мощность столбца	Да (примечание 1)	Да (примечание 2)
AVGCOLLEN	средняя длина столбца	Да	Да (примечание 2)
HIGH2KEY	второе по величине значение в столбце	Да	Да (примечание 2)
LOW2KEY	предпоследнее по величине значение в столбце	Да	Да (примечание 2)
NUMNULLS	число пустых значений (NULL) в столбце	Да	Да (примечание 2)

Таблица 10. Статистические показатели столбцов (SYSCAT.COLUMNS и SYSSTAT.COLUMNS) (продолжение)

Статистический показатель	**Описание**	**Опция RUNSTATS**	
		Таблица	**Индексы**
Примечание: 1. Значение оценки COLCARD вычисляется для всех столбцов таблицы. Если столбец - это одностолбцовый ключ разделения для таблицы в многораздельной базе данных, это значение определяются на основе значения для раздела базы данных, умноженного на число разделов. 2. Статистика столбцов собирается только для первого столбца в ключе индекса.			

Таблица 11. Статистика индексов (SYSCAT.INDEXES и SYSSTAT.INDEXES)

Статистический показатель	**Описание**	**Опция RUNSTATS**	
		Таблица	**Индексы**
NLEAF	число конечных страниц индекса	Нет	Да (примечание 3)
NLEVELS	число уровней индекса	Нет	Да
CLUSTERRATIO	степень кластеризации данных таблицы	Нет	Да (примечание 2)
CLUSTERFACTOR	более точное значение степени кластеризации	Нет	Подробная (примечания 1, 2)
DENSITY	Отношение (в процентах) значения SEQUENTIAL_PAGES к числу страниц в диапазоне страниц, занятом индексом (примечание 4)	Нет	Да
FIRSTKEYCARD	Число различных значений в первом столбце индекса	Нет	Да (примечание 3)
FIRST2KEYCARD	Число различных значений в первых двух столбцах индекса	Нет	Да (примечание 3)
FIRST3KEYCARD	Число различных значений в первых трех столбцах индекса	Нет	Да (примечание 3)
FIRST4KEYCARD	Число различных значений в первых четырех столбцах индекса	Нет	Да (примечание 3)

Таблица 11. Статистика индексов (SYSCAT.INDEXES и SYSSTAT.INDEXES) (продолжение)

Статистический показатель	Описание	Опция RUNSTATS	
		Таблица	**Индексы**
FULLKEYCARD	Число различных значений во всех столбцах индекса	Нет	Да (примечание 3)
PAGE_FETCH_PAIRS	оценки выборки страниц для различных размеров буферов	Нет	Подробная (примечания 1, 2)
SEQUENTIAL_PAGES	число конечных страниц, расположенных на диске в порядке ключей индекса, без пропусков между ними или с небольшими пропусками	Нет	Да

Примечание:

1. Подробная статистика для индексов собирается, если в команде RUNSTATS задано условие DETAILED или если в вызове API RUNSTATS параметр `statsopt` имеет значение `A`, `Y` или `X`.
2. Статистики CLUSTERFACTOR и PAGE_FETCH_PAIRS собираются при заданном условии DETAILED, только если таблица имеет достаточный размер. Для сбора этих показателей размер таблицы должен быть больше примерно 25 страниц. В это случае показатель CLUSTERRATIO имеет значение -1 (не собирается). Если размер таблица относительно невелик, RUNSTATS вычисляет CLUSTERRATIO и не вычисляет CLUSTERFACTOR и PAGE_FETCH_PAIRS. Если условие DETAILED не задано, определяется только CLUSTERRATIO.
3. Для многораздельной базы данных это значение определяются на основе значения для раздела базы данных путем умножения на число разделов.
4. Эта статистика описывает долю (в процентах) страниц, относящихся к этой таблице, в файле, содержащем индекс. Если для таблицы определен только один индекс, значение DENSITY будет равно 100. Значение DENSITY используется оптимизатором для оценки среднего числа ненужных страниц (страниц других индексов), которые могут быть прочитаны при предварительной выборке страниц индекса.

Таблица 12. Статистика распределения столбцов (SYSCAT.COLDIST и SYSSTAT.COLDIST)

Статистический показатель	Описание	Опция RUNSTATS	
		Таблица	**Индексы**
DISTCOUNT	Если TYPE = Q, это число разных значений, меньших или равных значению COLVALUE	Распределения (примечание 2)	Нет
TYPE	Указывает, какой показатель содержит строка: показатель частых значений или квантиль	Распределения	Нет

Таблица 12. Статистика распределения столбцов (SYSCAT.COLDIST и SYSSTAT.COLDIST) (продолжение)

Статистический показатель	**Описание**	**Опция RUNSTATS**	
		Таблица	**Индексы**
SEQNO	Номер ранжирования частоты для последовательного номера, помогающий однозначно идентифицировать строку в этой таблице	Распределения	Нет
COLVALUE	Значение данных, для которого собирается статистика частот или квантилей	Распределения	Нет
VALCOUNT	Частота, с которой в столбце встречается значение данных (COLVALUE), или, для квантилей, число значений, меньших или равных этому значению данных.	Распределения	Нет

Примечание:

1. Статистика распределения столбцов собирается, если в команде RUNSTATS задано условие WITH DISTRIBUTION или если в вызове API RUNSTATS параметр `statsopt` имеет значение `A`, `D` или `Y`. Учтите, что статистика распределения может **не** собираться, если значения в столбце слишком близки.
2. Показатель DISTCOUNT определяется только для первых столбцов ключей индексов.
3. Для многораздельной базы данных значение VALCOUNT определяются на основе значения для раздела базы данных путем умножения на число разделов. Исключение: Если TYPE имеет значение 'F', а столбец - это одностолбцовый ключ разделения таблицы, значение VALCOUNT совпадает со значением для раздела базы данных.

Дополнительную информацию о статистике распределения столбцов смотрите в разделе “Сбор и использование статистики распределения” на стр. 127.

Утилита RUNSTATS не собирает статистик для пользовательских функций. Статистику для этих функций необходимо обновлять вручную. Смотрите разделы “Статистика каталога, изменяемая пользователем” на стр. 139 и “Изменение статистики для пользовательских функций” на стр. 145.

Сбор и использование статистики распределения

Менеджер баз данных может собирать, сохранять и использовать “статистику частых значений” и “квантили” - два типа показателей, дающих краткую оценку распределения значений данных в столбце. Используя эти показатели, оптимизатор может получить гораздо более точную оценку числа строк, значение столбца в которых удовлетворяет заданному равенству или предикату диапазона. Эти более точные оценки, в свою очередь, увеличивают вероятность того, что оптимизатор выберет оптимальный план.

Для сбора статистической информации о распределении значений данных используется условие WITH DISTRIBUTION команды RUNSTATS. Хотя для сбора утилитой RUNSTATS этой дополнительной статистики требуются дополнительные затраты, компилятор SQL может использовать полученную информацию, чтобы повысить вероятность выбора наилучшего плана доступа.

В некоторых случаях менеджер баз данных не будет собирать статистику распределения, не выдавая при этом сообщения об ошибке. Например:

- Параметры конфигурации *num_freqvalues* и *num_quantiles* имеют значение 0 (ноль), что означает, что статистику распределения собирать не нужно. Дополнительную информацию об этих параметрах смотрите в разделах:
  - “Сколько статистических показателей следует сохранять” на стр. 131
  - “Число сохраняемых частых значений (num_freqvalues)” на стр. 471
  - “Число квантилей для столбцов (num_quantiles)” на стр. 472.
- Распределение данных известно и без использования статистики распределения. Например, столбец, в котором нет повторяющихся значений (то есть все значения данных в этом столбце уникальны).
- Для этого типа данных статистика не собирается. Это столбец, в определении которого задан тип данных длинного поля или большой объект.
- Квантили не собираются, если столбец содержит только одно непустое значение.

Для первого столбца индекса вычисляются точные значения статистики распределения. Для остальных столбцов менеджер баз данных вычисляет приблизительные значения статистики распределения, используя для их оценки методы хеширования и выборочной проверки, поскольку вычисление точных значений статистики потребовало бы слишком много времени и памяти. При этом используются обычные статистические методы с соответствующей степенью точности.

Статистику распределения можно удалить, присвоив значение 0 или -1 всем столбцам COLVALUE и VALCOUNT таблицы SYSSTAT.COLDIST для столбцов, статистика распределения которых более не нужна.

В следующих разделах представлена информация, которая поможет вам понять и использовать эту статистику распределения:

- Статистика распределения.
- Когда следует использовать статистику распределения
- Сколько статистических показателей следует сохранять
- Как оптимизатор использует статистику распределения
- Моделирование производственных баз данных.
- Правила изменения статистики распределения для столбцов.

Статистика распределения

Для фиксированного числа N>=1 *N наиболее частых значений* в столбце включают в себя: значение данных, которое встречается наиболее часто (то есть для которого число повторов максимально), второе по частоте значение данных и так далее, вплоть до N-го наиболее частого значения. Соответствующие *показатели частых значений* содержат эти "N" значений данных и их частоты для этого столбца.

K-квантиль для столбца - это наименьшее значение данных V, такое что по крайней мере "K" строк содержат значение данные, меньшее или равное V. K-квантиль можно вычислить, упорядочив строки по возрастанию значений данных в этом столбце - K-квантилью будет значение данных в K-й строке.

Например, пусть столбец содержит следующие данные:

```
C1
--
 B
 E
 Y
 B
 F
 G
 E
 A
 J
 K
 E
 L
```

Этот столбец можно отсортировать, чтобы получить следующий упорядоченный набор значений:

```
C1'
--
 A
 B
 B
 E
 E
 E
 F
 G
```

```
J
K
L
Y
```

В столбце C1 есть девять различных значений. Для N = 2 показатели частых значений будут такими:

```
   SEQNO    COLVALUE     VALCOUNT
-------    ---------    --------
     1          E            3
     2          B            2
```

Если вычисляется 5 квантилей (смотрите раздел “Число квантилей для столбцов (num_quantiles)” на стр. 472), K-квантили этого столбца для K = 1, 3, 6, 9 и 12 будут следующими:

```
SEQNO    COLVALUE     VALCOUNT
-----    ---------    --------
  1          A            1
  2          B            3
  3          E            6
  4          J            9
  5          Y           12
```

В этом примере 6-квантиль равна E, так как шестая строка отсортированного столбца содержит значение данных E (а 6 строк исходного столбца содержат значения, меньшие или равные E).

Одно и то же значение квантили может встречаться несколько раз, если это частое значение. Для конкретного значения сохраняется максимум две квантили. Первая из этих двух квантилей имеет VALCOUNT, указывающий число строк, значения в которых строго меньше значения COLVALUE, а вторая - число строк, значения в которых меньше или равны значению COLVALUE.

Когда следует использовать статистику распределения

Чтобы решить, нужно ли сохранять статистику распределения для конкретной таблицы, учтите следующие два фактора:

1. Использование статического или динамического SQL.

   Статистика распределения наиболее полезны для динамического SQL и статического SQL, не использующего переменные хоста. При использовании SQL с переменными хоста оптимизатор ограничивает использование статистики распределения.

2. Неравномерное распределение данных.

   Целесообразно сохранять статистику распределения, если по крайней мере в одном столбце таблицы распределение данных существенно “неравномерно” и этот столбец часто используется в предикатах равенства или диапазона, то есть в условиях, подобных следующим:

```
WHERE C1 = KEY;
WHERE C1 IN (KEY1, KEY2, KEY3);
WHERE (C1 = KEY1) OR (C1 = KEY2) OR (C1 = KEY3);
WHERE C1 <= KEY;
WHERE C1 BETWEEN KEY1 AND KEY2;
```

Могут быть два типа неравномерности распределения данных (они могут встречаться вместе):

- Один тип неравномерности возникает, когда данные не распределены равномерно между наибольшим и наименьшим значениями, а в основном попадают в некоторый меньший интервал, как в следующем примере, где большая часть данных находится в интервале (5,10):

```
   C1
-----
  0,0
  5.1
  6,3
  7.1
  8,2
  8,4
  8,5
  9,1
 93,6
100,0
```

  При таком типе неравномерности может быть полезно сохранить квантили.

  В следующем примере показан запрос, который может помочь определить, высока ли степень неравномерности распределения значений в столбце.

```
SELECT C1, COUNT(*) AS OCCURRENCES
  FROM T1
GROUP BY C1
ORDER BY OCCURRENCES DESC;
```

- Другой тип неравномерности возникает, когда некоторые значения данных встречаются гораздо чаще, чем другие, как в столбце со следующими частотами значений данных:

```
Значение     Частота
----------   ---------
   20            5
    30          10
    40          10
    50          25
    60          25
    70          20
   80            5
```

  При таком типе неравномерности может быть полезно сохранить квантили и статистику частых значений.

Для сбора статистики распределения можно использовать условие WITH DISTRIBUTION команды RUNSTATS или при вызове API RUNSTATS задать для параметра `statsopt` значение `D`, `E` или `A`. Дополнительную информацию об этом интерфейсе прикладного программирования смотрите в руководстве *Administrative API Reference*.

Сколько статистических показателей следует сохранять

Сохранение большого числа показателей распределения столбцов может улучшить выбор оптимизатором планов доступа, но соответственно возрастут затраты на сбор этих показателей и на компиляцию запросов. Число показателей, которые можно вычислять и сохранять, может ограничиваться размером динамической области статистики (смотрите раздел “Размер кучи статистики (stat_heap_sz)” на стр. 386).

Если запрошен сбор статистики распределения, менеджер баз данных по умолчанию сохраняет для столбца 10 наиболее частых значений. Для большинства практических задач может быть достаточно сохранять от 10 до 100 частых значений. В идеале нужно сохранять столько показателей частых значений, чтобы частоты остальных значений или были примерно равны друг другу, или были малы по сравнению с частотами наиболее частых значений.

Чтобы задать число собираемых частых значений, используйте параметр конфигурации *num_freqvalues*, как описано в разделе “Число сохраняемых частых значений (num_freqvalues)” на стр. 471. Менеджер баз данных может собирать меньшее число показателей частых значений, чем задано, поскольку эти показатели собираются только для значений данных, которые встречаются несколько раз. Если нужно собирать только квантили, этому параметру можно присвоить нулевое значение.

Если запрошен сбор статистики распределения, менеджер баз данных по умолчанию сохраняет для столбца 20 квантилей. Это значение гарантирует, что максимальная ошибка оценки будет примерно 2,5% для любого простого одностороннего предиката диапазона (>, >=, < или <=) и 5% для любого предиката BETWEEN. Для определения числа квантилей можно использовать следующее правило:

- Определите максимальную ошибку (в процентах), приемлемую при оценке числа строк для какого-либо запроса диапазона - P
- Число квантилей должно быть примерно равно 100/P для предиката BETWEEN и 50/P для любого другого типа предиката диапазона (<, <=, > или >=).

Например, 25 квантилей дадут максимальную ошибку оценки 4% для предикатов BETWEEN и 2% для предикатов ″>″. В общем случае следует сохранять не менее 10 квантилей; более 50 квантилей могут понадобиться только для чрезвычайно неравномерных данных.

Чтобы задать число квантилей, используйте параметр конфигурации *num_quantiles*, как описано в разделе “Число квантилей для столбцов (num_quantiles)” на стр. 472. Если нужно собирать только статистике частых значений, этому параметру можно присвоить нулевое значение. Если для этого параметра задано значение “1”, статистика квантилей также не будут собираться, поскольку один квантиль охватывает весь диапазон значений.

Как оптимизатор использует статистику распределения

Почему нужно собирать и сохранять статистику распределения? Ответ на этот вопрос заключается в том, что для выбора наиболее выгодного плана доступа оптимизатору нужно оценить число строк, удовлетворяющих предикату равенства или диапазона. Чем точнее эта оценка, тем больше вероятность того, что оптимизатор выберет оптимальный план доступа. Например, рассмотрим такой запрос

```
SELECT C1, C2
  FROM TABLE1
  WHERE C1 = 'NEW YORK'
  AND C2 <= 10
```

и предположим, что существуют индексы для столбцов C1 и C2. Один возможный план доступа: при помощи индекса для столбца C1 получить все строки, удовлетворяющие предикату `C1 = 'NEW YORK'`, а затем проверить каждую из этих строк на выполнение предиката `C2 <= 10`. Другой возможный план доступа: при помощи индекса для столбца C2 получить все строки, удовлетворяющие предикату `C2 <= 10`, и затем проверить каждую из этих строк на выполнение предиката `C1 = 'NEW YORK'`. Обычно основные затраты на выполнение такого запроса - это затраты на получение строк, поэтому желательно выбрать план доступа, для которого потребуется получать меньше строк. Чтобы выбрать лучший план, необходимо оценить число строк, удовлетворяющих каждому из этих предикатов.

Если не запрошен сбор статистик распределения, оптимизатор использует только второе наибольшее значение данных (HIGH2KEY), второе наименьшее значение данных (LOW2KEY), число различных значений (COLCARD) и число строк (CARD) для столбца. Затем для оценки числа строк, удовлетворяющих предикату равенства или диапазона, используется предположение, что частоты значений данных в столбце равны и что значения данных равномерно распределены в интервале (LOW2KEY, HIGH2KEY). В частности, число строк, удовлетворяющих предикату равенства `C1 = KEY` оценивается как CARD/COLCARD, а число строк, удовлетворяющих предикату диапазона `C1 BETWEEN KEY1 AND KEY2` оценивается как:

```
      KEY2 - KEY1
------------------  x CARD       (1)
 HIGH2KEY - LOW2KEY
```

Эти оценки являются точными, только если реальное распределение значений данных в столбце достаточно равномерно. Если статистика распределения

недоступна, а частоты значений данных сильно отличаются друг от друга или значения данных в основном находятся в меньшем интервале, чем интервал (LOW_KEY,HIGH_KEY), эти оценки могут быть неправильными и оптимизатор может выбрать неоптимальный план доступа.

Если статистика распределения доступна, описанные выше ошибки могут быть значительно сокращены - для оценки числа строк, удовлетворяющих предикату равенства, используются статистика частых значений, а для оценки числа строк, удовлетворяющих предикату диапазона, используются статистика частых значений и квантили.

Пример влияния на предикаты равенства:

Рассмотрим первый предикат, имеющий вид `C1 = KEY`. Если `KEY` - это одно из N наиболее частых значений, оптимизатор просто использует частоту значения `KEY`, сохраненную в каталоге. Если значение `KEY` не является одним из N наиболее частых значений, оптимизатор оценивает число строк, удовлетворяющих этому предикату, предполагая, что значения, не входящие в число наиболее частых (их общее число равно COLCARD - N), имеют равномерное распределение. Это значит, что число строк оценивается как:

$$\frac{CARD - NUM\_FREQ\_ROWS}{COLCARD - N} \qquad (2)$$

где NUM_FREQ_ROWS - общее число строк, значения в которых равны одному из N наиболее частых значений.

Например, рассмотрим столбец `C` со следующими частотами значений данных:

Значение	Частота
1	2
2	3
3	40
4	4
5	1

Предположим, что статистика частых значений доступна только для самого частого значения (то есть N = 1). Для этого столбца CARD = 50 и COLCARD = 5. Предикату `C = 3` удовлетворяют ровно 40 строк. Если исходить из предположения о равномерном распределении данных, полученная оценка числа строк, удовлетворяющих этом предикату, будет равна 50/5 = 10 (ошибка -75%). При использовании статистики частых значений полученная оценка числа строк будет равна 40 (без ошибки).

Аналогично, предикату `C = 1` удовлетворяют 2 строки. Без использования статистики частых значений оценка числа строк, удовлетворяющих этом

предикату, будет равна 10 (ошибка 400%). Для вычисления ошибки оценки (в процентах) можно использовать следующую формулу:

```
оценка - действительное значение
----------------------------  X 100
   действительное значение
```

При использовании статистики частых значений (N = 1) оптимизатор оценит число строк, содержащих это значение, используя приведенную выше формулу (2), например:

```
(50 - 40)
--------- = 3
 (5 - 1)
```

и ошибка будет значительно меньше:

```
 3 - 2
------- = 50%
2
```

Число строк, удовлетворяющих предикату диапазона, можно оценить с помощью квантилей, как это показано в следующих примерах. Рассмотрим столбец C:

```
    C
-------
   0,0
   5,1
   6,3
   7,1
   8,2
   8,4
   8,5
   9,1
  93,6
 100,0
```

и предположим, что доступны K-квантили для K = 1, 4, 7 и 10:

```
 K     K-квантиль
---  ----------
 1      0,0
 4      7,1
 7      8,5
10    100,0
```

Сначала рассмотрим предикат C <= 8,5. В показанных выше данных этому предикату удовлетворяет ровно 7 строк. Если исходить из предположения о равномерном распределении данных и использовать приведенную выше формулу (1), в которой вместо KEY1 подставлено LOW2KEY, оценка числа строк, удовлетворяющих этом предикату, будет следующей:

```
 8,5 - 5,1
---------- x 10 *= 0
93,6 - 5,1
```

где *= означает “примерно равно”. Ошибка этой оценки - около -100%.

Если для оценки числа строк, удовлетворяющих тому же предикату (C <= 8,5), используются квантили, то поскольку значение 8,5 является значением одной из K-квантилей, в качестве оценки используется соответствующее значение K, а именно 7. В этом случае ошибка уменьшается до 0.

Теперь рассмотрим предикат C <= 10. Этому предикату удовлетворяют ровно 8 строк. В отличие от предыдущего примера, значение 10 не является одним из значений сохраненных K-квантилей. Если исходить из предположения о равномерном распределении данных и использовать формулу (1), оценка числа строк, удовлетворяющих этому предикату, будет равна 1 (ошибка -87,5%).

При использовании квантилей оптимизатор оценивает число строк, удовлетворяющих этом предикату, как r_1 + r_2, где r_1 - это число строк, удовлетворяющих предикату C <= 8,5, а r_2 - число строк, удовлетворяющих предикату C > 8,5 AND C <= 10. Как и в предыдущем примере, r_1 = 7. Для оценки r_2 оптимизатор использует линейную интерполяцию:

```
                  10 - 8,5
r_2 *= ---------- x (число строк, значение в которых > 8,5 и <= 100,0)
       100 - 8,5
       10 - 8,5
r_2 *= ---------- x (10 - 7)
       100 - 8,5
       1,5
r_2 *= ---- x (3)
       91,5
r_2 *= 0
```

Итоговая оценка - r_1 + r_2 *= 7, и абсолютная ошибка равна только -12,5%.

В приведенных выше примерах использование квантилей повышает точность оценки, поскольку реальные значения данных ″сгруппированы″ в диапазоне 5 - 10, а стандартные формулы оценки предполагают, что значения данных равномерно распределены между 0 и 100.

Использование квантилей также улучшает точность, когда есть значительная разница между частотами различных значений данных. Рассмотрим столбец, содержащий значения данных со следующими частотами:

```
Значение     Частота
----------   ---------
   20            5
   30            5
   40           15
```

```
 50          50
 60          15
 70           5
 80           5
```

Предположим, что доступны K-квантили для K = 5, 25, 75, 95, и 100:

```
 K     K-квантиль
----   ----------
  5       20
 25       40
 75       50
 95       70
100       80
```

Предположим также, что доступна статистика частых значений для трех наиболее частых значений.

Рассмотрим предикат `C BETWEEN 20 AND 30`. Из распределения значений данных можно увидеть, что этому предикату удовлетворяют ровно 10 строк. Если исходить из предположения о равномерном распределении данных и использовать формулу (1), оценка числа строк, удовлетворяющих этом предикату, будет следующей:

```
      30 - 20
-------  x 100 = 25
70 - 30
```

что дает ошибку 150%.

При использовании статистики частых значений и квантилей число строк, удовлетворяющих этом предикату, оценивается как `r_1 + r_2`, где `r_1` - число строк, удовлетворяющих предикату `(C = 20)`, а `r_2` - число строк, удовлетворяющих предикату `C > 20 AND C <= 30`. С помощью формулы (2) значение `r_1` оценивается как:

```
100 - 80
-------- = 5
  7 - 3
```

С помощью линейной интерполяции значение `r_2` оценивается как:

```
       30 - 20
------- x (число строк, значение в которых > 20 и <= 40)
40 - 20
30 - 20
= ------- x (25 - 5)
40 - 20
= 10,
```

что дает итоговую оценку 15, уменьшая ошибку в 3 раза.

Сбор и использование подробной статистики индексов

Для индекса можно собрать необязательные более подробные статистические показатели, помогающие оптимизатору лучше оценить затраты на обращение к таблице при помощи этого индекса. Это можно сделать двумя способами: во-первых, можно использовать условие DETAILED команды RUNSTATS, во-вторых, можно задать значение A, Y или X для параметра `statsopt` при вызове API RUNSTATS. Показатели подробной статистики PAGE_FETCH_PAIRS и CLUSTERFACTOR будут собираться, только если размер таблицы достаточно велик (около 25 страниц). В этом случае CLUSTERFACTOR будет иметь значение между 0 и 1, а CLUSTERRATIO будет иметь значение -1 (статистика не собрана). Если размер таблицы меньше 25 страниц, CLUSTERFACTOR будет иметь значение -1 (статистика не собрана), а CLUSTERRATIO будет иметь значение между 0 и 100, даже если для индекса этой таблицы задано условие DETAILED.

Подробная статистика индексов

Подробная статистика описывают число физических операций ввода-вывода, необходимых для обращения к таблицам данных страницы при полном просмотре индекса при различных размерах буферов. Когда утилита RUNSTATS просматривает страницы индекса, она моделирует различные размеры буферов и собирает оценки того, насколько часто в буфере не оказывается нужной страницы. Например, если в буфере доступна только одна страница, каждая ссылка в индексе на новую страницу будет вызывать ситуацию отсутствия нужной страницы в буфере, а в худшем случае каждая ссылка на строку может указывать на другую страницу, что приведет к максимальному числу операций ввода-вывода. Другой крайний случай: когда размер буфера достаточен, чтобы вместить всю таблицу (когда используется максимальный размер буфера), каждая из страниц таблицы (их общее число равно NPAGES) будет физически считываться только один раз. Таким образом, зависимость числа операций ввода-вывода от размера буфера - это монотонная невозрастающая функция.

RUNSTATS использует для этих оценок приближение в виде кусочно-линейной кривой, сохраняя в показателе PAGE_FETCH_PAIRS строку из 11 пар чисел. Первое значение каждой пары - это предполагаемый размер буфера, а второе - оценка числа физических операций ввода-вывода, необходимых для выборки страниц данных при полном просмотре индекса, если для него доступен весь буфер этого размера. Затем оптимизатор использует показатель PAGE_FETCH_PAIRS для оценки числа физических операций ввода-вывода выборки страниц данных при полном или частичном просмотре этого индекса.

Форма кривой, сохраненной в PAGE_FETCH_PAIRS для индекса, зависит от свойств кластеризации этого индекса.

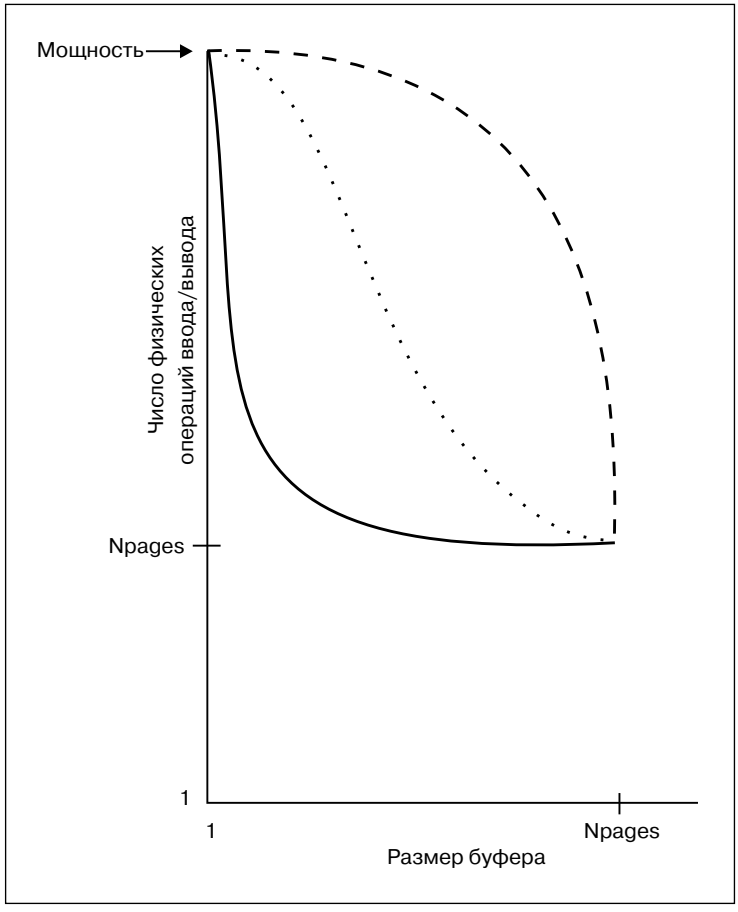


Рисунок 11. Три кривые для кластеризованных и некластеризованных индексов

Возможны три типа кривых:

1. Кривая 1 (штриховая линия) - сильно некластеризованный индекс, для которого буфер должен быть почти столь же большим, как вся таблица, чтобы при повторной ссылке на страницу ее можно было найти в буфере. В этой ситуации ссылки на одну и ту же страницу распределены по всему объему значений ключей индекса и поэтому буфер среднего размера недостаточен, чтобы избежать повторного считывания страницы при последующих ссылках на нее. Это наихудшая ситуация, так как для хорошей работы требуется наибольший объем буферного пространства. Оптимизатор, скорее всего, будет использовать для таких индексов стратегию доступа с предварительной выборкой, пытаясь сгруппировать обращения к страницам данных для нужных значений ключей индекса. Если этот индекс используется часто, он - первый кандидат на реорганизацию.

2. Кривая 2 (непрерывная линия) - индекс, некластеризованный локально. В случае очень маленьких буферов он столь же некластеризован, как индекс на кривой 1, но как только размер буфера позволяет сохранять последние считанные данные, степень попадания нужных страниц в буфер значительно возрастает. Это довольно благоприятная ситуация, в которой, несмотря на то, что индекс не является сильно кластеризованным, ссылки на одну и ту же страницу расположены среди значений ключей индекса близко друг к другу.
3. Кривая 3 (пунктирная линия) - промежуточная ситуация между этими двумя крайними - производительность равномерно растет с ростом размера буфера. Обычно это типичная ситуация для некластеризованных индексов; при отсутствии подробной статистики оптимизатор предполагает именно такую зависимость.

Когда следует использовать подробную статистику индексов

Подробную статистику индекса следует использовать, когда не все используемые в запросе столбцы входят в индекс. Кроме того, подробную статистку индекса следует использовать, когда:

- Есть несколько некластеризованных индексов с разными степенями кластеризации
- Для разных значений ключей существует разная степень кластеризации
- Значения в индексе изменяются неравномерно.

Может быть сложно обнаружить эти ситуации, не зная о них заранее и не попробовав выполнить просмотр индекса с разными размерами буферов, используя монитор для получения информации о числе физических операций ввода-вывода. Возможно, самый простой путь проверки на наличие таких ситуаций - собрать для индекса подробную статистику и сохранить ее, если полученная зависимость PAGE_FETCH_PAIRS окажется нелинейной.

Статистика каталога, изменяемая пользователем

Возможность изменять выбранные статистические показатели системного каталога позволяет:

- Смоделировать производительность выполнения запросов, используя в системе разработки статистику системы производства
- Выполнить “анализ возможных последствий” для задач производительности.

Не следует изменять статистику в системе производства, поскольку это может помешать оптимизатору найти лучший план доступа для запроса.

Чтобы изменить значения этих статистических столбцов, используйте оператор SQL UPDATE для производных таблиц, определенных в схеме SYSSTAT. Изменить статистику можно для:

- Таблиц, для которых вы обладаете явной привилегией CONTROL. Можно также изменить статистику для столбцов и индексов этих таблиц.

- Псевдонимов, для которых вы обладаете явной привилегией CONTROL в системе базы данных объединения. Можно также изменить статистику для столбцов и индексов этих псевдонимов. Учтите, что это изменение влияет только на локальные метаданные (статистика таблиц источника данных не изменяется). Эти изменения влияют только на создаваемую оптимизатором DB2 глобальную стратегию доступа.
- Пользовательских функций, которыми вы владеете (смотрите раздел “Изменение статистики для пользовательских функций” на стр. 145).

Вы можете также изменить эту статистику, если ваш ID пользователя обладает явными полномочиями DBADM для базы данных (то есть для него в таблице SYSCAT.DBAUTH заданы полномочия DBADM). Принадлежность к группе DBADM не дает явно этих полномочий.

Используя эти производные таблицы, пользователь с полномочиями DBADM может увидеть строки статистики для всех пользователей. Пользователь без полномочий DBADM может увидеть только строки, содержащие статистику для объектов, для которых он обладает привилегией CONTROL.

Далее показан пример обновления статистики таблицы для таблицы EMPLOYEE:

```
UPDATE SYSSTAT.TABLES
SET  CARD   = 10000,
     NPAGES = 1000,
     FPAGES = 1000,
     OVERFLOW = 2
WHERE TABSCHEMA = 'userid'
  AND TABNAME   = 'EMPLOYEE'
```

При изменении статистики каталога необходимо проявлять осторожность. Задание произвольных значений может сильно ухудшить производительность последующих запросов. Чтобы отменить внесенные в эти таблицы изменения, можно использовать следующие способы:

- Выполнить откат (ROLLBACK) единицы работы, в которой были сделаны эти изменения (если для этой единицы работы не было выполнено принятие).
- Используя утилиту RUNSTATS, заново вычислить и обновить статистики каталога.
- Изменить статистические показатели каталога, задав для них значения, указывающие, что они не были собраны. (Например, значение -1 показателя NPAGES столбца означает, что число страниц не было определено.)
- Задать для статистических показателей каталога значения, которые они имели перед изменением. Это можно сделать, только если перед внесением каких-либо изменений вы получили значения статистических показателей с помощью утилиты *db2look* (как это описано в разделе “Моделирование производственных баз данных” на стр. 147).

В некоторых случаях оптимизатор может определить, что некоторые конкретные значения показателей или комбинации этих значений неверны; в этих случаях он будет использовать значения по умолчанию и выдаст предупреждение. Однако такие случаи редки, так как большая часть проверок правильности при изменении статистики выполняется.

Дополнительная информация: Информацию об изменении статистики каталога смотрите в разделах:
- “Правила обновления статистики каталога”
- “Правила изменения статистики таблиц и псевдонимов” на стр. 142
- “Правила изменения статистики столбцов” на стр. 142
- “Правила изменения статистики распределения для столбцов” на стр. 143
- “Правила изменения статистики индексов” на стр. 144
- “Изменение статистики для пользовательских функций” на стр. 145
- “Моделирование производственных баз данных” на стр. 147.

Правила обновления статистики каталога

Самое важное общее правило при изменении статистики каталога: убедиться, что для различных показателей в производных таблицах статистики заданы правильные значения, диапазоны и форматы. Важно также сохранить правильные отношения между разными показателями.

Например, значение COLCARD в SYSSTAT.COLUMNS должно быть меньше значения CARD в SYSSTAT.TABLES (число различных значений в столбце не может быть больше числа строк). Предположим, вы хотите уменьшить значение COLCARD с 100 до 25, а значение CARD - с 200 до 50. Если сначала изменить SYSCAT.TABLES, возникнет ошибка (поскольку значение CARD станет меньше значения COLCARD). Правильный порядок: сначала изменить значение COLCARD в SYSCAT.COLUMNS, а затем - значение CARD в SYSSTAT.TABLES. Обратная ситуация возникает, когда вы хотите увеличить значение COLCARD с 100 до 250, а значение CARD - с 200 до 300. В этом случае нужно сначала изменить значение CARD, а затем - COLCARD.

При обнаружении конфликта между новым значением показателя и другим показателей выдается сообщение об ошибке. Однако такое сообщение не всегда выдается при возникновении конфликта. В некоторых случаях конфликт может быть трудно обнаружить, особенно если два связанных показателя находятся в разных каталогах. Поэтому нужно проявлять осторожность, чтобы не создавать такие конфликты.

Наиболее общие проверки, которые следует выполнить перед изменением статистики каталога:

1. Числовые показатели должны иметь значение -1 или значение, большее или равное нулю.
2. Числовые показатели, представляющие процентное содержание (например, CLUSTERRATIO в SYSSTAT.INDEXES), должны иметь значения от 0 до 100.

Примечание: Для типов строк нельзя изменять статистику уровня таблицы NPAGES, FPAGES и OVERFLOW для подтаблицы.

Правила изменения статистики таблиц и псевдонимов

В SYSTAT.TABLES можно обновить только четыре статистических показателя: CARD, FPAGES, NPAGES и OVERFLOW. Имейте в виду, что:

1. Значение CARD должно быть больше всех значений COLCARD в SYSSTAT.COLUMNS, относящихся к этой таблице.
2. Значение CARD должно быть больше значения NPAGES.
3. Значение FPAGES должно быть больше значения NPAGES.
4. Значение NPAGES не должно превышать ни одно из значений ″Fetch″ в столбце PAGE_FETCH_PAIRS для какого-либо индекса (если эта статистика соответствует этому индексу).
5. Значение CARD должно быть больше любого значения ″Fetch″ в столбце PAGE_FETCH_PAIRS для каждого индекса (если эта статистика соответствует этому индексу).

При работе в системе базы данных распределения будьте осторожны при ручном задании/изменении статистики для псевдонима удаленной производной таблицы. Статистическая информация (например, число строк), которая будет возвращаться для этого псевдонима, может не отражать реальных затрат на работу с этой удаленной производной таблицей и тем самым может ввести в заблуждение оптимизатор DB2. Изменение статистики может быть полезно для удаленных производных таблиц, определенных на основе одной базовой таблицы и без применения в списке выборки (SELECT) функций столбцов. Для сложных производных таблиц может потребоваться сложный процесс настройки, который может потребовать отдельной настройки для каждого запроса. Возможно, вместо этого лучше создать для таких псевдонимов локальные производные таблицы, чтобы оптимизатор DB2 мог более точно определить затраты для такой производной таблицы.

Правила изменения статистики столбцов

При изменении статистики в SYSSTAT.COLUMNS следуйте приведенным ниже правилам. Подробное описание изменения статистики распределения столбцов смотрите в разделе “Правила изменения статистики распределения для столбцов” на стр. 143.

1. Для значений HIGH2KEY и LOW2KEY (в SYSSTAT.COLUMNS) должны выполняться следующие правила:
   - Тип данных любых значений HIGH2KEY или LOW2KEY должен соответствовать типу данных пользовательского столбца, описываемого этими статистиками. Поскольку HIGH2KEY - это столбец типа VARCHAR, его значение нужно заключить в кавычки. Например, чтобы задать значение 25 для статистики HIGH2KEY для пользовательского столбца типа INTEGER, в оператор изменения нужно включить SET HIGH2KEY = '25'.

- Длина значения HIGH2KEY и LOW2KEY должна быть меньше 33 или максимальной длины типа данных в столбце назначения.
- HIGH2KEY должно быть больше, чем LOW2KEY, если в соответствующем столбце есть более 3 различных значений. Если число различных значений в столбце равно или меньше 3, HIGH2KEY может равняться LOW2KEY.

2. Мощность столбца (статистика COLCARD в SYSSTAT.COLUMNS) не может превышать мощности соответствующей ей таблицы (статистика CARD в SYSSTAT.TABLES).
3. Число пустых значений в столбце (статистика NUMNULLS в SYSSTAT.COLUMNS) не может превышать мощности соответствующей таблицы (статистика CARD в SYSSTAT.TABLES).
4. Никакие статистические показатели не поддерживаются для столбцов с типами данных: LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB.

Правила изменения статистики распределения для столбцов

Общая информация об изменении статистики каталога представлена в разделе “Статистика каталога, изменяемая пользователем” на стр. 139. Прочитайте его, прежде чем приступать к изменению статистик распределения столбцов.

Необходимо проявлять осторожность при изменении статистики распределения, чтобы не нарушить согласованности статистики в каталоге. В частности, для каждого столбца записи каталога для статистики частых значений и квантилей должны удовлетворять следующим ограничениям:

- Статистика частых значений (в каталоге SYSSTAT.COLDIST). Ограничения:
  - Значения в столбце VALCOUNT должны оставаться теми же или уменьшаться при увеличении значений SEQNO.
  - Число значений в столбце COLVALUE должно быть меньше или равно числу разных значений в столбце, которое хранится в столбце COLCARD производной таблицы каталога SYSSTAT.COLUMNS.
  - Число значений в столбце VALCOUNT должно быть меньше или равно числу значений строк в столбце, которое хранится в столбце CARD производной таблицы каталога SYSSTAT.TABLES.
  - В большинстве случаев значения в столбце COLVALUE должны лежать между вторым по величине и предпоследним по величине значениями данных для столбца, которые хранятся в столбцах HIGH2KEY и LOW2KEY, соответственно, производной таблицы каталога SYSSTAT.COLUMNS. Может быть одно частое значение, большее значения HIGH2KEY, и одно частое значение, меньшее значения LOW2KEY.
- Квантили (в каталоге SYSSTAT.COLDIST). Ограничения:
  - Значения в столбце COLVALUE должны оставаться теми же или уменьшаться при увеличении значений SEQNO
  - Значения в столбце VALCOUNT должны возрастать при увеличении значений SEQNO

– Для наибольшего значения в столбце COLVALUE соответствующее значение в столбце VALCOUNT должно равняться числу строк в столбце
– В большинстве случаев значения в столбце COLVALUE должны лежать между вторым по величине и предпоследним по величине значениями данных для столбца, которые хранятся в столбцах HIGH2KEY и LOW2KEY, соответственно, производной таблицы каталога SYSSTAT.COLUMNS.

Предположим, есть статистика распределения для столбца C1 с “R” строк и вы хотите изменить эту статистику, чтобы они соответствовали столбцу с теми же относительными долями значений данных, но содержащего “(F x R)” строк. Чтобы масштабировать значения статистики частых значений для числа строк, в F раз большего, нужно умножить на F все значения в столбце VALCOUNT. Аналогично, чтобы масштабировать значения квантилей для числа строк, в F раз большего, нужно умножить на F все значения в столбце VALCOUNT. Если не следовать этим правилам, оптимизатор может использовать неверные характеристики фильтрации, что приведет к непредсказуемой производительности при выполнении запроса.

Правила изменения статистики индексов

При изменении статистик в SYSSTAT.INDEXES следуйте описанным ниже правилам:

1. Для значений PAGE_FETCH_PAIRS (в SYSSTAT.INDEXES) должны выполняться следующие правила:
   - Отдельные значения показателя PAGE_FETCH_PAIRS должны быть разделены последовательностью пробелов.
   - Отдельные значения в показателе PAGE_FETCH_PAIRS не должны быть длиннее 10 цифр и должны быть меньше максимального целого значения (MAXINT = 2147483647).
   - Если значение CLUSTERFACTOR больше нуля, обязательно должно быть задано правильное значение PAGE_FETCH_PAIRS.
   - Одни показатель PAGE_FETCH_PAIR должна содержать ровно 11 пар.
   - Значения размеров буфера в PAGE_FETCH_PAIRS должны идти в порядке возрастания.
   - Любое значение размера буфера в записи PAGE_FETCH_PAIRS не может превышать MIN(NPAGES, 524287), где NPAGES - число страниц в соответствующей таблице (в SYSSTAT.TABLES).
   - Значения “Fetch” в PAGE_FETCH_PAIRS должны идти в порядке убывания, и ни одна из них не должна быть меньше NPAGES. Значения размера “Fetch” в записи PAGE_FETCH_PAIRS не могут быть больше значения CARD (мощности) соответствующей таблицы.
   - Если в двух идущих подряд парах задано одно значение размера буфера, в этих парах должны также совпадать и значения выборки страниц (в SYSSTAT.TABLES).

   Пример допустимого значения PAGE_FETCH_UPDATE:

```
PAGE_FETCH_PAIRS =
 '100 380 120 360 140 340 160 330 180 320 200 310 220 305 240 300
  260 300 280 300 300 300'
```

где

```
NPAGES = 300
CARD   = 10000
CLUSTERRATIO  =  -1
CLUSTERFACTOR = 0.9
```

2. Для значений CLUSTERRATIO и CLUSTERFACTOR (в SYSSTAT.INDEXES) должны выполняться следующие правила:
   - Допустимые значения для CLUSTERRATIO: -1 или от 0 до 100.
   - Допустимые значения для CLUSTERFACTOR: -1 или от 0 до 1.
   - По крайней мере одно из значений CLUSTERRATIO и CLUSTERFACTOR всегда должно быть равно -1.
   - Если значение CLUSTERFACTOR положительно, его должен сопровождать допустимый показатель PAGE_FETCH_PAIR.
3. Следующие правила применяются к значениям FIRSTKEYCARD, FIRST2KEYCARD, FIRST3KEYCARD, FIRST4KEYCARD и FULLKEYCARD:
   - Для одностолбцового индекса значение FIRSTKEYCARD должно совпадать со значением FULLKEYCARD.
   - Значение FIRSTKEYCARD должно совпадать со значением COLCARD (в SYSSTAT.COLUMNS) для соответствующего столбца.
   - Если какие-либо из этих статистических показателей индексов неприменимы, задайте для них значения -1. Например, если есть индекс только с 3 столбцами, задайте для FIRST4KEYCARD значение -1.
   - Для многостолбцовых индексов, для которых применимы все эти показатели, между ними должны быть следующие отношения:

     ```
     FIRSTKEYCARD <= FIRST2KEYCARD <= FIRST3KEYCARD <= FIRST4KEYCARD
        <= FULLKEYCARD <= CARD
     ```
4. Для значений SEQUENTIAL_PAGES и DENSITY применяются следующие правила:
   - Допустимые значения для SEQUENTIAL: -1 или от 0 до NLEAF.
   - Допустимые значения для DENSITY: -1 или от 0 до 100.

Изменение статистики для пользовательских функций

Используя производную таблицу каталога SYSSTAT.FUNCTIONS, можно изменить статистику для пользовательских функций. Если эта статистика доступна, оптимизатор будет использовать ее при оценке затрат для различных планов доступа. Если статистика не доступны (значения столбцов статистических показателей равны -1), оптимизатор будет использовать значения по умолчанию, в которых предполагается, что это простая пользовательская функция.

В следующей таблице представлена информация о столбцах статистических показателей, которые можно изменить для пользовательской функции:

Таблица 13. Статистика функций (SYSCAT.FUNCTIONS и SYSSTAT.FUNCTIONS)

Статистический показатель	**Описание**
IOS_PER_INVOC	Оценка числа запросов чтения/записи, выполняемых при каждом выполнении функции.
INSTS_PER_INVOC	Оценка числа команд компьютера, выполняемых при каждом выполнении функции.
IOS_PER_ARGBYTE	Оценка числа запросов чтения/записи на байт входного аргумента.
INSTS_PER_ARGBYTES	Оценка числа команд компьютера на байт входного аргумента.
PERCENT_ARGBYTES	Оценка средней доли (в процентах) от числа байтов входного аргумента, которые будет обрабатывать функция.
INITIAL_IOS	Оценка числа запросов чтения/записи, выполняемых при первом или последнем запуске функции.
INITIAL_INSTS	Оценка числа команд компьютера, выполняемых при первом или последнем запуске функции.
CARDINALITY	Оценка числа строк, генерируемых табличной функцией.

Например, рассмотрим пользовательскую функцию EU_SHOE, преобразующую значение размера обуви из американского стандарта в европейский. (Оба эти размера обуви должны иметь пользовательские типы.) Для этой функции следует задать значения столбцов статистических показателей так:

- INSTS_PER_INVOC следует присвоить значение оценки числа команд компьютера, необходимых для:
  - Запуска EU_SHOE
  - Инициализации выходной строки
  - Возврата результата.
- INSTS_PER_ARGBYTE следует присвоить значение оценки числа команд компьютера, необходимых для преобразования входной строки в значение размера обуви в европейском стандарте.
- PERCENT_ARGBYTES нужно присвоить значение 100, что означает, что будет преобразовываться вся входная строка.
- INITIAL_INSTS, IOS_PER_INVOC, IOS_PER_ARGBYTE и INITIAL_IOS нужно присвоить значения 0, так как эта пользовательская функция выполняет только вычисления.

PERCENT_ARGBYTES используется для функций, которые не всегда обрабатывают всю входную строку. Например, рассмотрим пользовательскую функцию LOCATE, которая получает два аргумента и возвращает начальную позицию, в которой первый аргумент найден внутри второго аргумента. Предположим, что длина первого аргумента достаточно мала, чтобы быть несущественной по сравнению с длиной второго, и что в среднем при поиске проверяется 75 процентов второго аргумента. Исходя из этого, для PERCENT_ARGBYTES нужно задать значение 75. Использованная выше оценка в 75 процентов получена на основе следующих дополнительных предположений:

- В половине случаев первый аргумент не будет найден во всем втором аргументе
- Первый аргумент может с равной вероятностью находиться в любом месте второго аргумента, поэтому когда первый аргумент будет найден во втором, для его обнаружения придется в среднем просмотреть половину второго аргумента.

Показатели INITIAL_INSTS и INITIAL_IOS можно использовать для задания оценки числа команд компьютера или запросов чтения/записи, выполняемых только при первом или последнем запуске функции. Например, с их помощью можно задать затраты на инициализацию временной памяти.

Чтобы узнать число используемых функцией операций ввода-вывода и команд компьютера, можно использовать выходную информацию компилятора языка программирования или инструменты мониторинга, доступные в конкретной операционной системы.

Моделирование производственных баз данных

Иногда желательно, чтобы система тестирования содержала подмножество данных системы производства. Однако выбираемые для такой системы тестирования планы доступа не обязательно будут такими же, как для производственной системы, если не задать для статистических показателей каталога и параметров конфигурации системы тестирования те же значения, что и для производственной системы.

Для производственной базы данных можно выполнить утилиту *db2look*, чтобы сгенерировать операторы изменения, необходимые для задания для тестовой базы данных той же статистики каталога, что и для производственной. Для генерации операторов изменения используется утилита *db2look* в режиме имитации (с опцией `-m`). В этом случае утилита *db2look* сгенерирует сценарий командного процессора, содержащий все операторы, необходимые для воспроизведения статистики каталога производственной базы данных. Это может быть полезно при анализе операторов SQL с помощью наглядного объяснения в среде тестирования.

Можно заново создать объекты базы данных (включая таблицы, производные таблицы, индексы и другие объекты базы данных), получив операторы DDL с

помощью команды *db2look -e*. Полученный таким образом сценарий командного процессора можно выполнить для другой базы данных, чтобы создать копию первой базы данных. Опцию `-e` можно использовать вместе с опцией `-m`.

Выполнив для системы тестирования созданные `db2look` операторы изменения, можно использовать систему тестирования для проверки планов доступа, которые будут сгенерированы в производственной системе. Поскольку оптимизатор использует для оценки затрат ввода-вывода информацию о типах и конфигурациях табличных пространств, система тестирования должна содержать табличные пространства с теми же характеристиками (то есть с тем же числом контейнеров того же типа: SMS или DMS).

Утилита *db2look* находится в подкаталоге *bin*.

Чтобы получить дополнительную информацию об использовании этой утилиты, введите команду:

```
db2look -h
```

Дополнительную информацию об этой утилите можно также найти в руководстве *Command Reference*.

В Центре управления есть также интерфейс для утилиты *db2look*, называемый "Генерировать SQL - Имя объекта". При использовании Центра управления предполагается, что сгенерированный утилитой файл результатов должен быть включен в Центр сценариев. С помощью Центра управления можно также внести команду *db2look* в расписание. При использовании Центра управления можно выполнить анализ только одной таблицы, в то время как при использовании команды *db2look* в одном вызове можно анализировать до тридцати таблиц. Учтите также, что Центр управления не поддерживает выходные данные в формате LaTex и в графическом формате.

Утилиту `db2look` можно также выполнить для базы данных OS/390. Утилита `db2look` получает операторы DDL и операторы изменения (UPDATE) статистики для объектов OS/390. Это очень удобно, если нужно получить информацию об объектах OS/390 и заново создать их в базе данных DB2 Universal Database (UDB). Дополнительную информацию об утилите `db2look` смотрите в руководстве *Command Reference*.

Существуют некоторые отличия между статистикой DB2 UDB и статистикой OS/390. Утилита `db2look` выполняет соответствующее преобразование статистических показателей DB2 for OS/390 в статистические показатели DB2 UDB, если это возможно, и задает значения по умолчанию (-1) для показателей DB2 UDB, для которых нет соответствующих показателей DB2 for OS/390. Ниже описано, как утилита `db2look` преобразует показатели DB2 for OS/390 в

показатели DB2 UDB. В этом описании именами “UDB_x” обозначены столбцы показателей DB2 UDB, а именами “S390_x” - столбцы показателей DB2 for OS/390.

1. Показатели уровня таблицы.

   UDB_CARD = S390_CARDF
   UDB_NPAGES = S390_NPAGES

   S390_FPAGES отсутствует. Однако в DB2 for OS/390 есть другой показатель с именем PCTPAGES, описывающий процент активных страниц табличного пространства, содержащих строки таблицы. Поэтому можно вычислить UDB_FPAGES на основе значений S390_NPAGES и S390_PCTPAGES:

   ```
   UDB_FPAGES=(S390_NPAGES * 100)/S390_PCTPAGES
   ```

   Отсутствует показатель S390_OVERFLOW, который можно было бы использовать для UDB_OVERFLOW. Поэтому утилита db2look просто задает для этого показателя значение по умолчанию:

   ```
   UDB_OVERFLOW=-1
   ```

2. Показатели уровня столбцов.

   UDB_COLCARD = S390_COLCARDF
   UDB_HIGH2KEY = S390_HIGH2KEY
   UDB_LOW2KEY = S390_LOW2KEY

   Отсутствует показатель S390_AVGCOLLEN, который можно было бы использовать для UDB_AVGCOLLEN, поэтому утилита db2look просто задает для этого показателя значение по умолчанию:

   ```
   UDB_AVGCOLLEN=-1
   ```

3. Показатели уровня индексов.

   UDB_NLEAF = S390_NLEAF
   UDB_NLEVELS = S390_NLEVELS
   UDB_FIRSTKEYCARD = S390_FIRSTKEYCARD
   UDB_FULLKEYCARD = S390_FULLKEYCARD
   UDB_CLUSTERRATIO = S390_CLUSTERRATIO

   Для других показателей, для которых нет соответствующих показателей OS/390, задается значение по умолчанию. Таким образом:

   ```
   UDB_FIRST2KEYCARD    = -1
   UDB_FIRST3KEYCARD    = -1
   UDB_FIRST4KEYCARD    = -1
   UDB_CLUSTERFACTOR    = -1
   UDB_SEQUENTIAL_PAGES = -1
   UDB_DENSITY          = -1
   ```

4. Статистика распределения столбцов.

В каталоге SYSIBM.SYSCOLUMNS в DB2 for OS/390 есть два типа показателей - типа “F” для частых значений и типа “C” для мощности. Только показатели типа “F” можно использовать для DB2 for UDB, поэтому будут рассмотрены только они.

```
UDB_COLVALUE = S390_COLVALUE
UDB_VALCOUNT = S390_FrequencyF * S390_CARD
```

В каталоге SYSIBM.SYSCOLUMNS в DB2 for OS/390 нет также показателей столбцов SEQNO, которые требуются для DB2 for UDB. Поэтому утилита db2look генерирует их автоматически.

Статистика подэлементов

Есть опция для сбора и использования статистики подэлементов. Это статистика о содержимом данных в столбцах, когда структура данных имеет вид последовательностей подполей и подэлементов, разделенных пробелами.

Например, предположим, что база данных содержит таблицу DOCUMENTS, в которой каждая строка описывает документ, и в DOCUMENTS есть столбец под названием KEYWORDS, содержащий список соответствующих ключевых слов, описывающих текст этого документа. Значения в KEYWORDS могут выглядеть так:

```
'база моделирование аналитика бизнес интеллект'
'моделирование модель дрозофила размножение температура'
'лес хвойный почва эрозия дождевой'
'лес температура почва осадки пожар'
```

В этом примере каждое значение столбца состоит из 5 подэлементов, каждый из которых является словом (ключевым словом), отделенным от других одним пробелом.

Для запросов, содержащих предикаты LIKE для таких столбцов с использованием символа %, который означает соответствие всем символам:

```
SELECT .... FROM DOCUMENTS WHERE KEYWORDS LIKE '%моделирование%'
```

оптимизатору часто полезно знать основные статистические характеристики структуры подэлементов этого столбца, а именно:

SUB_COUNT
: Среднее число подэлементов.

SUB_DELIM_LENGTH
: Средняя длина ограничителя, отделяющего подэлемент, где ограничитель в данном контексте - один или несколько последовательных символов пробела.

В приведенном примере SUB_COUNT равняется 5, а SUB_DELIM_LENGTH - 1, поскольку все ограничители состоят из одного символа пробела.

Системный администратор управляет сбором и использованием такой статистики при помощи расширения переменной реестра DB2_LIKE_VARCHAR. Эта переменная реестра определяет, как оптимизатор DB2 UDB работает с предикатами вида:

```
COLUMN LIKE '%xxxxxx'
```

где *xxxxxx* - любая строка символов; то есть с любыми предикатами LIKE, поисковое значение которых начинается с символа %. (Оно может и заканчиваться символом %, но это не обязательно). Далее мы будем называть их ″предикаты LIKE с подстановкой″. Для каждого предиката оптимизатор должен оценить, какое число строк отвечает этому предикату. Для предикатов LIKE с подстановкой оптимизатор полагает, что у COLUMN, для которого найдено совпадение, имеется структура из последовательности сочлененных элементов, образующих собственно столбец, и оценивает длину каждого из этих элементов на основе длины строки символов, исключая из нее начальные и конечные символы %. Новый синтаксис:

```
db2set  DB2_LIKE_VARCHAR=[Y|N|S|num1][,Y|N|num2]

  где
- первый член (перед запятой) означает следующее
   (только для столбцов, не имеющих позитивной
   статистики подэлементов)

   S                 Использовать алгоритм, как в DB2 версии 2.
   N                 Использовать алгоритм подэлементов с фиксированной длиной.
   Y  (по умолчанию) Использовать алгоритм подэлементов с переменной длиной с
                     используемым по умолчанию значением параметра алгоритма.
   num1              Использовать алгоритм подэлементов с переменной длиной
                     со значением num1 для параметра алгоритма.

- второй член (после запятой) означает:
   N  (по умолчанию) Не собирать или не использовать статистики подэлементов.
   Y                 Собирать статистики подэлементов. Использовать алгоритм
                     подэлементов с переменной длиной, использующий эти
                     статистики, а также устанавливаемое по умолчанию значение
                     для параметра алгоритма в случае столбцов с позитивными
                     статистиками подэлементов.
   num2              Собирать статистики подэлементов. Использовать алгоритм
                     подэлементов с переменной длиной, использующий эти
                     статистики, а также значение num2
                     для параметра алгоритма в случае столбцов с позитивными
                     статистиками подэлементов.
```

Если значение DB2_LIKE_VARCHAR содержит только первый член, статистика для подэлементов не собирается, а вся ранее собранная статистика игнорируется. Указанное значение влияет подсчет оптимизатором селективности предикатов LIKE с подстановкой так же, как и раньше, то есть:

- Если указано значение S, оптимизатор использует такой же алгоритм, как использовался в DB2 версии 2, в которой не предполагалось подэлементной модели.

- Если указано значение N, оптимизатор использует алгоритм, предполагающий подэлементную модель, и считает, что COLUMN имеет фиксированную длину даже в том случае, когда он определен как имеющий переменную длину.
- Если указано значение Y (по умолчанию) или константа с плавающей запятой, оптимизатор использует алгоритм, предполагающий подэлементную модель, и распознает, что COLUMN имеет переменную длину, если он так определен. Это также подразумевает получение статистики подэлементов непосредственно из запроса, а не из данных. Этот алгоритм содержит параметр (″параметр алгоритма″), указывающий, насколько элемент длиннее, чем строка символов, заключенная в символы %.
- Если указано Y, для параметра алгоритма оптимизатор по умолчанию использует значение 1,9.
- Если указана константа с плавающей запятой, для параметра алгоритма оптимизатор использует это значение. Константа должна быть в диапазоне от 0 до 6,2.

Если значение DB2_LIKE_VARCHAR содержит два члена и второй из них - Y или константа с плавающей запятой, статистика подэлементов собирается во время операции RUNSTATS по столбцам типа CHAR, VARCHAR, GRAPHIC или VARGRAPHIC со строками наборов однобайтных символов и используются во время компиляции запросов, включающих в себя предикаты LIKE с подстановкой. Оптимизатор использует алгоритм, предполагающий подэлементную модель, и вычисляет селективность предиката по показателям SUB_COUNT и SUB_DELIM_LENGTH, а также по параметру алгоритма. Параметр алгоритма указывается так же, как описано выше, то есть:

- Если указано Y, для параметра алгоритма оптимизатор по умолчанию использует значение 1,9.
- Если указана константа с плавающей запятой, для параметра алгоритма оптимизатор использует это значение. Константа должна быть в диапазоне от 0 до 6,2.

Если во время компиляции оптимизатор обнаружит, что статистики подэлементов для включенного в запрос столбца не собраны, он будет использовать ″дедуктивный″ алгоритм подэлементов, то есть алгоритм, используемый, если указан только первый член DB2_LIKE_VARCHAR. Таким образом, чтобы статистики подэлементов были использованы оптимизатором, второй член DB2_LIKE_VARCHAR должен быть установлен и во время RUNSTATS, и во время компиляции.

Значения статистик подэлементов можно посмотреть, запросив SYSIBM.SYSCOLUMNS. Например:

```
select substr(NAME,1,16), SUB_COUNT, SUB_DELIM_LENGTH
   from sysibm.syscolumns where tbname = 'DOCUMENTS'
```

Столбцов SUB_COUNT и SUB_DELIM_LENGTH нет в производной таблице статистики SYSSTAT.COLUMNS и поэтому изменить их нельзя.

Примечание: При использовании этой опции RUNSTATS может работать дольше. Например, RUNSTATS может работать на 15 - 40% дольше для таблицы со столбцами из пяти символов, если не используются опции DETAILED и DISTRIBUTION. Если указана опция DETAILED или DISTRIBUTION, процентное увеличение будет меньше, несмотря на то, что абсолютное значение увеличения останется таким же. Если вы собираетесь использовать эту опцию, надо сопоставить эти дополнительные расходы и повышение производительности запроса.

Глава 6. Компилятор SQL

При компиляции запроса SQL выполняется ряд действий, предшествующих исполнению или сохранению в системном каталоге “наилучшего” плана доступа.

В среде многораздельных баз данных все действия компилятора SQL над запросом SQL происходят в том разделе базы данных, с которым установлено соединение. Перед началом исполнения компилированный запрос посылается на все разделы базы данных.

Сведения о выполняемых компилятором SQL действиях приводятся в следующих темах:

- Обзор компилятора SQL
- Перезапись запросов компилятором SQL
- Объединение операций
- Перемещение операции
- Преобразование предикатов
- Концепции доступа к данным и оптимизация
- Стратегии оптимизации для внутрираздельного параллелизма
- Автоматические сводные таблицы
- Фазы компиляции запросов к базам данных объединения

Внешние факторы, способные влиять на результаты работы компилятора, описаны в разделах:

- “Глава 3. Особенности прикладного программирования” на стр. 43
- “Глава 4. Факторы среды” на стр. 95
- “Глава 5. Статистика системного каталога” на стр. 117.

“Глава 7. Возможность объяснения SQL” на стр. 225 описывает возможности проверки выбранного компилятором SQL плана доступа.

Обзор компилятора SQL

Перед составлением выполняемого плана доступа компилятор SQL выполняет ряд действий. Эти действия показаны на рис. 12 на стр. 156.

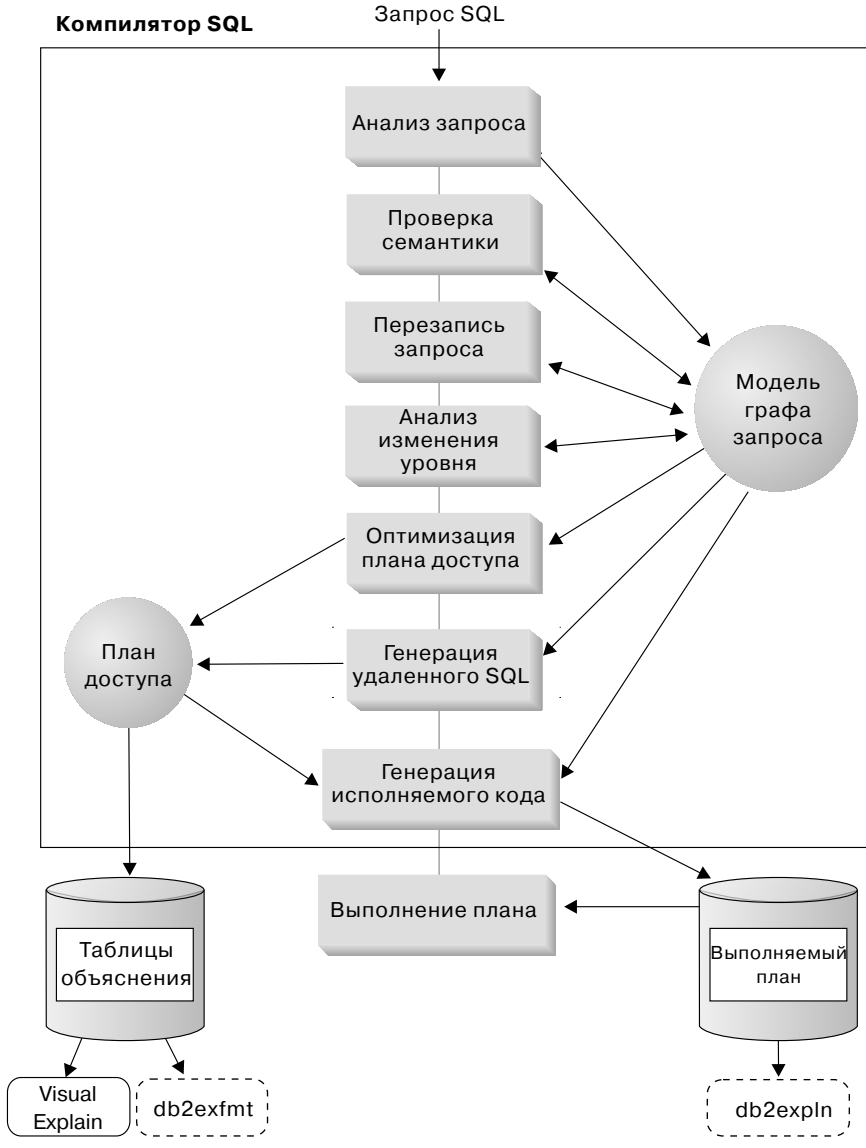


Рисунок 12. Действия, выполняемые компилятором SQL

Из этой диаграммы видно, что ключевой компонент компилятора SQL - **модель графа запроса**. *Модель графа запроса* - внутренняя база данных в памяти, используемая для представления запроса в процессе его компиляции, как описано ниже:

- **Анализ запроса**

  Первой задачей компилятора SQL является проверка правильности синтаксиса запроса SQL. При обнаружении синтаксических ошибок

компилятор SQL прекращает обработку и возвращает соответствующую ошибку SQL программе, пытавшейся компилировать оператор SQL. По завершении анализа создается внутреннее представление запроса.

- **Проверка семантики**

  Вторая задача компилятора - убедиться в отсутствии несоответствий между частями оператора. Простой пример семантической проверки - проверка, что тип данных в столбце для скалярной функции YEAR - тип дата-время. Кроме того, на втором этапе компилятор добавляет к модели графа запроса бихевиоральную семантику, в том числе влияние реляционных ограничений, проверочных ограничений таблицы, триггеров и производных таблиц.

  Модель графа запроса содержит всю семантику запросов, включая блокировки запросов, подзапросы, корреляции, порожденные таблицы, выражения, типы данных, преобразования типов данных и кодовых страниц, а также ключи разделения.

- **Перезапись запроса**

  На третьей стадии компилятор SQL использует глобальную семантику модели графа запроса для преобразования запроса в форму, легче поддающуюся оптимизации. Например, компилятор может переместить предикат, изменяя уровень его применения и в потенциале улучшая производительность запроса. Этот тип перемещения операций называется *общим изменением уровня предиката*. Дополнительную информацию смотрите в разделе “Перезапись запросов компилятором SQL” на стр. 159.

  При работе в среде многораздельной базы данных некоторые операции запросов требуют большей вычислительной работы:

  – Суммирование
  – Перераспределение строк
  – Связанные подзапросы.

  Связанный подзапрос - подзапрос, содержащий ссылку на столбец внешней по отношению к нему таблицы.

  В этой среде некоторые запросы могут в результате перезаписи стать несвязанными.

  Передаваемый запрос сохраняется в модели графа запроса.

- **Анализ изменения уровня (базы данных объединения)**

  Главной задачей данного этапа является рекомендация оптимизатору DB2, можно ли выполнить операцию на источнике данных (“изменить ее уровень”). Этот тип изменения уровня характерен для запросов к источникам данных и представляет собой расширение обычных операций изменения уровня предикатов.

Если не выполняются запросы к базе данных объединения, этот шаг пропускается. Дополнительную информацию смотрите в разделе “Анализ изменения уровня” на стр. 209.

- **Оптимизация плана доступа**

  Оптимизатор SQL как часть компилятора SQL, используя модель графа запроса в качестве исходных данных, генерирует несколько альтернативных планов выполнения требования пользователя. Он оценивает затраты на выполнение каждого из этих планов, используя статистику для таблиц, индексов, столбцов и функций, и выбирает план с наименьшей оценкой затрат. Оптимизатор использует модель графа запроса для анализа семантики запросов и получения информации о широком спектре факторов, включая информацию об индексах, базовых и порожденных таблицах, подзапросах, корреляциях и рекурсии.

  Оптимизатор способен также может рассмотреть другой тип операции изменения уровня: *группировку и сортировку*, что позволяет улучшить производительность, переместив выполнение этих операций на компонент Службы управления данными. Дополнительную информацию смотрите в разделе “Операции изменения уровня группировки и сортировки” на стр. 202.

  Кроме того, определяя выбор размера страниц, оптимизатор учитывает размеры пулов буферов. Принимается во внимание и то, включает ли среда многораздельную базу данных, а также возможность улучшения выбранного плана благодаря использованию внутреннего параллелизма обработки запроса в симметричной мультипроцессорной (SMP) среде. Эта информация используется оптимизатором при выборе наилучшего плана доступа для запроса. Дополнительную информацию смотрите в разделе “Концепции доступа к данным и оптимизация” на стр. 169.

  Результатом этого шага компилятора SQL является “план доступа”. План доступа дает основу для информации, помещаемой в таблицы объяснения. Информацию, используемую для выработки плана доступа, можно сохранить в снимке объяснения. (Дополнительную информацию об объяснении смотрите в разделе “Глава 7. Возможность объяснения SQL” на стр. 225).

- **Генерация удаленного SQL (базы данных объединения)**

  Выбранный оптимизатором DB2 конечный план может содержать серию шагов, которые могут выполняться на удаленном источнике данных. Для операций, выполняемых на каждом источнике данных, на этапе генерации удаленного SQL создается эффективный оператор SQL на базе диалекта SQL этого источника данных.

  Если не выполняются запросы к базе данных объединения, этот шаг пропускается. Дополнительную информацию смотрите в разделе “Генерация удаленного SQL и глобальная оптимизация” на стр. 218.

- **Генерация “исполняемого” кода**

  На последнем этапе компилятор SQL использует *план доступа* и модель графа запроса для создания исполняемого плана доступа для запроса. На этапе

генерации кода информация модели графа запроса позволяет избежать повторного вычисления выражений, которые при выполнении запроса достаточно вычислить один раз. Примеры такой оптимизации возможна - преобразование кодовых страниц и использование переменных хоста.

Информация о планах доступа для статического SQL хранится в таблицах системного каталога. При выполнении пакета менеджер баз данных использует информацию таблиц системного каталога, чтобы определить, как получить доступ к данным и как представить результаты запроса. Именно эту информацию использует *db2expln*. (Дополнительную информацию об объяснении смотрите в разделе “Глава 7. Возможность объяснения SQL” на стр. 225).

Если желательна высокая производительность, рекомендуется периодически выполнять RUNSTATS для используемых в запросах таблиц. Тогда у оптимизатора будет более релевантная статистическая информация о природе данных. Чтобы использовать преимущества новых данных статистики нужно также выполнить повторное связывание прикладной программы. Если RUNSTATS не выполнялась (или оптимизатор полагает, что RUNSTATS выполнялась на пустых или почти пустых таблицах), оптимизатор может использовать статистику по умолчанию или пытаться вывести некую статистику на основе числа страниц файлов, используемых для хранения таблицы на диске (FPAGES).

Перезапись запросов компилятором SQL

Работа компилятора SQL включает в себя стадию перезаписи запросов, на которой операторы SQL преобразуются в форму, упрощающую оптимизацию, что может улучшить выбранный путь доступа. Перезапись запросов особенно важна для сложных запросов, например со множеством подзапросов или объединений. Такие сложные запросы часто создаются средствами генерации запросов.

Число правил перезаписи запросов, применяемых к оператору SQL, можно изменить, изменив класс оптимизации (смотрите в разделе “Настройка класса оптимизации” на стр. 70).

Некоторые результаты перезаписи запросов можно посмотреть, воспользовавшись возможностью объяснения или наглядным объяснением.

Компилятор SQL может выполнять перезапись трех основных категорий:
- Объединение операций
- Перемещение операции
- Преобразование предикатов.

Объединение операций

Компилятор SQL будет перезаписывать запросы, пытаясь объединить операции запросов и построить запрос с наименьшим числом операций, особенно операций SELECT. В следующих примерах показаны некоторые операции, которые могут быть объединены компилятором SQL:

- Пример - Слияние производных таблиц

  Использование в операторе SELECT производных таблиц может ограничить порядок объединения таблицы, а также ввести избыточное объединение таблиц. Эти ограничения можно снять, объединив эти производные таблицы при перезаписи запроса.

- Пример - Преобразования подзапроса в объединение

  Если оптимизатор находит в операторе SELECT подзапрос, это может ограничить выбор порядка обработки таблиц.

- Пример - Исключение избыточного объединения

  Во время перезаписи запроса могут быть удалены избыточные объединения для дальнейшего упрощения оптимизируемого оператора SELECT.

- Пример - Совместно используемое суммирование

  При использовании различных функций перезапись запроса может привести к уменьшению числа необходимых вычислений.

Пример - Слияние производных таблиц

Предположим, вы обращаетесь к двум нижеприведенным производным таблицам таблицы EMPLOYEE; в одной показаны сотрудники с высоким уровнем образования, а в другой - сотрудники, зарабатывающие более 35000 долларов:

```
CREATE VIEW EMP_EDUCATION (EMPNO, FIRSTNME, LASTNAME, EDLEVEL) AS
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL
     FROM EMPLOYEE
 WHERE EDLEVEL > 17
CREATE VIEW EMP_SALARIES (EMPNO, FIRSTNAME, LASTNAME, SALARY) AS
SELECT EMPNO, FIRSTNME, LASTNAME, SALARY
     FROM EMPLOYEE
 WHERE SALARY > 35000
```

Теперь предположим, что выполняется нижеприведенный запрос списка сотрудников с высоким уровнем образования и заработком более 35000 долларов:

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMP_EDUCATION E1,
       EMP_SALARIES  E2
 WHERE E1.EMPNO = E2.EMPNO
```

При перезаписи запроса может быть создан следующий запрос, в котором эти две производные таблицы будут слиты:

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

Объединив операторы SELECT из этих двух производных таблиц с оператором SELECT, написанным пользователем, оптимизатор сможет при выборе плана доступа рассмотреть больше вариантов. Кроме того, если две производные таблицы, которые были слиты, используют одну базовую таблицу, может быть выполнена дополнительная перезапись, как описано в разделе “Пример - Исключение избыточного объединения”.

Пример - Преобразования подзапроса в объединение

Компилятор SQL рассматривает запрос с содержащимся в нем подзапросом, например:

```
SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO
     FROM EMPLOYEE
WHERE WORKDEPT IN
      (SELECT DEPTNO
  FROM DEPARTMENT
        WHERE DEPTNAME = 'OPERATIONS')
```

и преобразовывать его в запрос с объединением:

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME, PHONENO
  FROM EMPLOYEE EMP,
       DEPARTMENT DEPT
 WHERE EMP.WORKDEPT = DEPT.DEPTNO
   AND DEPT.DEPTNAME = 'OPERATIONS'
```

Обычно объединение выполняется значительно эффективнее, чем подзапрос.

Пример - Исключение избыточного объединения

Иногда могут быть написаны или сгенерированы запросы, в которых есть избыточные объединения. Кроме того, запросы, подобные приведенному ниже, могут быть получены на стадии перезаписи запроса, как описано в разделе “Пример - Слияние производных таблиц” на стр. 160.

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

Компилятор SQL может упростить этот запрос, исключив из него объединение:

```
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL, SALARY
     FROM EMPLOYEE
 WHERE EDLEVEL > 17
   AND SALARY  > 35000
```

В другом примере предполагается, что существует реляционная связь между таблицами примера EMPLOYEE и DEPARTMENT по номеру отдела. Сначала создается производная таблица.

```
CREATE VIEW PEPLVIEW
   AS SELECT FIRSTNME, LASTNAME, SALARY, DEPTNO, DEPTNAME, MGRNO
         FROM EMPLOYEE E DEPARTMENT D
         WHERE E.WORKDEPT = D.DEPTNO
```

Тогда запрос типа:

```
SELECT LASTNAME, SALARY
   FROM PEPLVIEW
```

преобразуется в:

```
SELECT LASTNAME, SALARY
     FROM EMPLOYEE
   WHERE WORKDEPT NOT NULL
```

Обратите внимание на то, что в этой ситуации, даже если пользователь знает о возможности перезаписи данного запроса, он не может воспользоваться ей, поскольку у него нет доступа к базовым таблицам. Ему может быть разрешено обращение только к производной таблице (показанной выше). Следовательно, такой тип оптимизации должен выполняться в менеджере баз данных.

Избыточность возможна в объединениях реляционной целостности, где:

- Производные таблицы определены с объединением
- Запросы генерируются автоматически.

  В менеджерах запросов есть, например, средства автоматизации, которые не дают пользователям писать оптимизированные запросы.

Пример - Совместно используемое суммирование

Использование в запросе нескольких функций может вызвать определенное число вычислений, которые будут отнимать время. Уменьшение числа вычислений, выполняемых в запросе, приводит к улучшению плана. Компилятор SQL рассматривает запрос, в котором используется несколько функций, например:

```
SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  AVG(SALARY+BONUS+COMM) AS OAVG,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE;
```

и преобразует его так:

```
SELECT OSUM,
  OSUM/OCOUNT
  OCOUNT
FROM (SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE) AS SHARED_AGG;
```

Такая перезапись позволяет в запросе вместо двух суммирований и двух подсчетов выполнять одно суммирование и один подсчет.

Перемещение операции

Компилятор SQL будет перезаписывать запросы, пытаясь переместить операции запроса и сконструировать запрос с наименьшим числом операций и предикатов. В следующих примерах показаны некоторые операции, которые могут быть перемещены компилятором SQL:

- Пример - Устранение DISTINCT

  При перезаписи запроса оптимизатор может изменить место выполнения операции DISTINCT, чтобы снизить ее стоимость. В приведенном ниже примере операция DISTINCT полностью удаляется.

- Пример - изменение уровня общего предиката

  При перезаписи запроса можно менять порядок применения предикатов, чтобы выполнять наиболее избирательные предикаты запроса как можно раньше.

- Пример - Декорреляция

  В среде многораздельной базы данных перемещение наборов данных между разделами базы данных - процесс дорогостоящий. Уменьшение размера и/или числа передач в другие разделы базы - одна из целей перезаписи запросов.

Пример - Устранение DISTINCT

Если в качестве первичного ключа таблицы EMPLOYEE был определен столбец EMPNO, следующий запрос:

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME
     FROM EMPLOYEE
```

будет перезаписан с удалением условия DISTINCT:

```
SELECT EMPNO, FIRSTNME, LASTNAME
       FROM EMPLOYEE
```

В этом примере, поскольку выбирается первичный ключ, компилятор SQL знает, что каждая возвращаемая строка уже уникальна. В данном случае ключевое слово DISTINCT избыточно. Если не перезаписать данный запрос, оптимизатор построит план с необходимой для получения различных значения обработкой (например, с сортировкой).

Пример - изменение уровня общего предиката

Изменение уровня применения предиката может улучшить производительность. Возьмем, например, производную таблицу “D11” со списком всех сотрудников в отделе:

```
CREATE VIEW D11_EMPLOYEE
 (EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM)
AS SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM
     FROM EMPLOYEE
    WHERE WORKDEPT = 'D11'
```

и следующий запрос:

```
SELECT FIRSTNME, PHONENO
  FROM D11_EMPLOYEE
 WHERE LASTNAME = 'BROWN'
```

На стадии перезаписи запроса компилятор вынесет предикат `LASTNAME = 'BROWN'` с уровня производной таблицы D11_EMPLOYEE. Это позволит выполнять данный предикат быстрее и потенциально более эффективно. Фактический запрос в этом примере будет следующим:

```
SELECT FIRSTNME, PHONENO
     FROM EMPLOYEE
 WHERE LASTNAME = 'BROWN'
   AND WORKDEPT = 'D11'
```

Изменение уровня предикатов не ограничивается производными таблицами. Другие ситуациям, в которых может быть применен этот метод - условия UNION, условия GROUP BY и полученные таблицы (вложенные и общие табличные выражения).

Пример - Декорреляция

В среде многораздельных баз данных компилятор SQL может перезаписать следующий запрос:

Найти всех сотрудников, работающих над программными проектами и получающих оплату меньше средней.

```
SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
       E.SALARY+E.BONUS+E.COMM AS COMPENSATION
  FROM EMPLOYEE E, PROJECT P
WHERE P.EMPNO = E.EMPNO
  AND P.PROJNAME LIKE '%PROGRAMMING%'
  AND E.SALARY+E.BONUS+E.COMM <
     (SELECT AVG(E1.SALARY+E1.BONUS+E1.COMM)
        FROM EMPLOYEE E1, PROJECT P1
        WHERE P1.PROJNAME LIKE '%PROGRAMMING%'
          AND P1.PROJNO = A.PROJNO
          AND E1.EMPNO = P1.EMPNO)
```

Это коррелированный запрос, и сомнительно, чтобы и PROJECT, и EMPLOYEE были разбиты на разделы по PROJNO, поэтому вероятно, что каждый проект распределен по всем разделам базы данных. Кроме того, оценка подзапроса здесь может вычисляться многократно.

Компилятор SQL может перезаписать данный запрос так:

- Определить отдельный список сотрудников, работающих над программными проектами, и назвать его DIST_PROJS. Этот список должен быть отдельным, чтобы для каждого проекта вычисление среднего с гарантией выполнялось только один раз:

```
WITH DIST_PROJS(PROJNO, EMPNO) AS
(SELECT DISTINCT PROJNO, EMPNO
 FROM PROJECT P1
 WHERE P1.PROJNAME LIKE '%PROGRAMMING%')
```

- Используя этот особый список сотрудников, работающих над программными проектами, объединить его с таблицей сотрудников, чтобы найти среднюю величину оплаты для проекта - AVG_PER_PROJ:

```
AVG_PER_PROJ(PROJNO, AVG_COMP) AS
(SELECT P2.PROJNO, AVG(E1.SALARY+E1.BONUS+E1.COMM)
 FROM EMPLOYEE E1, DIST_PROJS P2
 WHERE E1.EMPNO = P2.EMPNO
 GROUP BY P2.PROJNO)
```

- Тогда новый запрос будет выглядеть так:

```
SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
       E.SALARY+E.BONUS+E.COMM AS COMPENSATION
 FROM PROJECT P, EMPLOYEE E, AVG_PER_PROG A
WHERE P.EMPNO = E.EMPNO
  AND P.PROJNAME LIKE '%PROGRAMMING%'
  AND P.PROJNO = A.PROJNO
  AND E.SALARY+E.BONUS+E.COMM < A.AVG_COMP
```

Перезаписанный запрос SQL вычисляет значение AVG_COMP для проекта (AVG_PRE_PROJ) и затем передает результат во все разделы базы данных, где находится таблица EMPLOYEE.

Преобразование предикатов

Компилятор SQL будет перезаписывать подзапросы, преобразуя существующие предикаты в более оптимальные для конкретного подзапроса. В следующих примерах показаны некоторые предикаты, которые могут быть преобразованы компилятором SQL:

- Пример - Добавление неявных предикатов

  При перезаписи запроса в него могут быть добавлены предикаты, что позволяет оптимизатору рассмотреть дополнительные объединения таблиц при выборе для данного подзапроса оптимального плана доступа.

- Пример - Преобразования OR в IN

  При перезаписи запроса предикат OR может быть преобразован в предикат IN для выбора более эффективного плана доступа. Кроме того, компилятор SQL

может преобразовать предикат IN в предикат OR, если такое преобразование будет способствовать выбору более эффективного плана доступа.

Пример - Добавление неявных предикатов

Следующий запрос генерирует список руководителей, чьи отделы подотчетны “E01”, и проектов, за которые эти руководители ответственны:

```
SELECT DEPT.DEPTNAME DEPT.MGRNO, EMP.LASTNAME, PROJ.PROJNAME
  FROM DEPARTMENT DEPT,
       EMPLOYEE   EMP,
       PROJECT    PROJ
 WHERE DEPT.ADMRDEPT = 'E01'
   AND DEPT.MGRNO = EMP.EMPNO
   AND EMP.EMPNO  = PROJ.RESPEMP
```

В перезаписанный запрос будет добавлен следующий неявный предикат:

```
DEPT.MGRNO = PROJ.RESPEMP
```

В результате оптимизатор сможет рассмотреть дополнительные варианты объединения, когда будет выбирать оптимальный план доступа для запроса.

Кроме вышеописанного транзитивного замыкания предиката, при перезаписи запроса определяются дополнительные локальные предикаты на основе транзитивности, подразумеваемой предикатами равенства. Например, следующий запрос выводит список названий отделов, номера которых больше “E00”, и имен работающих в них сотрудников.

```
SELECT EMPNO, LASTNAME, FIRSTNAME, DEPTNO, DEPTNAME
  FROM EMPLOYEE EMP,
       DEPARTMENT DEPT
 WHERE EMP.WORKDEPT = DEPT.DEPTNO
   AND DEPT.DEPTNO > 'E00'
```

Для этого запроса на стадии перезаписи будет добавлен следующий неявный предикат:

```
EMP.WORKDEPT > 'E00'
```

В результате этой перезаписи оптимизатор уменьшает число объединяемых строк.

Пример - Преобразования OR в IN

Предположим, что условие OR соединяет два или более простых предикатов равенства по одному столбцу, как в следующем примере:

```
SELECT *
        FROM EMPLOYEE
     WHERE DEPTNO = 'D11'
       OR DEPTNO = 'D21'
       OR DEPTNO = 'E21'
```

Если индекса по столбцу DEPTNO нет, преобразование условия OR в следующий предикат IN повысит эффективность обработки запроса:

```
SELECT *
        FROM EMPLOYEE
    WHERE DEPTNO IN ('D11', 'D21', 'E21')
```

Примечание: В некоторых случаях менеджер баз данных может преобразовать предикат IN в набор условий OR, чтобы сделать возможным выполнение операции OR над индексами. Дополнительную информацию об операции OR над индексами смотрите в разделе “Доступ к нескольким индексам” на стр. 176.

Расчет корреляции столбцов

Некоторые программы содержат запросы, использующие объединения двух таблиц по нескольким предикатам. Такая ситуация, хоть и выглядит сложно, встречается, когда вы пытаетесь определить отношение между похожими или родственными столбцами разных таблиц.

Например, изготовитель производит изделия из сырья разного цвета, упругости и качества. Конечный продукт имеет тот же цвет и упругость, что и исходное сырье. Изготовитель дает запрос:

```
SELECT PRODUCT.NAME, RAWMATERIAL.QUALITY FROM PRODUCT, RAWMATERIAL
   WHERE PRODUCT.COLOR       =  RAWMATERIAL.COLOR
     AND PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

Этот запрос возвращает названия и качество сырья для всех изделий. Здесь есть два предиката объединения:

```
PRODUCT.COLOR       =  RAWMATERIAL.COLOR
PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

Когда оптимизатор DB2 UDB выбирает план выполнения данного запроса, он вычисляет, насколько селективен каждый из двух предикатов, подразумевая при этом, что они независимы, то есть для каждого цвета существуют все варианты упругости и, наоборот, для каждой степени упругости существует все варианты цвета сырья. Затем для вычисления суммарной селективности пары предикатов он применяет статистику, сколько в каждой таблице имеется степеней упругости и различных цветов. На этой основе он может предпочесть, к примеру, объединение с вложенным циклом объединению слияния или наоборот.

Возможно однако, что эти два предиката не являются независимыми. Например, имеются высокоупругие материалы только некоторых цветов, а низкоупругие материалы - некоторых других цветов (не таких, как цвета высокоупругих материалов). Тогда объединенная селективность предикатов меньше (то есть исключает меньше строк), и запрос вернет больше строк. Чтобы это понять, вообразите предельный случай, когда для каждого цвета есть только одна

степень упругости, и наоборот. Теперь один любой предикат можно полностью исключить, поскольку он является логическим следствием другого. Выбор оптимизатора может оказаться не наилучшим, например, будет выбран план объединения с вложенным циклом, хотя объединение слияния выполнялось бы быстрее.

С помощью других программ базы данных администраторы пытаются решить эту проблему производительности, искусственно изменяя статистику в каталоге, чтобы сделать один из предикатов менее селективным, но этот подход грозит нежелательными побочными действиями на другие запросы.

Оптимизатор UDB DB2 сделает попытку обнаружить и компенсировать корреляции предикатов объединения, если вы:

1. Определили индексы уникальности на коррелирующих столбцах, то есть на столбцах таблицы, встречающихся в коррелирующих предикатах.
2. Не задали для переменной реестра DB2_CORRELATED_PREDICATES значение “NO”.

В приведенном примере вы можете определить индекс уникальности:

```
PRODUCT.COLOR, PRODUCT.ELASTICITY
```

или

```
RAWMATERIAL.COLOR, RAWMATERIAL.ELASTICITY
```

или оба этих индекса.

Чтобы обнаружилась корреляция, все собственные (не включаемые) столбцы этого индекса должны коррелировать между собой. Этот индекс может (необязательно) содержать включаемые столбцы.

В целом в предикатах объединения может содержаться более двух коррелирующих столбцов, поэтому следует убедиться, что вы определили индекс уникальности, охватывающий все эти столбцы.

Во многих случаях коррелирующие столбцы одной таблицы образуют ее первичный ключ. Первичный ключ всегда уникален, поэтому если он существует для коррелирующих столбцов, не требуется определять другой индекс уникальности.

Проделав это, убедитесь, что статистика таблиц не устарела, и что истинность значений не утрачена, например, в результате попытки повлиять на оптимизатор.

Оптимизатор будет использовать показатели FIRSTnKEYCARD и FULLKEYCARD статистики индексов уникальности для выявления корреляции

и динамически корректировать комбинированные селективности коррелирующих предикатов, повышая таким образом точность оценки размера и затрат объединения.

В дополнение к корреляции предиката JOIN оптимизатор подсчитывает корреляцию с предикатами простых уравнений типа COL = “константа”. Рассмотрим, к примеру, таблицу автомобилей различных типов, каждый из которых описывается столбцами MAKE (изготовитель), MODEL, STYLE (тип кузова - седан, универсал, спортивный), YEAR и COLOR. Предикаты COLOR практически независимы от предикатов MAKE, MODEL, STYLE или YEAR, поскольку почти каждый изготовитель год за годом представляет весь стандартный набор расцветок для каждой модели или стиля. Однако предикаты MAKE и MODEL, безусловно, зависимы, потому что только один изготовитель производит модель с определенным названием. Идентичные названия моделей у разных изготовителей встречаются крайне редко и явно нежелательны для производителей. Если есть индекс для двух столбцов MAKE и MODEL, оптимизатор использует его статистику для определения объединенного числа отдельных значений и установления оценки селективности или мощности для корреляции между этими двумя столбцами. Для других предикатов (не предикатов объединения) оптимизатору для выполнения корректировки не требуется индекс уникальности.

Концепции доступа к данным и оптимизация

При компиляции оператора SQL оптимизатор SQL оценивает затраты на выполнение запроса разными способами. На основе этой оценки оптимизатор выбирает оптимальный с его точки зрения план доступа. *План доступа* задает порядок операций, требуемых для выполнения оператора SQL. При связывании прикладной программы создается *пакет*. Этот пакет содержит планы доступа для всех статических операторов SQL данной прикладной программы. Планы доступа для динамических операторов SQL создаются во время выполнения программы.

Есть два способа доступа к данным таблицы: непосредственное чтение таблицы (реляционный просмотр) и предварительное обращение к индексу таблицы (просмотр индекса).

При *реляционном просмотре* менеджер баз данных последовательно обращается к каждой строке таблицы. Чтобы узнать, как действует просмотр индекса, обратитесь к разделу “Концепции просмотра индекса” на стр. 170; в разделе “Сравнение реляционного просмотра с просмотром индекса” на стр. 180 описано, при каких условиях используется тот или другой тип просмотра.

Другие методы, используемые в плане доступа для доступа к данным таблиц и получения результатов запроса, описаны в следующих темах:

- “Терминология предикатов” на стр. 180

- “Концепции объединения” на стр. 182
- “Стратегии объединения в многораздельной базе данных” на стр. 193
- “Влияние сортировки на оптимизатор” на стр. 201.

Другие темы, связанные с данной:

- Раздел “Настройка класса оптимизации” на стр. 70 содержит информацию о том, как управлять числом альтернативных планов доступа, оцениваемых компилятором SQL
- “Глава 7. Возможность объяснения SQL” на стр. 225 содержит сведения о том, как получить информацию о плане доступа, выбранном компилятором SQL.

Концепции просмотра индекса

При *просмотре индекса* менеджер баз данных обращается к индексу для выполнения следующих задач:

- Сокращение набора подходящих строк (путем просмотра строк в определенном интервале индекса) перед обращением к базовой таблице. *Интервал просмотра* индекса (начальная и конечная точки просмотра) определяется значениями запроса, с которыми сравниваются значения в столбцах индекса.
- Вывод результатов в определенном порядке.
- Полная выдача запрошенных данных. Если все запрошенные данные содержатся в индексе, к базовой таблице можно не обращаться. Такой метод называется *доступом только к индексу*.

Просмотр индексов может выполняться в порядке, обратном порядку их определения. Дополнительную информацию смотрите в описании опции ALLOW REVERSE SCANS оператора CREATE INDEX в справочнике *SQL Reference*.

Далее изложены следующие темы:

- Структура индексов
- Просмотр индекса для ограничения интервала
- Просмотр индекса для упорядочения данных
- Доступ только к индексу
- Доступ к нескольким индексам
- Кластеризованные индексы
- Предварительная выборка страниц индекса.

Структура индексов

Менеджер баз данных использует для хранения индексов структуру B+-дерева. B+-дерево содержит один или несколько уровней, как показано на диаграмме (где `RID` - ID строки):

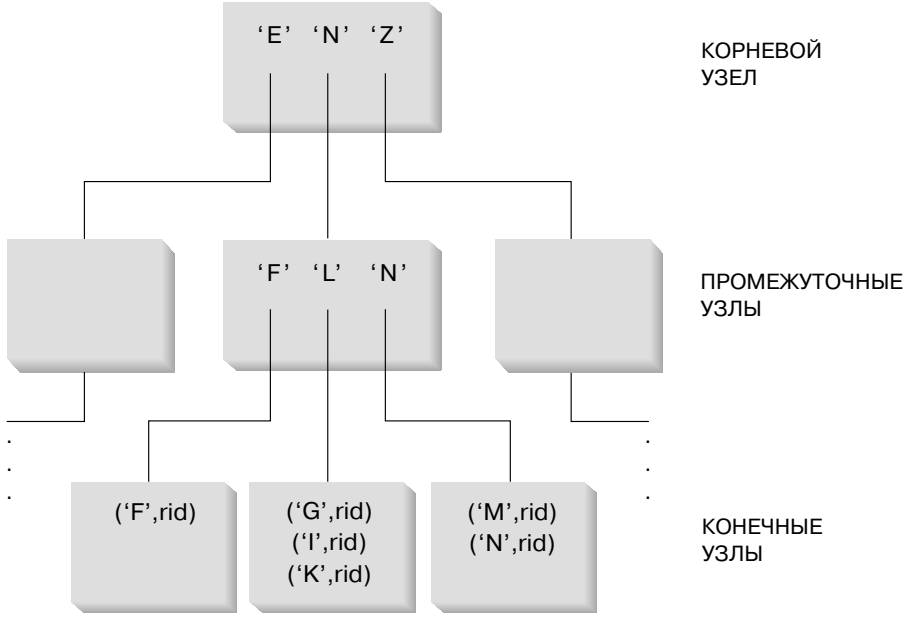


Рисунок 13. Структура B+-дерева

Верхний уровень называют *корневым узлом*. Нижний уровень состоит из *конечных узлов* (листьев), где хранятся сами значения индексных ключей и указатели на реальные строки таблицы. Уровни между корневым и конечными узлами называются *промежуточными узлами*.

При поиске отдельного значения индексного ключа менеджер индексов просматривает дерево индексов, начиная с корневого узла. Корень содержит по одному ключу для каждого узла следующего уровня. Значение каждого из этих ключей - это наибольшее существующее значение ключа для соответствующего узла следующего уровня. Например, в индексе три уровня, как показано на рис. 13, то чтобы найти значение индексного ключа, менеджер индексов должен просмотреть корневой узел в поиске первого ключевого значения, большего или равного искомому ключу. Этот ключ корневого узла укажет на конкретный промежуточный узел. Та же процедура нужна, чтобы определить, к какому конечному узлу перейти. Конечный индексный ключ должен обнаружиться в конечном узле. На рис. 13 искомый ключ - "I". Первый ключ в корневом узле, больший или равный "I" - это "N". Он указывает на средний узел следующего уровня. Первый ключ на этом промежуточном уровне, больший или равный "I" - это "L". Он указывает на конкретный конечный узел, где будет найден индексный ключ для "I" и соответствующие ID строки (ID строки для соответствующих строк базовой таблицы).

Примечание: На уровне конечных узлов могут находиться указатели на предыдущие конечные узлы. Это большое преимущество, так как,

обнаружив при просмотре дерева отдельное ключевое значение в индексе, менеджер индексов может для получения интервала значений просматривать конечные узлы в любом направлении. Просмотр в любом направлении возможен, только если при создании индекса был задан параметр ALLOW REVERSE SCANS.

Дополнительную информацию смотрите в описании опций оператора CREATE INDEX в справочнике *SQL Reference*.

Просмотр индекса для ограничения интервала

При определении, можно ли использовать индекс для данного конкретного запроса, оптимизатор оценивает каждый столбец индекса, начиная с первого, чтобы увидеть, можно ли его использовать, чтобы проверить:

- Любой из предикатов EQUAL в условии WHERE оператора
- Любые другие предикаты в условии WHERE.

Предикат - это элемент критерия поиска в условии WHERE, выражающий или подразумевающий операцию сравнения. Предикаты, которые можно использовать для ограничения интервала просмотра индекса, включают столбец индексов, для которого истинно одно из следующих условий:

- Значение из столбца индексов проверяется на равенство константе, переменной хоста, оцениваемому константой выражению или ключевому слову
- Результатом проверки столбца индексов является “IS NULL” или “IS NOT NULL”
- Производится проверка на равенство базовому подзапросу (то есть такому, который не содержит ANY, ALL или SOME), и подзапрос не содержит коррелирующей ссылки на столбец на его ближайший родительский блок запроса (то есть на SELECT, для которого этот подзапрос является подвыбором).
- Проверка на предикат неравенства при описанных ниже условиях.

Например, дан индекс со следующим определением:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

для задания границ просмотра индекса IX1 могут использоваться следующие предикаты:

```
  WHERE  NAME  = :hv1
AND   DEPT  = :hv2
```

или

```
WHERE  MGR  = :hv1
  AND  NAME = :hv2
  AND  DEPT = :hv3
```

Заметим, что во втором примере предикаты WHERE не обязаны указываться в порядке ключевых столбцов индекса. Хотя в примерах используются переменные хоста, маркеры параметров, выражения и константы дают тот же эффект.

Созданный с использованием параметра ALLOW REVERSE SCANS в операторе CREATE INDEX одиночный индекс можно просматривать в прямом или обратном направлении. Иными словами, такие индексы поддерживают просмотр как в направлении, заданном при их создании, так и в противоположном (обратном) направлении. Оператор может иметь следующий вид:

```
CREATE INDEX имя-индекса ON имя-таблицы (имя-столбца DESC) ALLOW REVERSE SCANS
```

В этом случае индекс (`имя-индекса`) формируется на основе DESCending (убывающих) значений столбца `имя-столбца`. Разрешение обратного просмотра означает, что несмотря на то, что индекс столбца определен для просмотра в порядке убывания, просмотр может вестись в порядке возрастания. Действительное направление использования индекса выбираете не вы, а оптимизатор при создании и принятии планов доступа.

В следующем условии WHERE для установки границ интервала просмотра индекса будут использоваться только предикаты для NAME и DEPT, но не для SALARY или YEARS:

```
    WHERE  NAME  = :hv1
AND   DEPT  = :hv2
 AND  SALARY = :hv4
 AND  YEARS  = :hv5
```

Это происходит потому, что в ключе индекса есть MGR, отделяющий эти столбцы от первых двух столбцов ключа, поэтому порядок нарушен. Однако когда диапазон определен предикатами NAME = :hv1 и DEPT = :hv2, остальные предикаты можно сравнивать с остальными столбцами в ключе индекса.

Помимо описанных выше предикатов равенства для установки границ интервала просмотра индекса можно использовать определенные предикаты неравенства. Мы обсудим два типа предикатов неравенства: предикаты строгого неравенства и предикаты допустимого неравенства.

Предикаты строгого неравенства: Для ограничения интервала могут использоваться операторы строгого неравенства > и <.

При ограничении интервала просмотра индекса будет приниматься во внимание только один столбец с предикатами строгого неравенства. В следующем

примере предикаты для столбцов NAME и DEPT могут использоваться для указания границ интервала, а предикат столбца MGR - нет.

```
 WHERE  NAME  = :hv1
AND  DEPT   > :hv2
AND  DEPT   < :hv3
AND  MGR    < :hv4
```

Предикаты нестрого неравенства: Следующие предикаты нестрогого неравенства могут использоваться для ограничения интервала:

- >= и <=
- BETWEEN
- LIKE

При ограничении интервала просмотра индекса будут приниматься во внимание несколько столбцов с предикатами допустимого неравенства. В следующем примере для указания границ интервала просмотра индекса могут использоваться все предикаты:

```
 WHERE  NAME  = :hv1
AND  DEPT  >= :hv2
AND  DEPT  <= :hv3
AND  MGR   <= :hv4
```

В продолжение иллюстрации этого примера предположим, что `:hv2 = 404`, `:hv3 = 406`, а `:hv4 = 12345`. Менеджер баз данных полностью просмотрит индекс для подразделений 404 и 405, но прекратит просматривать подразделение 406, когда достигнет первой записи с номером сотрудника (столбец MGR) больше 12345.

Более подробную информацию смотрите в разделе “Предикаты ограничения интервала и предикаты с аргументом поиска индекса” на стр. 180.

Просмотр индекса для упорядочения данных

Если запрос включает упорядочение, для упорядочения данных можно использовать индекс, если столбцы упорядочения появляются в индексе последовательно, начиная с первого столбца индексных ключей. (Упорядочение или сортировка может быть результатом таких операций, как ORDER BY, DISTINCT, GROUP BY, подзапрос “= ANY”, подзапрос“> ALL”, подзапрос “ < ALL”, INTERSECT или EXCEPT, UNION.) Исключением является случай проверки столбцов индексных ключей на равенство со “значениями констант” (то есть любых выражений, приравниваемых к константам). В этом случае столбец упорядочения может быть иным, чем первые столбцы индексных ключей. Например, в запросе:

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY MGR
```

индекс может использоваться для упорядочения строк, так как NAME и DEPT будут всегда иметь одни и те же значения и тем самым упорядоченными. Иначе говоря, приведенные выше условия WHERE и ORDER BY эквивалентны следующим:

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY NAME, DEPT, MGR
```

Индекс уникальности может также использоваться для усечения требования упорядочения. Например, если даны следующие определение индекса и условие ORDER BY:

```
UNIQUE INDEX IX0:  PROJNO  ASC
SELECT PROJNO, PROJNAME, DEPTNO
   FROM PROJECT
ORDER BY PROJNO, PROJNAME
```

дополнительное упорядочение столбца PROJNAME не требуется, поскольку индекс IX0 гарантирует уникальность PROJNO. Эта уникальность означает, что для каждого значения PROJNO существует единственное значение PROJNAME.

Доступ только к индексу

В некоторых случаях все требуемые данные можно получить из индекса, не обращаясь к таблице. Это называется *доступом только к индексу*.

Чтобы проиллюстрировать доступ только к индексу, рассмотрим следующее определение индекса:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

тогда для следующего запроса можно обращаться только к индексу, без чтения базовой таблицы:

```
SELECT NAME, DEPT, MGR, SALARY
     FROM EMPLOYEE
  WHERE NAME = 'SMITH'
```

В других запросах могут встречаться столбцы, которых нет в индексе. Для получения данных этих столбцов надо читать строки базовой таблицы. Например, при заданном индексе IX1 для следующего запроса требуется обращаться к базовой таблице, чтобы получить данные столбцов PHONENO и HIREDATE:

```
 SELECT NAME, DEPT, MGR, SALARY, PHONENO, HIREDATE
     FROM EMPLOYEE
  WHERE NAME = 'SMITH'
```

Создавая индекс уникальности с включением столбцов, можно улучшить производительность получения данных, увеличив число обращений только к индексу.

Чтобы проиллюстрировать использования столбцов включения, рассмотрим следующее определение индекса:

```
CREATE UNIQUE INDEX IX1 ON EMPLOYEE
   (NAME ASC)
    INCLUDE (DEPT, MGR, SALARY, YEARS)
```

Это создает индекс уникальности, обеспечивающий уникальность столбца NAME, а кроме того, хранящий и поддерживающий данные столбцов DEPT, MGR, SALARY и YEARS.

Для следующего запроса можно обращаться только к индексу, без чтения базовой таблицы:

```
SELECT NAME, DEPT, MGR, SALARY
     FROM EMPLOYEE
   WHERE NAME='SMITH'
```

Доступ к нескольким индексам

Во всех вышеприведенных примерах для получения результата выполнялся просмотр одного индекса. Чтобы удовлетворить предикатам условия WHERE, оптимизатор может выбрать просмотр нескольких индексов. Например, если есть два определения индексов:

```
INDEX IX2:  DEPT    ASC
INDEX IX3:  JOB     ASC,
            YEARS   ASC
```

обращения к этим двум индексам, позволяют обработать следующие предикаты:

```
WHERE DEPT = :hv1
  OR (JOB   = :hv2
  AND YEARS >= :hv3)
```

В этом примере просмотр индекса IX2 выдаст список ID строк (RID), удовлетворяющий предикату `DEPT = :hv1`. Просмотр индекса IX3 выдаст список RID, удовлетворяющий предикату `JOB = :hv2 AND YEARS >= :hv3`. Перед обращением к таблице эти два списка RID можно объединить, а повторения удалить. Это называют операцией *OR над индексами*.

Операцию OR над индексами можно использовать также для предикатов с выражением IN, как в следующем примере:

```
 WHERE DEPT IN (:hv1, :hv2, :hv3)
```

Задача операции OR над индексами - удалить повторения RID; задача же *операции AND над индексами* - найти общие RID. Операция AND над индексами

может встречаться при работе с прикладными программами, где есть несколько индексов по разным столбцам одной и той же таблицы, и запрос с несколькими предикатами, соединенными операцией AND, выполняется на такой таблице. Просмотр нескольких индексов для каждого индексированного столбца при таком запросе выдает значения, хешируемые для создания битовых множеств. Второе битовое множество используется, чтобы проверить первое битовое множество для генерации нужных строк, вызываемых для создания конечного возвращаемого набора данных.

Например, если есть два определения индексов:

```
 INDEX IX4: SALARY   ASC
INDEX IX5: COMM     ASC
```

обращения к этим двум индексам, позволяют обработать следующие предикаты:

```
WHERE SALARY BETWEEN 20000 AND 30000
   AND COMM BETWEEN 1000 AND 3000
```

В этом примере просмотр индекса `IX4` дает битовое множество, удовлетворяющее предикату `SALARY BETWEEN 20000 AND 30000`. Просмотр `IX5` и проверка битового множества для `IX4` дает список нужных RID, удовлетворяющий обоим предикатам. Это называется “динамической операцией AND над битовыми множествами”. Это происходит, только если таблица имеет достаточную мощность, а столбцы - достаточно значений в нужном интервале или достаточно повторений, если используются предикаты равенства.

Чтобы реализовать возможности улучшения производительности от использования динамических битовых образов при просмотре нескольких индексов, может понадобиться изменение значений параметра конфигурации базы данных *sortheap* (размер кучи сортировки) и параметра конфигурации менеджера баз данных *sheapthres* (порог кучи сортировки).

Если в плане доступа используются динамические битовые образы, требуется дополнительный объем кучи сортировки. Если заданное значение *sheapthres* относительно близко к значению *sortheap* (то есть для одновременных запросов они различаются менее чем в два или три раза), для работы с динамическими битовыми образами будет использоваться гораздо меньше памяти, чем предполагал оптимизатор.

Чтобы исправить эту ситуацию, увеличьте значение *sheapthres* по отношению к значению *sortheap*.

Примечание: При доступе к любой отдельной таблице DB2 не объединяет операции AND и OR над индексами.

Кластеризованные индексы

При выборе плана доступа оптимизатор рассматривает затраты ввода-вывода на выборку страниц с диска в пул буферов. В процессе своей работы оптимизатор оценивает число операций ввода-вывода, требуемых для выполнения запроса. Эта оценка включает предсказание использования пула буферов, поскольку для чтения строк уже находящейся в буферном пуле страницы не требуются дополнительные операции ввода-вывода.

Для просмотра индекса оптимизатор использует информацию из таблиц системного каталога (SYSCAT.INDEXES), чтобы помочь оценить затраты ввода-вывода при считывании страниц данных в буферный пул. Используются следующие столбцы таблицы SYSCAT.INDEXES:

- CLUSTERRATIO, указывающий степень кластеризации табличных данных по отношению к этому индексу. Большое число означает, что на страницах данных в последовательности индексного ключа строки упорядочены. Следовательно, все строки страницы данных можно считывать, пока страница находится в буфере. Если в этом столбце находится значение -1, оптимизатор будет пытаться использовать PAGE_FETCH_PAIRS и CLUSTERFACTOR.

  или

- PAGE_FETCH_PAIRS, содержащие несколько пар чисел, моделирующих число операций ввода-вывода, требуемых для считывания страниц данных в буферные пулы различного размера вместе с CLUSTERFACTOR. При сборе статистики индекса эта информация входит в состав подробной статистики.

Если данные статистики недоступны, оптимизатор использует для статистики значения по умолчанию, что подразумевает низкую кластеризацию данных в индексе. Смотрите также разделы “Глава 5. Статистика системного каталога” на стр. 117 и “Сбор статистики с помощью утилиты RUNSTATS” на стр. 119.

Можно указать индекс кластеризации, который будет использоваться и для кластеризации строк при реорганизации таблицы, и для сохранения этой характеристики при обработке вставки. (Информацию о реорганизации таблиц смотрите в разделе “Реорганизация каталогов и пользовательских таблиц” на стр. 281.) Последующие обновления и вставки могут сделать индекс менее кластеризованным (по результатам статистики, собранной RUNSTATS), поэтому может требоваться периодическая реорганизация таблицы. Чтобы уменьшить частоту реорганизаций таблицы, данные в которой часто изменяются с помощью операторов INSERT, UPDATE и DELETE, используйте при изменении определения таблицы параметр PCTFREE. Это позволит кластеризовать дополнительные вставки в имеющиеся данные.

Степень кластеризации данных по отношению к индексу может оказать значительное воздействие на производительность, и нужно стараться, чтобы один из индексов таблицы был кластеризован почти на 100 процентов.

Как правило, только один из индексов может быть кластеризован на 100 процентов; исключение составляют случаи, когда ключи являются надмножеством ключей индекса кластеризации, или когда фактически существует корреляция между ключевыми столбцами этих двух индексов.

Дополнительную информацию о влиянии использования кластеризации индексов на производительность смотрите в разделе “Управление индексами: советы по улучшению производительности” на стр. 107. Как создавать кластерные индексы, смотрите в описании оператора CREATE INDEX справочника *SQL Reference*.

Чтение кластерных страниц с применением предварительной выборки списка: Если оптимизатор использует индекс для доступа к строкам, он может отложить чтение страниц данных до получения из индекса всех идентификаторов строк (RID). Пусть, например, определен индекс IX1:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

и заданы следующие критерии поиска:

```
WHERE NAME BETWEEN 'A' and 'I'
```

Тогда оптимизатор может выполнить просмотр индекса на IX1 для определения требуемых строк (и страниц данных). Если данные не кластеризованы в соответствии с этим индексом, предварительная выборка списка будет включать шаг сортировки списка RID, полученного при просмотре индекса. Дополнительную информацию смотрите в разделе “Как работает предварительная выборка списка” на стр. 272.

Предварительная выборка страниц индекса

Если это возможно, менеджер баз данных обнаруживает последовательный доступ к страницам индекса и генерирует запросы на предварительную выборку. Это должно значительно сократить время, затрачиваемое на неселективный просмотр индекса или на селективный просмотр индекса, охватывающий значительную часть индекса.

Для оценки числа подлежащих предварительной выборке индексных страниц оптимизатор использует статистические показатели индексов (такие как DENSITY и SEQUENTIAL_PAGES), характеристики содержащих индекс табличных пространств и действие любых предикатов ограничения. Эти оценки дают суммарную оценку затрат использования отдельно взятого индекса.

Дополнительную информацию смотрите в разделе “Анализ последовательной предварительной выборки” на стр. 270.

Сравнение реляционного просмотра с просмотром индекса

Если для запроса нельзя использовать индекс, или если оптимизатор определяет, что просмотр индекса потребует больших затрат, оптимизатор выберет реляционный просмотр. Просмотр индекса может потребовать больших затрат, если:

- Таблица мала
- Кластеризация индекса низка
- Затрагивается большая часть таблицы.

Чтобы определить, использует ваш план доступа реляционный просмотр или просмотр индекса, можно воспользоваться возможностями объяснения SQL. Смотрите раздел “Глава 7. Возможность объяснения SQL” на стр. 225.

Терминология предикатов

Программа пользователя запрашивает набор строк из базы данных с помощью оператора SQL, задавая нужные строки путем использования предикатов. Когда оптимизатор принимает решение, как обрабатывать оператор SQL, каждый предикат попадает в одну из четырех категорий. Категория определяется тем, как и когда данный предикат используется в процессе оценки. Эти категории перечисляются ниже в порядке убывания производительности:

1. Предикаты ограничения интервала
2. Предикаты с аргументом поиска индекса
3. Предикаты с аргументом поиска данных
4. Остаточные предикаты.

С аргументом поиска означает, что в предикате есть нечто, что можно использовать как аргумент поиска.

В “Сводка использования предикатов” на стр. 181 приведена сравнительная оценка характеристик, влияющих на производительность различных категорий предикатов.

Предикаты ограничения интервала и предикаты с аргументом поиска индекса

Предикаты ограничения интервала используются для ограничения области просмотра индекса. Они содержат ключевые значения начала и конца поиска в индексе. Предикаты с аргументом поиска индекса не используются для ограничения области поиска, но их можно оценить из индекса, потому что включенные в предикат столбцы являются частью индексного ключа. Например, если даны определенный ранее (в разделе “Концепции просмотра индекса” на стр. 170) индекс IX1 и следующее условие WHERE:

```
WHERE  NAME  = :hv1
 AND   DEPT  = :hv2
 AND   YEARS > :hv5
```

первые два предиката (NAME = :hv1, DEPT = :hv2) - предикаты ограничения интервала, а YEARS > :hv5 - предикат с аргументом поиска индекса.

Менеджер баз данных выберет использование при обработке этих предикатов данных индекса, а не чтение из базовой таблицы. Такие *предикаты с аргументом поиска индекса* уменьшают число читаемых страниц данных, сокращая набор строк, подлежащих чтению из таблицы. Они не влияют на число читаемых индексных страниц.

Предикаты с аргументом поиска данных

Предикаты, которые не могут быть обработаны менеджером индексов, но могут быть обработаны службой управления данными называются *предикатами с аргументом поиска данных*. Обычно они требуют доступа к отдельным строкам базовой таблицы. Если требуется, службы управления данными читают необходимые для оценки предиката столбцы, как и любые другие, необходимые для обработки столбцов списка SELECT, которые нельзя получить из индекса.

Пусть, например, есть одиночный предикат, определенный на таблице PROJECT:

```
INDEX IX0:  PROJNO ASC
```

В следующем запросе предикат `DEPTNO = 'D11'` считается предикатом с аргументом поиска данных.

```
SELECT PROJNO, PROJNAME, RESPEMP
  FROM PROJECT
 WHERE DEPTNO = 'D11'
 ORDER BY PROJNO
```

Остаточные предикаты

Как правило, остаточными являются предикаты, требующие операций ввода/вывода помимо простого обращения к базовой таблице. Примерами остаточных предикатов являются предикаты, использующие коррелирующие подзапросы, множественные подзапросы (подзапросы с ANY, ALL, SOME или IN) или считывающие данные типа LONG VARCHAR или LOB (хранимые в отдельном от таблицы файле). Эти предикаты обрабатываются службами реляционных данных.

Иногда предикаты, применяемые только к индексу, должны быть применены повторно при обращении к странице данных. Например, планы доступа, использующие операции OR или AND над индексами (смотрите раздел “Доступ к нескольким индексам” на стр. 176), всегда повторно применяют предикаты в качестве остаточных, когда получен доступ к странице данных.

Сводка использования предикатов

Использование предикатов в запросе может помочь уменьшить количество считываемых для обработки запроса данных. Разные категории предикатов оказывают разное воздействие на производительность запроса, причем эти воздействия учитываются оптимизатором. Ниже в таблице показаны

ранжирование разных типов предикатов по степени влияния каждого типа предиката на производительность.

Таблица 14. Сводные характеристики типов предикатов

Характеристика	Тип предиката			
	Ограничения интервала	**С аргументом поиска индекса**	**С аргументом поиска данных**	**Остаточные**
Уменьшение ввода/вывода для индекса	Да	Нет	Нет	Нет
Уменьшение ввода/вывода страниц данных	Да	Да	Нет	Нет
Уменьшение числа внутренней передаваемых строк	Да	Да	Да	Нет
Уменьшение числа отобранных строк	Да	Да	Да	Да

Концепции объединения

Объединение означает, что строки таблицы присоединяются к строкам одной или нескольких таблиц. Например, даны две таблицы:

```
     TABLE1                    TABLE2
-----------------         -----------------
 PROJ     PROJ_ID         PROJ_ID     NAME
------    -------         -------    ------
  A          1              1        Sam
  B          2              3        Joe
  C          3              4        Mary
  D          4              1        Sue
                            2        Mike
```

Объединение первой и второй таблиц по совпадающим ID столбцов можно представить следующим оператором SQL:

```
SELECT PROJ, x.PROJ_ID, NAME
 FROM TABLE1 x, TABLE2 y
 WHERE x.PROJ_ID = y.PROJ_ID
```

Его результатом будет следующий набор строк:

```
 PROJ     PROJ_ID     NAME
------    -------    ------
  A          1        Sam
```

```
A          1          Sue
B          2          Mike
C          3          Joe
D          4          Mary
```

При объединении двух таблиц одна выбирается в качестве внешней, а другая - в качестве внутренней. Сначала производится обращение к внешней таблице; она просматривается только один раз. Будет ли внутренняя таблица просматриваться несколько раз, зависит от типа объединения и того, какие индексы построены. Независимо от количества объединяемых при запросе таблиц оптимизатор объединяет по две таблицы за раз. Если нужно, создаются временные таблицы промежуточных результатов.

Оптимизатор выберет один из двух методов объединения (объединение с вложенным циклом или объединение слияния) в зависимости от существования предиката объединения (определенного в разделе “Объединение слияния” на стр. 184), и от оценок затрат, определяемых по статистическим показателям таблиц и индексов.

Объединение со вложенным циклом

Объединение со вложенным циклом выполняется одним из двух способов:

1. Просмотр внутренней таблицы для каждой рассматриваемой строки внешней таблицы.

   Пусть, например, столбец A в таблицах T1 и T2 содержит следующие значения:

   ```
   Внешняя таблица T1: столбец A      Внутренняя таблица T2: столбец A
       ------------------------      ------------------------
                  2                          3
                 3                                 2
                 3                                 2
                                                   3
                                            1
   ```

   Этапы выполнения вложенного цикла:

   - Чтение первой строки T1. Значение A равно “2”
   - Просмотр T2, пока не будет найдено соответствие (“2”), с последующим объединением двух строк
   - Просмотр T2, пока не будет найдено очередное соответствие (“2”), с последующим объединением двух строк
   - Просмотр T2 до конца таблицы
   - Возврат к T1 и чтение следующей строки (“3”)
   - Просмотр T2, начиная с первой строки, пока не будет найдено соответствие (“3”), с последующим объединением двух строк
   - Просмотр T2, пока не будет найдено очередное соответствие (“3”), с последующим объединением двух строк

- Просмотр T2 до конца таблицы
- Возврат к T1 и чтение следующей строки (“3”)
- Просмотр T2, как указано раньше, с объединением всех строк с соответствием (“3”).

2. Поиск по индексу во внутренней таблице для каждой рассматриваемой строки внешней таблицы.

   Этот метод может использоваться для указанных предикатов, если существует предикат следующей формы:

   ```
   выражение(внешняя_таблица.столбец)  операция  внутренняя_таблица.столбец
   ```

   где операция - операция отношения (например, =, >, >=, <, or <=), а выражение - допустимое выражение на внешней таблице. Примеры:

   ```
   OUTER.C1 + OUTER.C2 <= INNER.C1
   ```

   и

   ```
   OUTER.C4 < INNER.C3
   ```

   Этим методом можно значительно сократить число читаемых строк внутренней таблицы для каждой строки внешней таблицы (хотя это зависит от ряда факторов, включая селективность предиката объединения).

При оценке объединения с вложенным циклом оптимизатор также должен определить, надо ли сортировать внешнюю таблицу перед выполнением объединения. Упорядочением внешней таблицы на основе столбцов объединения можно сократить число операций чтения с диска страниц, поскольку они с большой вероятностью они уже будут находиться в пуле буферов. Если для доступа к внутренней таблице объединения используется высококластеризованный индекс, сортировкой внешней таблицы можно минимизировать число читаемых страниц индекса.

Кроме того, оптимизатор может предпочесть выполнить сортировку до объединения, если ожидается, что сортировка после объединения повлечет большие затраты. Последующая сортировка может требоваться для обработки условий GROUP BY, DISTINCT, ORDER BY или объединения слияния.

Объединение слияния

Объединение слиянием (другие названия - объединение просмотром слияния или сортировка объединением слиянием) требует предиката в форме таблица1.столбец = таблица2.столбец. Этот предикат называется *объединяющим предикатом равенства*. Объединение слиянием требует упорядоченного ввода для объединяемых столбцов - либо через доступ по индексу, либо путем сортировки. При использовании объединения слиянием столбец объединения не может быть столбцом длинного поля (LONG) или столбцом большого объекта (LOB).

Объединяемые таблицы просматриваются одновременно. Внешняя таблица при объединении слиянием просматривается только один раз. Внутренняя таблица тоже просматривается один раз, если во внешней таблице нет повторяющихся значений. Если во внешней таблице есть повторяющиеся значения, группа строк внутренней таблицы может просматриваться повторно. Пусть, например, столбец A в таблицах T1 и T2 содержит следующие значения:

```
Внешняя таблица T1: столбец A      Внутренняя таблица T2: столбец A
-----------------------      -----------------------
          2                                  1
          3                                  2
          3                                  2
                                             3
                                             3
```

Этапы объединения слиянием:

- Чтение первой строки T1. Значение A равно “2”
- Просмотр T2, пока не будет найдено соответствие, с последующим объединением двух строк
- Продолжение просмотра T2, пока столбцы соответствуют, объединение строк.
- Когда прочитано “3” в T2, возврат к T1 и чтение следующей строки
- Следующее значение T1 равно “3”, что соответствует T2, поэтому строки объединяются
- Продолжение просмотра T2 пока столбцы соответствуют, объединение строк.
- Достигнут конец T2
- Возврат к T1 для получения следующей строки – заметим, что следующее значение в T1 совпадает с предыдущим, поэтому T2 просматривается снова, начиная с первой “3” в T2 (менеджер баз данных запоминает эту позицию).

Хеш-объединение

Для хеш-объединения требуются один или несколько предикатов вида таблица1.столбецX = таблица2.столбецY, для которых типы столбцов **одинаковы**. Длина столбцов типа CHAR должна быть одинакова. Для столбцов типа DECIMAL должны быть одинаковы точность и масштаб. Столбец не может иметь тип длинное поле или большой объект.

Вначале просматривается одна таблица (называемая внутренней таблицей), и строки копируются в буферы памяти, выделяемые из кучи сортировки (смотрите описание параметра конфигурации базы данных “Размер кучи сортировки (sortheap)” на стр. 381). Буферы памяти делятся на разделы на основе “хеш-кода”, вычисляемого по значениям столбцов предикатов объединения. Если размер первой таблицы превышает доступное пространство кучи сортировки, буферы из выбранных разделов записываются во временные таблицы. По завершении обработки внутренней таблицы просматривается вторая таблица (называемая внешней таблицей). Для строк внешней таблицы

производится поиск соответствующих строк внутренней таблицы: по значения в столбцах предикатов объединения вычисляется “хеш-код”. Для строки с определенным “хеш-кодом” из внешней таблицы выбираются строки внутренней таблицы с соответствующим “хеш-кодом”, и с ними производится сравнение самих столбцов предикатов объединения.

Если строка внешней таблицы соответствует разделу, не записанному во временную таблицу, ее сравнение со строками внутренней таблицы производится в памяти немедленно. Если же соответствующий внутренней таблицы INNER был записан во временную таблицу, строка внешней таблицы также записывается во временную таблицу. Наконец, соответствующие пары разделов считываются из временных таблиц, “хеш-коды” их строк сравниваются, и проверяются предикаты объединения.

Для реализации преимуществ производительности хеш-объединения может потребоваться увеличение параметра *sortheap* конфигурации базы данных и параметра *sheapthres* конфигурации менеджера базы данных.

Для запросов поддержки решений планы доступа для хеш-объединения используют больше пространства кучи сортировки, чем планы для других методов объединения. Когда значение *sheapthres* относительно близко к значению *sortheap* (то есть их отношение меньше, чем два или три к одному), для выполнения хеш-объединения отводится гораздо меньше памяти, чем предполагал оптимизатор. Хеш-объединение при недостатке памяти может выполняться очень медленно. Эта проблема возникает при запросах со многими сортировками и хеш-объединениями, когда сортировки и хеш-объединения занимают большую часть доступной памяти.

Решение состоит в таком задании *sheapthres*, чтобы этот параметр был достаточно велик (по сравнению с *sortheap*).

Выбор внешней и внутренней таблицы

Как при объединении определяются внутренняя и внешняя таблицы? Ниже приводятся общие правила того, как оптимизатор решает, какая таблица будет внешней, а какая внутренней.

При **хеш-объединении** внутренняя таблица помещается в буферы памяти. Если буферов памяти недостаточно, хеш-объединение потребует построения временных таблиц. Оптимизатор пытается избежать этого, принимая меньшую из таблиц за внутреннюю, а большую - за внешнюю.

Порядок доступа к таблицам важен, в частности, для **объединения с вложенным циклом**, потому что доступ к внешней таблице происходит один раз, а к внутренней - по одному разу на каждую строку внешней таблицы. Оптимизатор выбирает внутреннюю и внешнюю таблицы на основе оценки затрат. На эти оценки влияют следующие факторы:

- Размер

  Для сокращения числа повторных обращений к внутренней таблице меньшая таблица часто принимается за внешнюю. Однако при предварительной выборке часто выгоднее обратное. Предварительная выборка может значительно уменьшить затраты на доступ к большой таблице. Однако обычно предварительная выборка эффективна только для внешней таблицы объединения. Следовательно, первым может осуществляться доступ к большей таблице. Дополнительную информацию смотрите в разделе “Предварительная выборка данных в пул буферов” на стр. 269.

- Предикаты

  Таблица с большей вероятностью будет выбрана внешней, если к ней можно применить предикаты отбора, поскольку обращение происходит только к тем строкам внутренней таблицы, которые удовлетворяют предикатам, применяемым к внешней таблице.

- Буферизация

  Если для каждой строки внешней таблицы надо просмотреть всю внутреннюю таблицу (то есть для внутренней таблицы нельзя использовать индекс), меньшая из двух таблиц может быть выбрана как внутренняя для использования преимуществ буферизации. Это будет зависеть от размера таблиц и размера пула буферов. Заметим, что поскольку решения об объединении зависят от размера пула буферов, план доступа для ваших программ может меняться при их повторном связывании с базой данных после изменения размера пула буферов.

  Возможность создания нескольких пулов буферов, изменения размера пула буферов и управления табличными пространствами, использующими этот пул буферов может оказать влияние при использовании буферизации для внутренней и внешней таблиц.

- Индексы

  Если для одной из таблиц можно использовать просмотр индекса, эта таблица - хороший кандидат на роль внутренней таблицы. Для доступа к ней можно просматривать индексные ключей, используя предикат объединения ключей внешней таблицы в качестве одного из ключевых значений. Если у таблицы нет индекса, она - не лучший кандидат на роль внутренней таблицы, поскольку в этом случае для каждой строки внешней таблицы потребуется полный просмотр внутренней таблицы.

- Требования к порядку

  К таблице, по которой определяется требуемый порядок, доступ должен осуществляться в первую очередь. Например, если итог объединения t1 и t2 должен быть упорядочен по t1.c, хорошим выбором может быть доступ к t1 как внешней с индексом по t1.c. Итог объединения будет упорядочен, и сортировка не потребуется.

```
SELECT * FROM t1, t2
  WHERE t1.a = t2.b
  ORDER BY t1.c
```

Порядок доступа к таблицам менее важен для **объединения слиянием**, потому что и внутренняя, и внешняя таблицы считываются однократно. Однако части внутренней таблицы, соответствующие повторяющимся значениям объединения во внешней, хранятся в буфере памяти. Буфер считывается повторно, если следующая внешняя строка совпадает с предыдущей, если же нет - буфер отбрасывается. Если число повторяющихся значений объединения превышает емкость буфера в памяти, хранятся не все дубликаты. Это случится, только если некоторое значение повторяется многократно, и для этого значения есть соответствие во внешней таблице.

Все соображения о повторении значений в большинстве случаев приводят к тому, что таблица с меньшим числом повторений будет выбрана в качестве внешней таблицы объединения. В конечном счете, однако, оптимизатор выберет внешнюю и внутреннюю таблицы на основе детальной оценки затрат.

Стратегии поиска для выбора оптимального объединения

Оптимизатор может определить оптимальные методы объединения, используя разные стратегии поиска. Выбор стратегии поиска определяется используемым классом оптимизации (смотрите раздел “Настройка класса оптимизации” на стр. 70). Стратегии поиска и их характеристики:

- ″Жадный″ перебор методов объединения
  - Эффективно по отношению к пространству и времени
  - Однонаправленный перебор (выбранный метод объединения двух таблиц не меняется при дальнейшей оптимизации)
  - Может пропустить лучший план доступа при объединении многих таблиц. Если ваш запрос объединяет две-три таблицы, план доступа, выбранный ″жадным″ перебором, будет тем же, что и выбранный динамически программируемым перебором методов объединения. Это верно, в частности, при запросе, содержащем много предикатов объединения (явно заданных или неявно полученных применением транзитивности предикатов) по одному столбцу.
- Динамически программируемый перебор методов перечисления
  - Требования к пространству и времени растут экспоненциально при увеличении числа объединяемых таблиц.
  - Эффективный и исчерпывающий поиск наилучшего плана доступа
  - Подобен стратегии, используемой DB2 для OS/390.

Алгоритм перебора методов объединения - ключевой определитель числа вариантов плана, исследуемых оптимизатором.

Стратегии поиска для объединения типа ″звезда″

Как правило, таблицы, на которые ссылается запрос, должны быть соединены предикатами объединения. Если две таблицы объединяются без предиката объединения, образуется декартово произведение двух таблиц. Это значит, что

каждая заданная строка первой таблицы объединяется с каждой заданной строкой второй таблицы, создавая итоговую таблицу с размером, равным произведению размеров двух таблиц (то есть, как правило, таблицу очень большого размера). Поскольку маловероятно, что такой план будет выполняться очень хорошо, оптимизатор даже не будет пытаться определить затраты такого плана доступа. Единственное исключение - когда задан класс оптимизации 9 или имеет место особый случай “схем типа ″звезда″”. Дополнительную информацию смотрите в разделе “Настройка класса оптимизации” на стр. 70.

Планы доступа, включающие декартовы произведения, выполняются хорошо обычно в случае больших баз данных для поддержи решений, построенных с применением схемы типа ″звезда″. Схема ″звезда″ - это план базы данных, в котором основная масса необработанных данных держится в одной большой таблице с множеством столбцов, обычно называемой “таблицей фактов”. Многие столбцы содержат кодированные значения, характеризующие массивы отдельных данных, хранящихся в таблице фактов. Чтобы облегчить анализ некоторых поднаборов фактов, для декодирования кодированных значений используются таблицы ассоциаций. Типичный запрос должен состоять из множества локальных предикатов, ссылающихся на декодированные значения в таблицах ассоциаций, и содержать предикаты объединения, связывающие таблицы ассоциаций с таблицей фактов. Для такого сорта запросов может оказаться выгодным вычислить декартово произведение множества малых таблиц ассоциаций до получения доступа к большой таблице фактов. Этот метод дает преимущества, когда многочисленные предикаты объединения соответствуют многостолбцовому индексу.

DB2 умеет распознавать запросы к базам данных, сконструированным по схеме ″звезда″ с как минимум двумя таблицами ассоциаций, и увеличивать сферу поиска для включения потенциальных планов с декартовыми произведениями таблиц ассоциаций. Если план, включающий декартовы произведения, дает наименьшую оценку затрат, он будет выбран оптимизатором.

Обсуждаемая выше техника схемы ″звезда″ предполагает использование в объединении индексов первичных ключей. Другой сценарий может включать индексы внешних ключей. Если известно, что столбцы внешних ключей в таблице фактов соответствуют одностолбцовыми индексами, и что всем таблицам ассоциаций присуща сравнительно высокая селективность, может использоваться следующая техника объединения типа ″звезда″:

1. Над каждой таблицей ассоциаций:
   - Выполняется полуобъединение между таблицей ассоциаций и индексом внешнего ключа в таблице фактов
   - Значения ID строки (RID) хешируются для динамического создания битового множества.

2. Каждое битовое множество логически умножается (предикат AND) на предыдущее битовое множество (смотрите раздел “Доступ к нескольким индексам” на стр. 176).
3. После обработки последнего битового множества определяются оставшиеся RID.
4. Эти RID сортируются (не обязательно).
5. Производится выборка строки базовой таблицы.
6. Проводится повторное объединение таблицы фактов с каждой из ее таблиц ассоциаций, с обращением к необходимым по условию SELECT столбцам таблиц ассоциаций.
7. Повторно применяются предикаты (остаточные предикаты).

При использовании данного метода не требуются многостолбцовые индексы. Налагаемые ограничения на целостность ссылок между таблицей фактов и таблицами ассоциаций не требуются для выбора этого метода, хотя взаимосвязь между таблицей фактов и таблицами ассоциаций должна обладать этим свойством.

Для динамических битовых образов, создаваемых и используемых в технологии объединения типа ″звезда″, используется память кучи сортировки. Дополнительную информацию о параметре конфигурации базы данных *sortheap* (размер кучи сортировки) смотрите в главе 13, ″Конфигурирование DB2″ книги *Administration Guide: Performance*.

Составные таблицы

Еще один важный параметр определяет вид последовательности объединений в запросе. Результат объединения пары таблиц - новая таблица, называемая составной таблицей. Часто полученная составная таблица становится внешней таблицей объединения с еще одной внутренней таблицей. Ее называют “составной внешней” таблицей. В некоторых ситуациях, в частности, при использовании метода ″жадного″ перебора методов объединения, полезно сделать результат объединения двух таблиц внутренней таблицей последующего объединения. Когда внутренняя таблица объединения представляет результат объединения двух или нескольких таблиц, говорят, что план содержит “составную внутреннюю” таблицу. Например, в запросе:

```
SELECT COUNT(*)
FROM T1, T2, T3, T4
WHERE T1.A = T2.A AND
      T3.A = T4.A AND
      T2.Z = T3.Z
```

может оказаться выгодным объединить таблицы T1 и T2 (T1xT2), затем объединить T3 и T4 (T3xT4) и, наконец, выбрать результат первого объединения в качестве внешней, а второго - в качестве внутренней таблицы. В итоговом плане ((T1xT2) x (T3xT4)) результат объединения (T3xT4) будет

составной внутренней таблицей. В зависимости от класса оптимизации запроса оптимизатор налагает различные ограничения на максимальное число таблиц, которые могут быть внутренними таблицами объединения. Составные внутренние таблицы разрешаются при классах оптимизации 5, 7 и 9.

Реплицируемые сводные таблицы

Используя реплицируемые таблицы сводок в среде многораздельных баз данных, можно улучшить производительность, используя предварительно вычисленные значения для данных базовой таблицы. Например, следующий запрос выиграет от создания нижеприведенной реплицируемой таблицы сводок. Мы предполагаем, что:

- Таблица SALES находится в многораздельном табличном пространстве REGIONTABLESPACE и разделена по столбцу REGION.
- Таблицы EMPLOYEE и DEPARTMENT находятся в одноразделной группе узлов.

Вы создаете реплицируемую таблицу сводок на основе информации таблицы EMPLOYEE.

```
CREATE TABLE R_EMPLOYEE
  AS (
      SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT
      FROM EMPLOYEE
     )
DATA INITIALLY DEFERRED REFRESH IMMEDIATE
IN REGIONTABLESPACE
REPLICATED;
```

Содержимое созданной реплицируемой таблицы сводок обновляется при выполнении оператора:

```
REFRESH TABLE R_EMPLOYEE;
```

В следующем примере вычисляются объемы продаж для каждого сотрудника, подытоги по подразделениям и общий итог:

```
SELECT d.mgrno, e.empno, SUM(s.sales)
FROM   department AS d, employee AS e, sales AS s
WHERE  s.sales_person = e.lastname
   AND e.workdept = d.deptno
GROUP BY ROLLUP(d.mgrno, e.empno)
ORDER BY d.mgrno, e.empno;
```

Вместо использования таблицы EMPLOYEE, существующей только на одном разделе базы данных, менеджер баз данных использует таблицу R_EMPLOYEE, реплицированную на каждом из разделов, где есть таблица SALES. Производительность увеличивается, потому что для вычисления объединения информацию о сотрудниках не надо передавать по сети каждому разделу.

Реплицируемые сводные таблицы могут использоваться для обеспечения локального выполнения объединений. Например, если есть схема типа ″звезда″, в которой большая таблица фактов распределена по двенадцати узлам, объединения между таблицей фактов и таблицами ассоциаций будут более эффективны, если эти таблицы размещены совместно.

Если все эти таблицы размещены в одной группе узлов, для локально выполняемого объединения корректно может быть распределена только одна таблица ассоциации. Все другие таблицы ассоциаций нельзя будет использовать в локально выполняемом объединении, так как столбцы объединения таблицы фактов не будут соответствовать ключу разделения таблицы фактов.

Например, вы можете иметь таблицу фактов FACT (C1, C2, C3, ...), для разделения которой используется столбец C1, таблицу ассоциации DIM1 (C1, dim1a, dim1b, ...), для разделения которой используется столбец C1, таблицу ассоциации DIM2 (C2, dim2a, dim2b, ...), для разделения которой используется столбец C2, и т.д.

В этом примере можно увидеть, что объединение между таблицами FACT и DIM1 будет выполняться лучше всего, так как для предиката DIM1.C1 = FACT.C1 будет выполняться условие совместного размещения. Обе эти таблицы используют для разделения столбец C1.

Для объединения с DIM2 с предикатом WHERE DIM2.C2 = FACT.C2 не может использоваться локально выполняемое объединение, так как для разделения таблицы FACT используется столбец C1, а не C2.

В этом случае хорошим решением может быть репликация таблицы DIM2 в группу узлов таблицы фактов. При этом объединение может быть выполнено локально на каждом разделе.

Примечание: Реплицируемые сводные таблицы используются для репликации внутри одной базы данных. При репликации между базами данных используются регистрации и управляющие таблицы, а данные располагаются в различных базах данных и в различных операционных системах. Дополнительную информацию о репликации между базами данных смотрите в руководстве *Replication Guide and Reference*.

При создании реплицируемой сводной таблицы исходная таблица может располагаться в одноузловой группе узлов или в многоузловой группе узлов. В большинстве случаев эта таблица имеет небольшой размер и ее можно поместить в одноузловую группу узлов. Объем реплицируемых данных можно ограничить, задав только часть столбцов этой таблицы, ограничив число строк при помощи предикатов или используя при создании реплицируемой сводной таблицы оба эти метода.

Примечание: Для работы реплицируемой сводной таблицы не требуется опция захвата данных.

Реплицируемую сводную таблицу можно также создать в многоузловой группе узлов. Это та же группа узлов, в которой размещены таблицы большого объема. В этом случае копии исходной таблицы создаются на всех разделах этой группы узлов. В такой среде объединения между большой таблицей фактов и таблицами ассоциаций будут с большей вероятностью выполняться локально, и не будет выполняться передача данных исходной таблицы на все разделы.

Индексы для реплицируемых таблиц не создаются автоматически. Созданные индексы могут отличаться от индексов исходной таблицы.

Примечание: Для реплицируемых таблиц нельзя создать индексы уникальности (или задать какие-либо ограничения). Этим предотвращаются нарушения ограничений, которые не определены для исходных таблиц. Эти ограничения не разрешены, даже если такие же ограничения определены для исходной таблицы.

После использования оператора REFRESH следует выполнить утилиту RUNSTATS для реплицируемой таблицы, так же как для любой другой таблицы.

Реплицируемые таблицы можно использовать прямо в запросе. Однако нельзя использовать с реплицируемой таблицей предикат NODENUMBER(), чтобы узнать данные таблицы на конкретном разделе.

Чтобы узнать, используется ли созданная реплицируемая сводная таблица для запроса, в котором указана исходная таблица, можно использовать средство EXPLAIN. Сначала убедитесь, что существуют таблицы EXPLAIN. Затем создайте план объяснения для интересующего вас оператора SELECT. Наконец, используйте утилиту db2exfmt для форматирования выходных данных EXPLAIN.

В зависимости от того, какие данные будут использоваться в объединении, выбранный оптимизатором план доступа может использовать или не использовать реплицируемую сводную таблицу. Если оптимизатор определит, что выгоднее передать исходную таблицу на другие разделы этой группы узлов, реплицируемая сводная таблица не будет использоваться.

Стратегии объединения в многораздельной базе данных

В следующих разделах описываются стратегии объединения, возможные в среде многораздельных баз данных. В зависимости от требований каждой прикладной программы оптимизатор DB2 автоматически выберет наилучшую стратегию объединения. Описанные здесь стратегии объединения дают понять, что

происходит при каждой стратегии. “Очередь таблиц” - механизм передачи строк между разделами базы данных или между процессорами одноразделной базы данных.

В следующих описаниях *направленная* очередь таблиц - это очередь, в которой строки хешируются по принимающим разделам базы данных. *Всенаправленная* очередь таблиц - та, чьи строки посылаются всем принимающим разделам базы данных (то есть не хешируются). На диаграммах этого раздела q1, q2 и q3 - это очереди таблиц в соответствующих примерах. Таблицы в целях этих сценариев разделяются по двум разделам базы данных. Стрелками указаны направления пересылки очередей таблиц. Узел координатора является разделом 0.

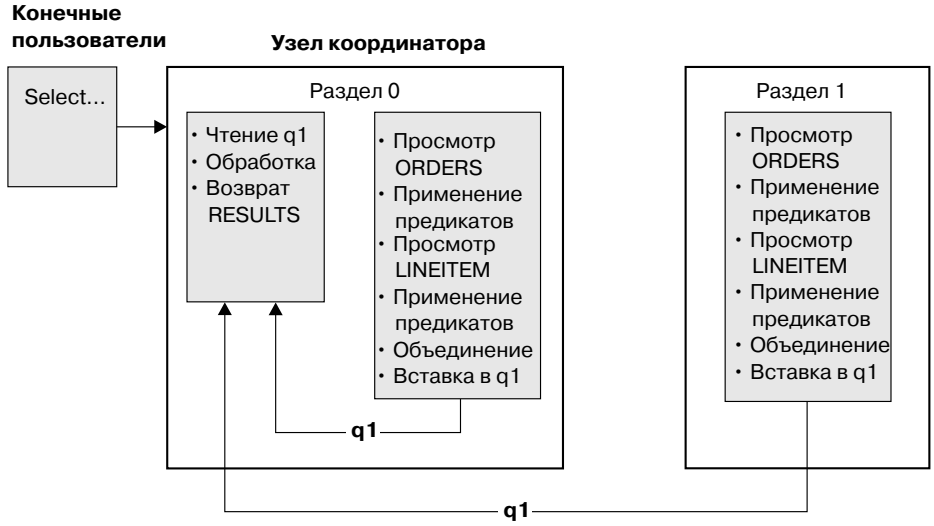
Одно соображение для таблиц многораздельной базы данных, участвующих в частых объединениях, касается совместного размещения таблиц. Совместное размещение таблиц в многораздельной базе данных дает средства находить данные из одной таблицы с помощью данных из другой таблицы того же раздела, поскольку для них используется один и тот же ключ разделения. Совместно размещенные данные при объединении, если оно потребуется в запросе, не нуждаются в перемещении в другой раздел базы данных. Только выходной набор для объединения перемещается на узел координатора. Дополнительную информацию смотрите в разделе “Совместное размещение таблиц” книги *Administration Guide: Planning*.

Информацию о зависимостях объединения смотрите в справочнике *SQL Reference*.

Совместное объединение

Чтобы оптимизатор рассмотрел вариант совместного объединения, объединяемые таблицы должны быть размещены совместно, и все пары соответствующих ключей разделения должны участвовать в предикатах равенства, задающих объединение. Пример приводится в разделе рис. 14 на стр. 195.

Примечание: Реплицируемые таблицы сводок повышают вероятность совместных объединений. Дополнительную информацию смотрите в разделе “Реплицируемые сводные таблицы” на стр. 191.



И таблица LINEITEM, и таблица ORDERS разделены по столбцу ORDERKEY.
Объединение выполняется локально на каждом разделе базы данных.
В этом примере используется предикат объединения:
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

Рисунок 14. Пример совместного объединения

Всенаправленные внешнетабличные объединения

Эта стратегия параллельного объединения может использоваться, если между объединяемыми таблицами не задано предикатов равенства. Она может использоваться и в других ситуациях, если обеспечивает минимальные расходы. Обычно это происходит, когда одна таблица очень велика, а другая очень мала, причем ни одна из них не разделена по столбцам предикатов объединения. Может оказаться “дешевле” не разделять обе таблицы, а передать меньшую таблицу на все разделы большей. Пример показан на рис. 15 на стр. 196.

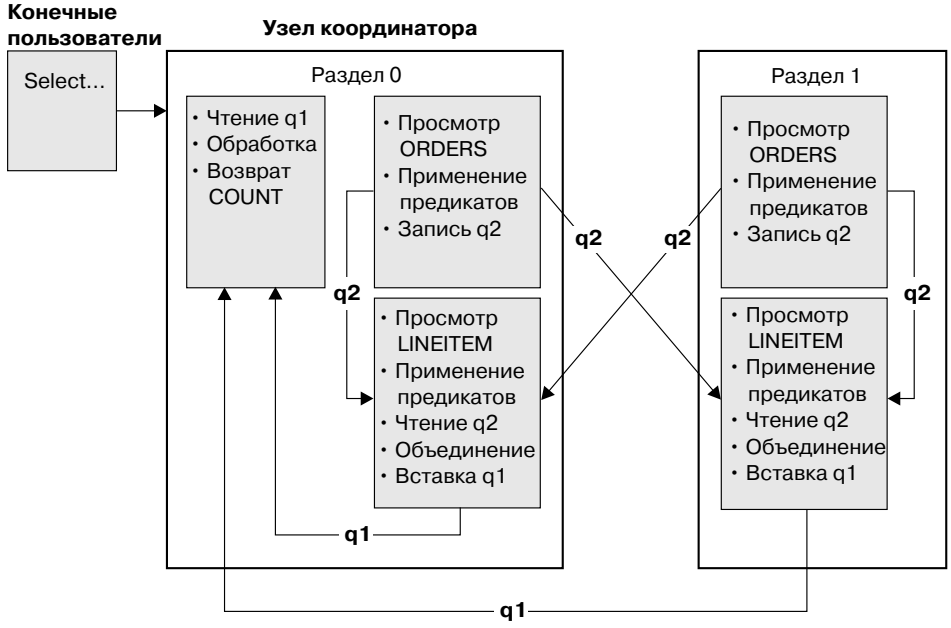


Таблица ORDERS посылается на все разделы базы данных, где есть таблица LINEITEM. Очередь таблиц q2 направляется на все разделы базы данных внутренней таблицы.

Рисунок 15. Пример всенаправленного внешнетабличного объединения

Однонаправленные внешнетабличные объединения

При этой стратегии объединения каждая строка внешней таблицы посылается на один раздел базы данных внутренней таблицы (на базе атрибутов разделения внутренней таблицы). Объединение происходит на этом разделе базы данных. Пример показан на рис. 16 на стр. 197.

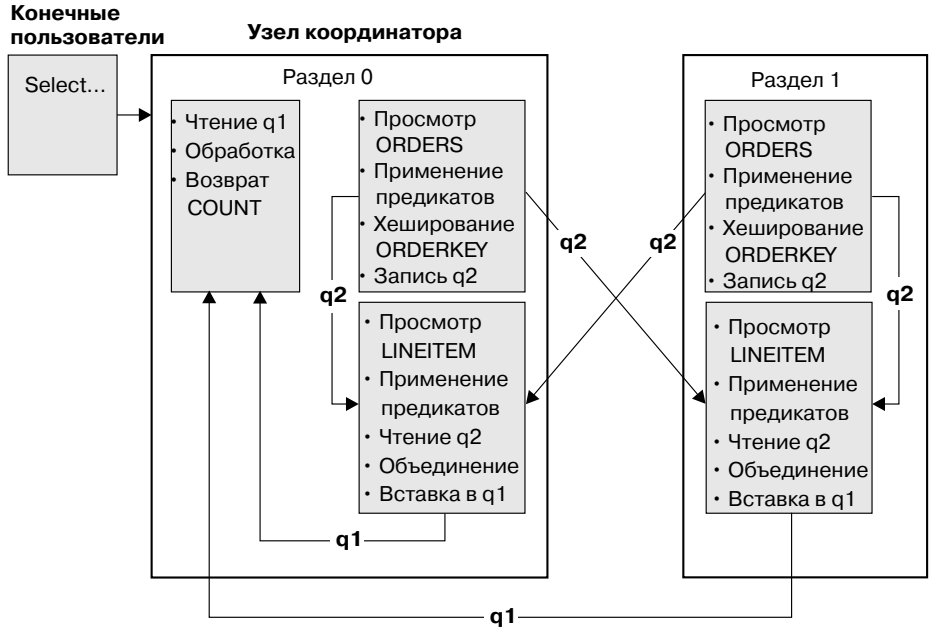
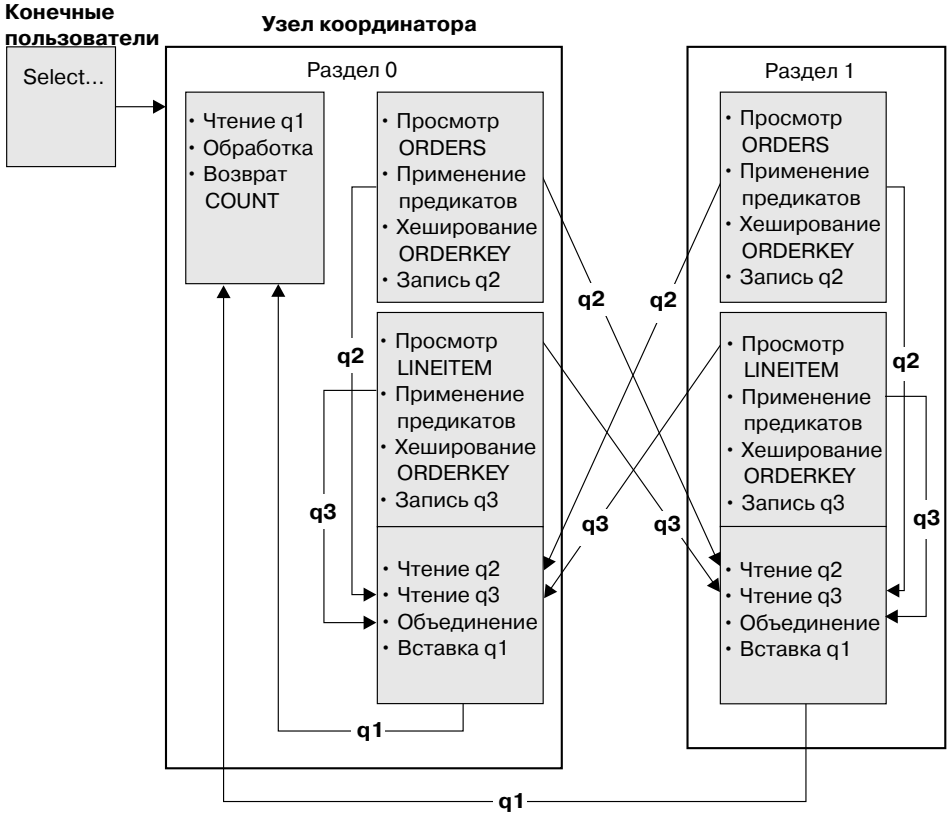


Таблица LINEITEM разделена по столбцу ORDERKEY.
Таблица ORDERS разделена по другому столбцу.
Таблица ORDERS хешируется и посылается на правильные разделы базы данных таблицы LINEITEM.
В этом примере используется предикат объединения:
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

Рисунок 16. Пример однонаправленного внешнетабличного объединения

Однонаправленные объединения внешней и внутренней таблиц

При помощи этой стратегии строки внешней и внутренней таблиц направляются набору разделов базы данных на основе значений столбцов объединения. Объединение происходит на этих разделах базы данных. Пример показан на рис. 17 на стр. 198.



Ни одна из таблиц не разделена по столбцу ORDERKEY.
Обе таблицы хешируются и посылаются на новые разделы базы данных, где происходит их объединение.
Обе очереди таблиц q2 и q3 - направленные.
В этом примере используется предикат объединения:
ORDERS.ORDERKEY = LINEITEM.ORDERKEY

Рисунок 17. Пример направленного объединения внешней и внутренней таблиц

Всенаправленные объединения внутренних таблиц

При помощи этой стратегии внутренняя таблица направляется на все разделы базы данных внешней таблицы объединения. Пример показан на рис. 18 на стр. 199.

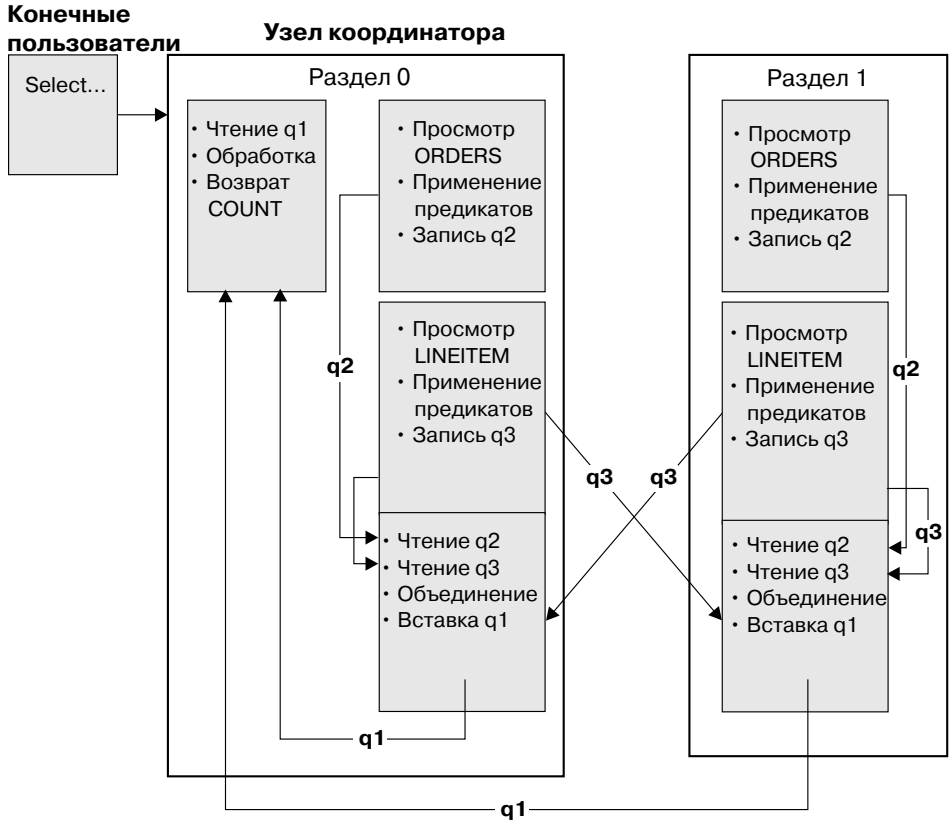


Таблица LINEITEM посылается на все разделы базы данных, где есть таблица ORDERS. Очередь таблиц q3 направляется на все разделы базы данных внешней таблицы.

Рисунок 18. Пример всенаправленного объединения внутренних таблиц

Направленные внутреннетабличные объединения

При этой стратегии каждая строка внутренней таблицы посылается на один раздел базы данных внешней таблицы объединения (на базе атрибутов разделения внешней таблицы). Объединение происходит на этом разделе базы данных. Пример показан на рис. 19 на стр. 200.

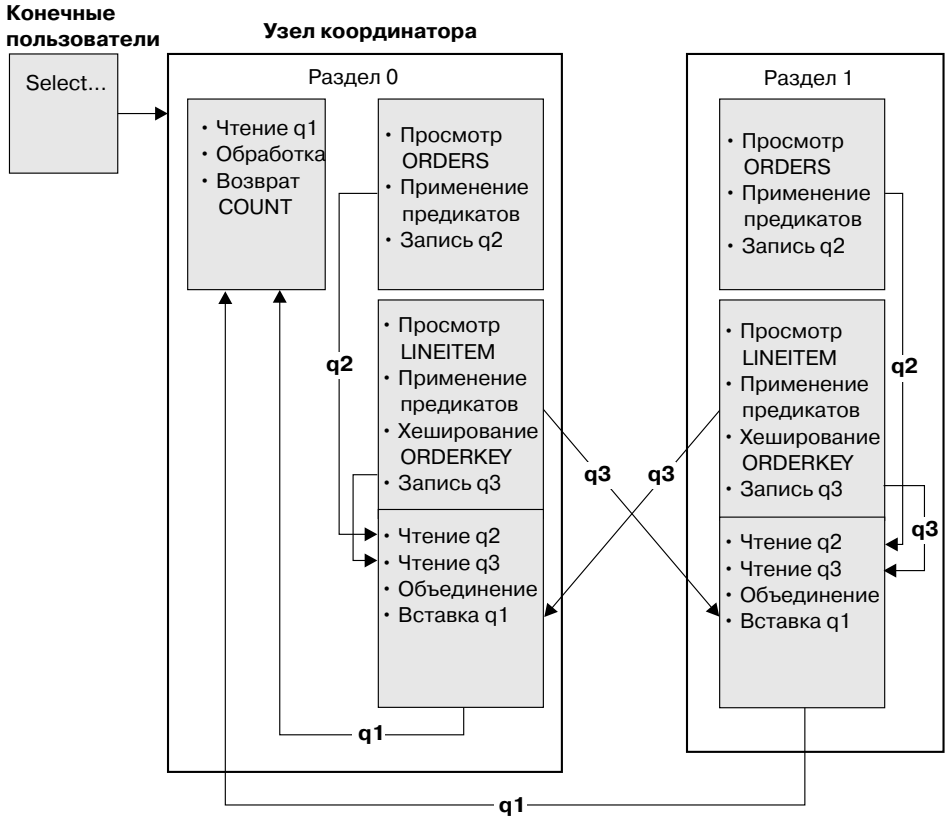


Таблица ORDERS разделена по столбцу ORDERKEY.

Таблица LINEITEM разделена по другому столбцу.

Таблица LINEITEM хешируется и посылается на нужный раздел базы данных ORDERS.

В этом примере используется предикат объединения:

ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

Рисунок 19. Пример однонаправленного внутреннетабличного объединения

Очереди таблиц

Очередь таблиц используется:

- Для пересылки данных таблицы из одного раздела базы данных на другой при использовании межраздельного параллелизма
- Для пересылки данных таблицы внутри раздела базы данных при использовании внутриразделного параллелизма
- Для пересылки данных таблицы внутри раздела базы данных при использовании одноразделной базы данных.

Каждая очередь таблиц используется для пересылки данных в одном направлении.

Компилятор решает, где требуются очереди таблиц, и включает их в план. При выполнении этого плана соединения между разделами базы данных инициируют очереди таблиц. Очереди таблиц закрываются по завершении обработки.

Существует несколько типов очередей таблиц:

- *Асинхронные очереди таблиц.* Эти очереди таблиц называются асинхронными, потому что считывают строки заранее, до операций FETCH прикладной программой. Когда выполняется FETCH, строка извлекается из очереди таблиц.

  Асинхронные очереди таблиц используются при указании условия FOR FETCH ONLY в операторе SELECT. Если производится выборка только строк, асинхронная очередь таблиц выполняется быстрее.
- *Синхронные очереди таблиц.* Эти очереди таблиц называются так потому, что считывают по одной строке на каждую операцию FETCH прикладной программой. В каждом разделе базы данных указатель ставится на следующую строку, подлежащую считыванию из этого раздела базы данных.

  Синхронные очереди таблиц используются, если вы не указываете условие FOR FETCH ONLY в операторе SELECT. В среде многораздельных баз данных при изменении строк менеджер баз данных будет использовать синхронные очереди таблиц.
- *Упорядоченные очереди таблиц.* Эти очереди таблиц сохраняют порядок.
- *Неупорядоченные очереди таблиц.* Эти очереди таблиц также называются "обычными". Они не сохраняют порядок.
- *Принимаемые очереди таблиц.* Эти очереди таблиц используются с коррелирующими подзапросами. При использовании этого типа очереди таблиц значения корреляции передаются подзапросу, а результаты возвращаются родительскому блоку запроса.

Влияние сортировки на оптимизатор

Когда оптимизатор выбирает план доступа, он принимает в расчет влияние сортировки данных на производительность. Сортировка производится, когда нет индекса, удовлетворяющего требуемому порядку выборки строк. Сортировка может также производиться, если оптимизатор определяет, что затраты на сортировку меньше, чем на просмотр индекса. При сортировке данных оптимизатор может выполнять одно из следующих действий:

- "Конвейерная" передача результатов сортировки после выполнения запроса. Смотрите разделы "Сравнение конвейерной и неконвейерной сортировки" на стр. 202 и "Параметры конфигурации, влияющие на оптимизацию запросов" на стр. 95.
- Внутренняя обработка сортировки внутри менеджера базы данных. Смотрите раздел "Операции изменения уровня группировки и сортировки" на стр. 202.

Сравнение конвейерной и неконвейерной сортировки

При завершении сортировки, если конечный сортированный список данных может читаться за один последовательный проход, результаты можно направить на *конвейер*. Конвейерная передача результатов сортировки происходит быстрее, чем другие (неконвейерные) методы передачи результатов сортировки. Если это возможно, оптимизатор выбирает конвейерную сортировку.

Независимо от того, каким способом передаются результаты сортировки, время на сортировку будет зависеть от ряда факторов, включая число сортируемых строк, размер ключа и длину строки. Если сортируемые строки занимают больше, чем доступно пространства в куче сортировки, сортировка выполняется в несколько проходов, причем за каждый проход сортируется поднабор целого набора строк. Каждый проход хранится во временной таблице в пуле буферов. (В ходе управления пулом буферов возможна запись страниц временной таблицы на диск.) По завершении всех проходов сортировки все сортированные поднаборы должны быть слиты в единый сортированный набор строк. Для конвейерной сортировки по мере слияния строки передаются непосредственно службам реляционных данных.

Дополнительную информацию смотрите в разделе “Индикаторы проблем с производительностью сортировки” на стр. 279 или в обсуждении параметра конфигурации *sortheap* в разделе “Параметры конфигурации, влияющие на оптимизацию запросов” на стр. 95.

Операции изменения уровня группировки и сортировки

В некоторых случаях оптимизатор может решить передать сортировку или группировку с уровня служб реляционных данных на уровень служб управления данными. Изменение уровня этих операций повышает производительность, разрешая службам управления данными передавать данные непосредственно процедуре сортировки или группировки. Без этого службы управления данными сначала передадут эти данные службам реляционных данных, которые затем вызывают процедуры сортировки или группировки. Например, следующий запрос выигрывает от этой оптимизации:

```
SELECT WORKDEPT, AVG(SALARY) AS AVG_DEPT_SALARY
    FROM EMPLOYEE
 GROUP BY WORKDEPT
```

Группировка при сортировке

Если сортировка используется для получения требуемого для операции GROUP BY порядка, оптимизатор имеет возможность частично или полностью выполнить требуемую GROUP BY группировку еще при выполнении сортировки. Это дает преимущество, если число строк в каждой группе велико. Это преимущество усиливается, если при частичной группировке при сортировке уменьшается или исчезает необходимость записи данных сортировки на диск.

При использовании группировки при сортировке используется до трех стадий группировки, чтобы убедиться в правильности вычисленных результатов. На первой стадии группировки, “частичной группировке”, вычисляются суммарные значения, пока не наполнится куча сортировки. Частичная группировка - процесс обработки несгруппированных данных с получением частичных суммарных значений. Если куча сортировки заполнена, остаток данных сбрасывается на диск, включая все частичные суммарные значения, вычисленные на этой стадии наполнения кучи сортировки. Вслед за очисткой кучи сортировки запускаются новые группировки.

На второй стадии группировки - “промежуточной” - берутся все записанные на диск частичные результаты сортировки и производится дальнейшее суммирование по ключам группировки. Группировка не может завершиться, потому что столбцы ключей группировки - поднабор столбцов ключей разделения. Промежуточная группировка берет имеющиеся частичные суммарные значения и формирует новые частичные суммарные значения. Эта стадия не обязательна; она используется как для внутри-, так и для межраздельного параллелизма. В последнем случае группировка завершается, если получен глобальный ключ группировки. При межраздельном параллелизме это происходит, когда ключ группировки является подмножеством ключа разделения, делящего группы по разделам и таким образом для завершения группировки нужно повторное разбиение по разделам. Подобный случай имеет место и при внутриразделном параллелизме, когда каждый агент завершает слияние своих записанных на диск проходов сортировки перед тем, как единственный агент завершит группировку.

Последняя стадия группировки, “конечная группировка”, берет все частичные суммарные значения и завершает группировку. Конечная группировка берет частичные суммарные значения и формирует конечные суммарные значения. Этот этап всегда выполняется для оператора GROUP BY. Сортировка не может выполнить полную группировку, потому что нет гарантий, что результаты сортировки не будет сброшены на диск. Полная группировка берет несгруппированные данные и образует конечные суммарные значения. Этот метод группировки обычно используется при группировке данных, уже следующих в правильном порядке, если разделение не препятствует его использованию.

Стратегии оптимизации для внутриразделного параллелизма

Оптимизатор может выбрать план доступа так, чтобы запрос выполнялся параллельно внутри раздела базы данных, если степень параллелизма указана при компиляции оператора SQL.

Во время выполнения для обработки запроса создаются многочисленные агенты базы данных, называемые “подагентами”. Число подагентов меньше или равно степени параллелизма, определяемой при компиляции оператора SQL.

Дополнительную информацию о задании степени параллелизма для операторов SQL смотрите в разделе “Параллельная обработка прикладных программ” на стр. 92. Дополнительную информацию об агентах и подагентах смотрите в разделе “Агенты базы данных” на стр. 287.

В многораздельной базе данных степень параллелизма применяется отдельно к каждому разделу. Например, часть запроса, выполняемая на данном разделе базы данных, будет распараллелена на базе степени параллелизма, определенной на этом разделе базы данных для данного оператора SQL.

План доступа распараллеливается путем его деления на части, выполняемые каждым подагентом, и часть, выполняемую координирующим агентом. Подагенты посредством очередей таблиц передают данные координирующему агенту или другим подагентам. В многораздельной базе данных подагенты через очереди таблиц могут обмениваться данными с подагентами других разделов базы данных.

Далее описываются стратегии распараллеливания для отдельного раздела базы данных.

Стратегии параллельного просмотра

Реляционный просмотр и просмотр индекса могут выполняться параллельно на одной и той же таблице или индексе. Для параллельного реляционного просмотра таблица делится на интервалы страниц или строк. Интервал страниц или строк назначается подагенту. Подагент просматривает отведенный ему интервал и по завершении просмотра получает новый интервал.

При параллельном просмотре индекса индекс делится на интервалы записей на основе значений индексных ключей и числа индексных записей для значения ключа. Параллельный просмотр индекса происходит подобно параллельному просмотру таблиц; подагентам отводятся интервалы записей. Когда подагент завершает работу с текущим интервалом, ему выделяется новый интервал.

Единица просмотра (страница или строка) и кратность просмотра определяются оптимизатором.

Параллельный просмотр обеспечивает равное распределение работы между подагентами. Цель параллельного просмотра - сбалансировать загрузку подагентов и равномерно загрузить их работой. Если число занятых подагентов равно числу доступных процессоров, а диски не перегружены запросами ввода/вывода, ресурсы компьютера используются эффективно.

Другие операции плана доступа могут вызвать дисбаланс данных при выполнении запроса. Оптимизатор выбирает стратегии параллелизма так, чтобы поддерживать баланс данных.

Стратегии параллельной сортировки

Оптимизатор может выбрать одну из следующих стратегий параллельной сортировки:

Карусельная сортировка

Этот вид сортировки называется также “сортировкой с перераспределением”. Это эффективная сортировка с совместным использованием памяти, которая пытается перераспределять данные между всеми подагентами, насколько возможно, поровну. Для равномерности распределения она использует карусельный алгоритм. Вначале задает каждому подагенту отдельную сортировку. На фазе вставки подагенты вставляют значения в каждую отдельную сортировку по принципу карусели. Этим достигается более равномерное распределение данных.

Распределенная сортировка

При этой сортировке, как и при карусельной сортировке, для каждого подагента создается своя сортировка. Подагенты применяют функцию хеширования к столбцам сортировки, чтобы определить, в какую из сортировок надо вставить строку. Например, если внутренняя и внешняя таблицы объединения слиянием отсортированы по разделам, подагент может использовать объединение слиянием для объединения соответствующих разделов. Это позволяет выполнять объединение слиянием параллельно.

Реплицируемая сортировка

Эта сортировка используется, когда всем подагентам требуются все выходные данные сортировки. Создается одна сортировка, и подагенты синхронизируются при вставке в сортировку. Когда сортировка завершается, каждый подагент считывает всю сортировку. Эта сортировка может использоваться для повторного уравновешивания потока данных, если число строк мало.

Совместная сортировка

Эта сортировка совпадает с реплицируемой за тем исключением, что подагенты открывают параллельный просмотр результата сортировки. При этом данные распространяются среди подагентов, как при карусельной сортировке.

Параллельные временные таблицы

Подагенты могут кооперироваться для создания временной таблицы путем вставки строк в одну и ту же таблицу. Ее называют совместно используемой временной таблицей. Подагенты могут открывать частный просмотр или параллельный просмотр совместно используемой временной таблицы в зависимости от того, реплицируется или разделяется поток данных.

Стратегии параллельной группировки

Операции группировки могут выполняться подагентами параллельно. Операция группировки требует, чтобы данные в группируемых столбцах были упорядочены. Если подагент в состоянии гарантировать получение всех строк набора значений группируемых столбцов, он может выполнить полную

группировку. Это может случиться, если поток уже распределен по группируемым столбцам в результате предварительно выполненной распределенной сортировки.

В противном случае этот подагент может выполнить частичную группировку и использовать другую стратегию завершения группировки. Некоторые из этих стратегий:

- Переслать частично сгруппированные данные координирующему агенту через очередь таблиц слияния. Координатор завершит группировку.
- Вставить частично сгруппированные данные в распределенную сортировку. Эта сортировка распределена по группируемым столбцам. Этим гарантируется, что все строки для набора группируемых столбцов содержатся в одном разделе сортировки.
- Если поток нуждается в репликации по причинам балансировки, частично сгруппированные данные можно вставить в реплицируемую сортировку. Каждый подагент завершает группировку, используя реплицируемую сортировку, и получает идентичную копию результата группировки.

Стратегии параллельного объединения

Операции объединения могут выполняться подагентами параллельно. Стратегии параллельного объединения определяются характеристиками потока данных.

Объединение можно распараллелить путем распределения и/или реплицирования потока данных на внутренней и внешней таблицах объединения. Например, объединение с вложенным циклом можно распараллелить, если его внешний поток распределен благодаря параллельному просмотру, а внутренний поток проверяется каждым подагентом независимо. Объединение слиянием можно распараллелить, если его внутренний и внешний потоки распределены по значениям благодаря сортировкам разделения.

Автоматические сводные таблицы

Сводные таблицы - мощное средство уменьшить время ответа на запрос. Во многих средах, где могут быть предусмотрены некоторые базовые структуры запросов, сводные таблицы можно использовать для:

- Группировки данных по одному или нескольким столбцам
- Объединения и группировки данных в группе таблиц
- Идентификации поднабора данных частого доступа (то есть “активного” горизонтального или вертикального раздела)
- Повторного разбиения таблицы или ее части в среде многораздельной базы данных

Компилятор SQL принимает во внимание сводные таблицы. При компиляции SQL на этапах перезаписи запроса (смотрите раздел “Перезапись запросов

компилятором SQL" на стр. 159) и оптимизации (смотрите раздел "Концепции доступа к данным и оптимизация" на стр. 169) запросы сопоставляются со сводными таблицами и определяется, можно ли заменить обращение к базовым таблицам обращением к сводной таблице. Когда сводные таблицы используются для ответов на запросы, возможности объяснения (смотрите "Глава 7. Возможность объяснения SQL" на стр. 225) позволяют узнать, какая сводная таблица была использована. Так как во многих случаях сводные таблицы сводок ведут себя как обычные, к ним применимы те же соображения по поводу оптимизации доступа к данным с использованием определений табличных пространств, создания индексов и выполнения RUNSTATS.

Чтобы помочь вам ощутить возможности сводных таблиц, приведем следующий пример запроса многоразмерного анализа, демонстрирующий использование преимуществ сводных таблиц.

В этом примере мы предположим, что хранилище данных содержит массив покупателей и массив счетов кредитных карт. В хранилище записывается набор транзакций, сделанных по кредитным картам. Каждая транзакция содержит набор товаров, которые приобретались вместе. Эту среду можно охарактеризовать, как ″мультизвездную″, потому что обе таблицы, одна из которых содержит элементы транзакций, а другая идентифицирует транзакции приобретения товаров, велики; вместе образуют центр звезды.

Существуют три иерархических атрибута, описывающих транзакцию: продукт, положение и время. Иерархия продуктов записана в двух нормализованных таблицах, представляющих группу продуктов и линейку продуктов. Иерархия положений содержит информацию о городе, штате и стране и представлена одиночной ненормализованной таблицей. Иерархия времени содержит информацию о дне, месяце и годе, которые закодированы в одном поле даты. Отдельные части даты извлекаются из поля даты транзакции при помощи встроенных функций. В этом сценарии есть и другие таблицы, представляющие информацию о счетах покупателей и о самих покупателях.

Создана сводная таблица, содержащая суммы и количества покупок для каждого уровня:
- Иерархии товаров
- Иерархии положений
- Иерархии времени по годам, месяцам и дням.

Из этой совокупности хранимых данных можно почерпнуть ответы на широкий спектр запросов. В следующем примере вычисляется сумма и счет продаж по группе и линейке товаров, по городу, штату и стране, а также по времени. Пример включает также ряд других столбцов в условии GROUP BY.

```
CREATE TABLE dba.PG_SALESSUM
  AS (
      SELECT l.id AS prodline, pg.id AS pgroup,
```

```
            loc.country, loc.state, loc.city,
            l.name AS linename, pg.name AS pgname,
            YEAR(pdate) AS year, MONTH(pdate) AS month,
            t.status,
            SUM(ti.amount) AS amount,
            COUNT(*) AS count
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
         AND YEAR(pdate) > 1990
      GROUP BY l.id, pg.id, loc.country, loc.state, loc.city,
               year(pdate), month(pdate), t.status, l.name, pg.name
    )
DATA INITIALLY DEFERRED REFRESH DEFERRED;

REFRESH TABLE dba.SALESCUBE;
```

Сводная таблица обычно намного меньше базовых таблиц фактов. Вы можете задать моменты обновления сводной таблицы, указав опцию DEFERRED (как показано в нашем примере).

К запросам, использующим преимущества подобных предвычисленных сумм, относятся запросы:

- Продаж по месяцам и группам товаров
- Годовые итоги продаж после 1990 года
- Продажи за 1995 или 1996 годы
- Суммы продаж по группе или линейке товаров
- Сумма продаж для указанной группы товаров или линейке товаров за 1995 и 1996 годы
- Сумма продаж для указанной страны.

Несмотря на то, что точного ответа на любой из этих запросов в сводной таблице нет, затраты на вычисление ответа с использованием сводной таблицы могут оказаться значительно меньше, чем при использовании большой базовой таблицы, поскольку часть ответа уже вычислена. Затраты на объединения, сортировки и группировки базовых данных при использовании сводных таблиц исключаются или сокращаются.

Ниже приводятся примеры запросов, получающих значительный выигрыш в производительности в результате использования уже вычисленных результатов из сводных таблиц. Первый пример возвращает общий объем продаж за 1995 и 1996 годы:

```
 SET CURRENT REFRESH AGE=ANY

SELECT YEAR(pdate) AS year, SUM(ti.amount) AS amount
FROM   cube.transitem AS ti, cube.trans AS t,
```

```
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY year(pdate);
```

Второй пример возвращает общий объем продаж по группе товаров за 1995 и 1996 годы:

```
SET CURRENT REFRESH AGE=ANY

SELECT pg.id AS "PRODUCT GROUP",
       SUM(ti.amount) AS amount
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY pg.id;
```

Экономия для таких запросов тем существеннее, чем больше размеры баз данных. Это происходит потому, что сводная таблица растет медленнее, чем базовая таблица. Преимущество сводных таблиц состоит в том, что DB2 Universal Database использует их для эффективного исключения повторяющихся действий над запросами путем выполнения расчетов один раз при построении сводных таблиц и повторном использовании их содержания для огромного числа запросов.

Фазы компиляции запросов к базам данных объединения

В этом разделе описываются дополнительные фазы обработки запросов в системе базы данных объединения. В нем также даются рекомендации по повышению производительности запросов к базе данных объединения. Основные темы:

- “Анализ изменения уровня”
- “Генерация удаленного SQL и глобальная оптимизация” на стр. 218.

Анализ изменения уровня

Анализ изменения уровня сообщает оптимизатору DB2, может ли выполнить операцию на удаленном источнике данных. Операция может быть функцией (реляционной операцией, системной или пользовательской функцией) или оператором SQL (GROUP BY, ORDER BY и т.п.).

Функции, которые нельзя передать на источник, могут значительно повлиять на производительность запроса. Рассмотрим случай, когда предикат отбора надо выполнять локально, не передавая на источник данных. В этом случае от DB2 может потребоваться получить всю таблицу из удаленного источника данных и отфильтровать ее локально по этому предикату. Если пропускная способность вашей сети ограничена, а таблица велика, производительность запроса может сильно пострадать.

Операции, которые нельзя передать на источник, также могут существенно повлиять на производительность запроса. Например, если операция GROUP BY суммирует удаленные данные локально, DB2 снова должна получить всю таблицу из удаленного источника данных.

Предположим для примера, что псевдоним N1 ссылается на таблицу EMPLOYEE на источнике данных DB2 для OS/390. Предположим далее, что эта таблица содержит 10000 строк, один из столбцов содержит фамилии сотрудников, а другой - заработную плату. Для оператора:

```
SELECT LASTNAME, COUNT(*)  FROM N1
  WHERE LASTNAME  > 'B' AND SALARY > 50000
  GROUP BY LASTNAME;
```

есть несколько возможностей:

- Если последовательности упорядочивания в DB2 и DB2 для OS/390 одинаковы, скорее всего, этот предикат запроса будет передан на DB2 для OS/390. Как правило, эффективнее фильтровать и группировать результаты на источнике данных, а не копировать всю таблицу на систему объединения и выполнять эти операции локально. Анализ изменения уровня в системах объединения определяет, можно ли выполнить операции на источнике данных. В данном случае и предикат, и операцию GROUP BY можно выполнить на источнике данных.
- Если последовательности упорядочивания различны, анализ изменения уровня определит, что весь предикат нельзя оценить на источнике данных; однако оптимизатор может решить передать на источник часть предиката `SALARY > 50000`. Сравнение LASTNAME > ’B’ придется проводить на системе объединения.
- Если последовательности упорядочивания совпадают, но оптимизатор знает, что локальный сервер DB2 работает гораздо быстрее, оптимизатор, возможно, решит, что выполнение операции `GROUP BY` локально в DB2 - лучший (наименее затратный) вариант. Этот предикат будет оцениваться на источнике данных. Это пример анализа уровня в сочетании с глобальной оптимизацией. DB2 примет во внимание доступные пути и затем выберет наиболее эффективный план.

Основная цель - сделать так, что оптимизатор мог рассмотреть вариант передачи функций и операторов на источник данных. Будут ли функция или

оператор SQL выполняться на удаленном источнике данных, зависит от многих факторов. Есть три группы ключевых факторов: характеристики сервера, характеристики псевдонима и характеристики запроса.

Характеристики сервера, влияющие на возможность изменения уровня

В следующих разделах описываются зависимые от источника данных факторы, способные влиять на возможность изменения уровня. В целом эти факторы существуют потому, что DB2 позволяет использовать богатый диалект SQL для формулировки запросов. Возможно, что этот диалект предлагает большую функциональность, чем диалект SQL, поддерживаемый сервером, к которому происходит обращение при запросе DB2. DB2 может компенсировать отсутствие функции на сервере данных, однако это может потребовать выполнения соответствующей операции на системе объединения.

Возможности SQL: Каждый источник данных поддерживает некоторую разновидность диалекта SQL и различные уровни функциональности. Например, рассмотрим список GROUP BY. Большинство источников данных поддерживает оператор GROUP BY, но в некоторых существуют ограничения на число элементов в списке GROUP BY или на выражения, разрешенные для этого списка. Если на удаленном источнике данных есть ограничение, DB2 может оказаться вынужденным выполнять операцию GROUP BY локально.

Ограничения SQL: Каждый источник данных может иметь различные ограничения SQL. Например, некоторым источникам данных требуются маркеры параметров для связывания значений с удаленными операторами SQL. Следовательно, должны быть проверены ограничения на маркеры параметров, чтобы убедиться, что каждый источник данных сможет поддерживать такой механизм связывания. Если DB2 не может определить хороший метод связывания значения функции, эта функция должна выполняться локально.

Пределы SQL: DB2 может разрешить использование больших по величине целых чисел, чем удаленные источники данных, однако числа, превышающие предельное значение, не могут включаться в операторы, посылаемые источникам данных. Следовательно, функция или операция, использующие такую константу, должны оцениваться локально.

Специфика сервера: В эту категорию попадают многие факторы. Один из примеров - сортировка значений NULL (как наибольшее, как наименьшее или в зависимости от порядка). Например, если значение NULL сортируется на источнике данных не так, как в DB2, операции ORDER BY на выражении, допускающем пустые значения, нельзя выполнять удаленно.

Последовательность упорядочивания: Если для базы данных объединения задана та же последовательность упорядочивания, что и на источнике данных, и

опция сервера *collating_sequence* имеет значение 'Y', оптимизатор может рассмотреть возможность передачи на источник предикатов сравнения интервала символов.

Если запрос сервера объединения требует сортировки, место, где выполняется сортировка, зависит от ряда факторов. Если последовательность упорядочивания для базы данных объединения та же, что на источнике, где хранятся запрашиваемые данные, сортировку можно производить на источнике данных. Если последовательности упорядочивания совпадают, оптимизатор может выбрать, какая сортировка - локальная или на источнике данных - эффективнее для выполнения запроса. Аналогичным образом если запрос требует сравнения символьных данных, это сравнение также можно выполнять на источнике данных.

Числовые сравнения в целом могут выполняться в любом месте, даже если последовательности упорядочивания различны. Однако вы можете получить неожиданные результаты, если порядок символов null на базе данных объединения и на источнике данных отличается. Более того, будьте осторожны с операторами сравнения, передавая их регистронезависимому источнику данных. На регистронезависимом источнике данных ″I″ и ″i″ считаются совпадающими. По умолчанию DB2 учитывает регистр и может назначать этим символам разные веса.

Если последовательности упорядочивания базы данных объединения и источника данных различаются, DB2 принимает данные на систему объединения, чтобы сортировать и сравнивать их локально. Причина в том, что пользователи ожидают увидеть результаты запроса сортированными согласно последовательности упорядочивания, определенной на сервере объединения; упорядочивая данные локально, сервер объединения гарантирует, что это ожидание осуществится.

Получение данных для локальной сортировки и сравнения обычно снижает производительность. Следовательно, продумайте конфигурирование базы данных объединения, чтобы использовать ту же последовательность упорядочивания, что и на ваших источниках данных. Таким образом можно увеличить производительность, потому что сервер объединения может передать сортировку и сравнение на источники данных. Например, в DB2 UDB for OS/390 сортировки, определяемые условиями ORDER BY, осуществляются на основе последовательности упорядочивания кодовой страницы EBCDIC. Если вы хотите использовать сервер объединения, чтобы получать данные DB2 для OS/390 сортированными в соответствии с условиями ORDER BY, желательно сконфигурировать базу данных объединения так, чтобы она использовала предопределенную последовательность упорядочивания на базе кодовой страницы EBCDIC.

Если последовательности упорядочивания на базе данных объединения и на источнике данных различаются, а вам требуются данные, сортированные в последовательности источника данных, вы можете выполнить свой запрос через промежуточный сервер или определить запрос к производной таблице источника данных.

Дополнительную информацию о последовательностях упорядочивания и их задании смотрите в руководстве *Administration Guide: Planning*, дополнительную информацию об опции сервера *collating_sequence* смотрите в разделе Табл. 8 на стр. 112.

Опции сервера: Ряд опций сервера может влиять на возможности изменение уровня. В частности, посмотрите, какие значения заданы для параметров *collating_sequence*, *varchar_no_trailing_blanks* и *pushdown*. Информацию о задании этих параметров смотрите в разделе “Опции сервера, влияющие на запросы базы данных объединения” на стр. 111.

Факторы отображения типов и отображения функций DB2: Предоставляемые DB2 по умолчанию отображения типов локальных данных (сведения о таблицах типов данных смотрите в разделе *Application Development Guide*) созданы так, чтобы каждому типу данных источника отводилось достаточное буферное пространство (во избежание потери данных). Пользователь может выполнить настройку отображения типов для конкретного источника данных, чтобы удовлетворить требования конкретной прикладной программы. Например, если вы обращаетесь к столбцу источника данных Oracle с типом данных DATE (который по умолчанию отображается в DB2 на тип данных TIMESTAMP), вы можете изменить локальный тип данных на DB2 DATE.

DB2 может компенсировать функции, не поддерживаемые источником данных. Компенсация функций происходит в трех случаях:

- Функция не существует в удаленном источнике данных.
- Функция существует, однако характеристики операнда не соответствуют ограничениям этой функции. Пример этой ситуации - реляционная операция IS NULL. Она поддерживается на большинстве источников данных, но некоторые из них могут иметь ограничения, например, в левой части оператора IS NULL может допускаться только имя столбца.
- Функция при удаленном выполнении может возвращать другой результат. Пример этой ситуации - операция ’>’ (больше чем). Для источников данных с иными последовательностями упорядочивания она может возвращать иные результаты, чем при локальной оценке в DB2.

Характеристики псевдонима, влияющие на возможности изменения уровня

В следующих разделах описаны специфичные для псевдонима факторы, способные повлиять на возможности изменения уровня.

Тип локальных данных столбца псевдонимов: Убедитесь, что тип локальных данных столбца не препятствует передаче предиката на источник данных. Как упоминалось ранее, отображения типов данных по умолчанию призваны избежать возможного переполнения. Однако предикат объединения двух столбцов различной длины может не рассматриваться на источнике данных, столбец объединения у которого короче, в зависимости от того, как DB2 связывает более длинный столбец. Эта ситуация может влиять на число возможностей последовательности объединения, оцениваемых оптимизатором DB2. Например, столбцам источника данных Oracle, созданным с использованием типа данных INTEGER или INT, соответствует тип NUMBER(38). Столбцу псевдонима для этого типа данных Oracle будет дан локальный тип данных FLOAT, потому что интервал целочисленных значений integer в DB2 - от 2**31 до (-2**31)-1, что приблизительно эквивалентно NUMBER(9). В этом случае объединения между столбцом целых DB2 и столбцом целых Oracle не могут происходить на источнике данных DB2 (более короткий объединяемый столбец); однако если значения этого целочисленного столбца Oracle можно поместить в тип данных DB2 INTEGER, измените с помощью оператора ALTER NICKNAME его локальный тип данных, так чтобы объединение могло производиться на источнике данных DB2.

Опции столбцов: Оператор ALTER NICKNAME SQL можно использовать, чтобы добавить или изменить опции столбца для псевдонимов.

Одна из этих опций - ″varchar_no_trailing_blanks″. Ее можно использовать для идентификации столбца, не содержащего конечных пробелов. При компиляции на этапе изменение уровня анализа эта информация будет принята во внимание при проверке всех операций, выполняемых на столбцах с таким атрибутом. На основе этого DB2 может сгенерировать другую, но эквивалентную, форму предиката для удаленного операторе SQL, посылаемом источнику данных. Пользователь может увидеть, что на источнике данных обрабатывается другой предикат, но итоговый результат будет тем же.

Еще одна опция столбца - numeric_string. Используйте эту опцию для указания, что значения в этом столбце всегда числовые без конечных пробелов.

В разделе Табл. 15 на стр. 215 описаны значения опций столбцов и значения по умолчанию.

Таблица 15. Опции столбцов и их значения

Опция	Допустимые значения		Значение по умолчанию
numeric_string	'Y'	Этот столбец содержит только строки числовых данных. ВНИМАНИЕ: Если этот столбец содержит только числовые строки, дополненные хвостовыми пробелами, не рекомендуется задавать значение 'Y'.	'N'
	'N'	Этот столбец может содержать не только строки числовых данных.	
	Задав для опции столбца numeric_string значение 'Y', вы указываете оптимизатору, что этот столбец не содержит пробелов, которые могут повлиять на сортировку данных столбца. Эта опция полезна, если последовательность слияния источника данных отличается от последовательности слияния DB2. Столбцы, для которых задана эта опция, не будут исключаться из локальных (на источнике данных) вычислений при несовпадении последовательностей слияния.		
varchar_no_trailing_blanks	Задает, что источник данных использует семантику сравнения varchar без дополнения пробелами. В некоторых СУБД методы сравнения символьных строк переменной длины без хвостовых пробелов возвращают те же результаты, что и методы сравнения DB2. Если точно известно, что все столбцы типа VARCHAR таблиц или производных таблиц этого источника данных не содержат хвостовых пробелов, возможно, имеет смысл задать значение 'Y' для этой опции сервера этого источника данных. Эта опция часто используется с источниками данных Oracle**. Убедитесь, что учтены все объекты (включая производные таблицы), которые потенциально могут иметь псевдонимы.		'N'
	'Y'	Этот источник данных имеет методы сравнения данных, не дополненных пробелами, аналогичные методам сравнения DB2.	
	'N'	Этот источник данных не имеет таких методов сравнения данных, не дополненных пробелами, как DB2.	

Характеристики запроса, влияющие на возможности изменения уровня

Запрос может ссылаться на оператор SQL, включающий псевдонимы из множества источников данных. Если DB2 должен объединить результаты двух заданных источников данных при помощи одной операции над множествами (например, UNION), операция должна производиться на систем объединения. Такие операции нельзя выполнять непосредственно на удаленном источнике данных.

Проверка и понимание решений анализа изменения уровня

Перезапись операторов SQL может предоставить дополнительные возможности изменения уровня для обработки запросов в DB2. В этом разделе описаны инструменты для определения, где происходит обработка запроса, перечисляются наиболее частые вопросы (и предлагаемые области исследования), связанные с анализом запросов, и в заключительном разделе описаны обновления источников данных.

Анализ места выполнения запроса: В DB2 есть две утилиты, показывающие, где выполняются запросы:

- Наглядное объяснение Visual explain. Запускается командой **db2cc** или **db2vexp**. Используйте его для просмотра графа плана доступа для запроса. Место выполнения для каждого оператора включается в подробный показ оператора.

  Если запрос полностью передан на источник, вы увидите операцию RETURN выше операции RQUERY. Операция RETURN - стандартная операция DB2; операция RQUERY уникальна для операций базы данных объединения. RQUERY посылает источнику данных оператор SQL SELECT, чтобы получить результаты запроса. Оператор SELECT генерируется с использованием диалекта SQL, поддерживаемого источником данных. Он может содержать любой правильный для этого источника данных запрос.
- Объяснение SQL. Запускается командой **db2expln** или **dynexpln**. Используйте его для просмотра стратегии плана доступа в виде текста.

Как понять, почему запрос выполняется на источнике данных или в системе объединения: В этом разделе дан перечень типичных вопросов по анализу плана и областей исследования по увеличению возможностей изменение уровня. Ключевые вопросы:

- Почему этот предикат не обрабатывается удаленно?

  Этот вопрос возникает, если предикат очень селективен и поэтому мог бы использоваться для фильтрации строк и уменьшения сетевого трафика. Удаленная оценка предиката определяет также, может ли объединение двух таблиц одного и того же источника данных выполняться удаленно.

  Области для исследования включают:

  – Предикаты подзапроса. Содержит ли этот предикат подзапрос, относящийся к другому источнику данных? Содержит ли этот предикат подзапрос, включающий не поддерживаемый этим источником данных оператор SQL? Не все источники данных поддерживают операции над множествами в предикате.

  – Функции предикатов. Содержит ли этот предикат функцию, которую нельзя выполнить на этом удаленном источнике данных? Операции отношений рассматриваются как функции.

- Требования связи предикатов. Требует ли этот предикат при удаленной обработке связывания с некоторым значением. Если да, не будет ли оно нарушать ограничения этого источника данных?
- Глобальная оптимизация. Оптимизатор может решить, что локальная обработка требует меньших затрат. Дополнительную информацию смотрите в разделе “Генерация удаленного SQL и глобальная оптимизация” на стр. 218.

- Почему оператор GROUP BY не выполняется удаленно?

  Можно проверить ряд областей:

  - Вычисляются ли удаленно исходные данные оператора GROUP BY? Если ответ отрицательный, проверьте входные данные.
  - Налагает ли источник данных ограничения на эту операцию? Примеры:
    - Ограниченное число элементов GROUP BY
    - Ограниченное число байтов в объединенных элементах GROUP BY
    - Спецификация столбцов только для списка GROUP BY
  - Поддерживает ли источник данных эту операцию SQL?
  - Глобальная оптимизация. Оптимизатор может решить, что локальная обработка требует меньших затрат. Дополнительную информацию смотрите в разделе “Генерация удаленного SQL и глобальная оптимизация” на стр. 218.

- Почему операция над множествами не выполняется удаленно?

  Можно проверить ряд областей:

  - Оба ли операнда полностью вычисляются на одном и том же удаленном источнике данных? Если ответ отрицательный, а вы предполагали, что он должен быть положительным, проверьте каждый операнд.
  - Налагает ли источник данных ограничения на эту операцию над множествами? Например, допускаются ли большие объекты или длинные поля в качестве операндов для данной операции над множествами?

- Почему операция ORDER BY не выполняется удаленно?

  Проверьте:

  - Могут ли входные данные операции ORDER BY вычисляться удаленно? Если ответ отрицательный, проверьте входные данные.
  - Содержит условие ORDER BY символьное выражение? Если да, совпадают ли последовательности упорядочивания на удаленном источнике данных и на системе объединения?
  - Налагает ли источник данных ограничения на эту операцию? Например, нет ли ограничения числа элементов ORDER BY? Ограничивает ли источник данных спецификацию столбцов для списка ORDER BY?

Обновления и настройки источников данных: Хотя компилятор SQL в DB2 знает многое о поддержке SQL источниками данных, эти данные могут со временем

потребовать коррекции, поскольку источники данных могут обновляться и/или настраиваться. В подобных случаях оповестите об этом DB2, изменив информацию в локальном каталоге. Используйте операторы DB2 DDL (такие как CREATE FUNCTION MAPPING и ALTER SERVER) для изменения каталога. Дополнительную информацию смотрите в справочнике *SQL Reference*.

Генерация удаленного SQL и глобальная оптимизация

Эта стадия помогает выработать глобально оптимальную стратегию доступа для оценки запроса. Для базы данных объединения стратегия доступа может включать разбиение оригинального запроса на набор единиц удаленного запроса с последующим объединением результатов.

Используя в качестве рекомендации выходные данные изменение уровня анализа, оптимизатор решает, будет ли каждая операция выполняться локально в DB2 или удаленно на источнике данных. Это решение основывается на результатах оценки ее модели затрат, включающих не только затраты на оценку операции, но и затраты на пересылку данных или сообщений между DB2 и источниками данных.

Цель - выработка оптимизированного запроса, однако на выходные данные глобальной оптимизации, а следовательно, и на производительность запроса могут повлиять многие факторы. Ключевые обсуждаемые факторы делятся на две группы: характеристики сервера и характеристики псевдонима.

Характеристики и опции сервера, влияющие на глобальную оптимизацию

Факторы сервера источника данных, способные повлиять на глобальную оптимизацию, включают:

- Сравнительное отношение скорости процессоров

  Используйте опцию сервера *cpu_ratio* для задания, насколько быстрее или медленнее процессор источника данных по сравнению с процессором DB2. Низкое отношение указывает, что процессор рабочей станции источника данных быстрее, чем процессор рабочей станции DB2. В таком случае оптимизатор DB2 с большей вероятностью передаст операции с большой нагрузкой на источник данных. Дополнительную информацию об этом показателе смотрите в разделе “Опции сервера, влияющие на запросы базы данных объединения” на стр. 111.

- Сравнительное отношение скоростей ввода/вывода

  Опция сервера *io_ratio* показывает, насколько быстрее или медленнее ввод/вывода в системе источника данных по сравнению с системой DB2. Низкое отношение указывает, что скорость ввода/вывода рабочей станции источника данных больше, чем у рабочей станции DB2. При низких отношениях оптимизатор DB2 скорее передаст операции с большой нагрузкой на системы ввода/вывода на источник данных. Дополнительную информацию об этом показателе смотрите в разделе “Опции сервера, влияющие на запросы базы данных объединения” на стр. 111.

- Скорость связи между DB2 и источником данных

  Опция сервера *comm_rate* задает пропускную способность сети. Низкие показатели (указывающие на медленную связь между DB2 и источником данных) побудят оптимизатор DB2 сократить число сообщений на этот источник данных и от него. Если данный показатель имеет значение 0, оптимизатор выберет вариант запроса, требующий минимального сетевого трафика. Дополнительную информацию об этом показателе смотрите в разделе “Опции сервера, влияющие на запросы базы данных объединения” на стр. 111.

- Последовательность упорядочивания для источника данных

  Используйте опцию сервера *collating_sequence* для указания, соответствуют ли последовательности упорядочивания на источнике данных и в локальной базе данных DB2. Если значение этой опции - не ’Y’, оптимизатор будет считать данные, возвращаемые этим источником, неупорядоченными. Дополнительную информацию о влиянии последовательностей упорядочивания на производительность смотрите в разделе “Последовательность упорядочивания” на стр. 211.

- Советы по удаленным планам

  Используйте опцию сервера *plan_hints* для указания, поддерживает ли источник данных советы по планам. Советы по плану - это фрагменты оператора, передающие дополнительную информацию оптимизаторам источников данных. Эта информация может улучшить производительность для определенных типов запросов. Советы по плану помогают оптимизатору источника данных решить, использовать индекс или нет, какой индекс использовать или какую использовать последовательность объединения таблиц.

  Если советы по планам разрешены, посылаемый источнику данных запрос содержит дополнительную информацию. Например, так может выглядеть оператор с советами по плану, посылаемый оптимизатору Oracle:

  ```
  SELECT /*+ INDEX (table1, t1index)*/
    col1
    FROM table1
  ```

  Совет по плану - строка `/*+ INDEX (table1, t1index)*/`.

- Информация в базе знаний оптимизатора DB2

  DB2 содержит базу знаний оптимизатора, содержащую данные о собственных источниках данных. Оптимизатор DB2 не генерирует планы удаленного доступа, которые не могут генерироваться конкретными СУБД. Другими словами, DB2 избегает генерировать планы, которые оптимизаторы на удаленных источниках данных не могут понять или принять.

Характеристики псевдонимов, влияющие на глобальную оптимизацию

Следующие разделы содержат специфические для псевдонимов факторы, способные повлиять на глобальную оптимизацию.

Особенности индексов: DB2 может использовать информацию об индексах на источниках данных для оптимизации запросов. По этой причине важно, чтобы доступная DB2 информация об индексах была актуальной. Информация об индексах для псевдонимов первоначально получается во время создании псевдонима. Информация об индексах не собирается для псевдонимов производных таблиц.

Создание спецификаций индексов на псевдонимах: Можно создать спецификацию индекса для псевдонима. Спецификации индексов строят определение индекса (но не сам индекс) в каталоге для использования оптимизатором DB2. Используйте для создания спецификаций индексов оператор CREATE INDEX SPECIFICATION ONLY. Синтаксис создания спецификации индекса на псевдониме подобен синтаксису создания индекса на локальной таблице. Дополнительную информацию смотрите в книге *Administration Guide: Planning*.

Рассмотрите создание спецификаций индексов, если:

- При создании псевдонима DB2 не может получить информацию об индексах из источника данных.
- Нужен индекс для псевдонима производной таблицы.
- Вы хотите заставить оптимизатор DB2 использовать специфический псевдоним как внутреннюю таблицу объединения с вложенным циклом. Пользователь может создать индекс по объединяемому столбцу, если его еще не существует.

Оцените свои потребности перед выполнением операторов CREATE INDEX над псевдонимом для производной таблицы. В случае, когда производная таблица определяется простым условием SELECT на таблице с индексом, создание индексов на псевдониме (локально), соответствующих индексам на таблице на источнике данных, может значительно повысить производительность запроса. Однако если индексы создаются локально на производных таблицах, не определяемых простыми операторами выбора (например, производная таблица, созданная объединением двух таблиц), производительность запроса может пострадать. Например, если индекс создается на производной таблице, являющейся объединением двух таблиц, оптимизатор может выбрать эту производную таблицу внутренним элементом объединения с вложенным циклом. Запрос будет иметь низкую производительность, потому что объединение будет оцениваться несколько раз. Альтернатива - создать псевдонимы для каждой таблицы, на которую ссылается производная таблица в источнике данных, и создать локальную производную таблицу в DB2, которая ссылается на оба псевдонима.

Особенности статистики каталога: Статистика каталога описывают общий размер псевдонимов и интервал значений в связанных столбцах. Они используются оптимизатором при вычислении наименее затратного способа

обработки запросов, содержащих псевдонимы. Статистика псевдонимов хранится в той же производной таблице каталога, что и статистика таблиц. Дополнительную информацию о типах статистики и их локальном обновлении смотрите в разделах “Глава 5. Статистика системного каталога” на стр. 117 и “Правила изменения статистики таблиц и псевдонимов” на стр. 142.

DB2 может получать статистические данные, хранимые на источнике данных, но она не может автоматически обнаруживать изменения существующих статистических данных на источниках данных. Более того, DB2 не имеет механизма обработки определения объекта или структурных изменений (добавления столбца) объектов на источниках данных. Если статистические данные или структурные данные объекта изменены, у вас есть два выбора:

- Выполнить эквивалент RUNSTATS на источнике данных. Затем отбросить текущий псевдоним и повторно создать псевдоним. Используйте этот подход, если изменилась информация о структуре.
- Обновите вручную статистику в производной таблице SYSSTAT.TABLES. Этот подход требует меньше этапов, но он не поможет, если изменилась информация о структуре.

Анализ и понимание решений глобальной оптимизации

В этом разделе описаны инструменты для анализа оптимизации запросов и общие вопросы (а также предлагаемые области исследования), связанные с оптимизацией запросов.

Анализ оптимизации запросов: В DB2 есть две утилиты, показывающие глобальные планы доступа:

- Наглядное объяснение Visual explain. Запускается командой **db2cc** или **db2vexp**. Используйте его для просмотра графа плана доступа для запроса. Место выполнения для каждого оператора включается в подробный показ оператора. Вы можете также посмотреть удаленный оператор SQL, генерируемый для каждого источника данных, в операторе RQUERY (операция выбора). Исследуя подробности каждого оператора, можно увидеть число строк, оцениваемых оптимизатором DB2 в качестве входных и выходных данных каждого оператора. Можно также увидеть оценку затрат на выполнение каждого оператора, включая затраты на связь. Дополнительную информацию смотрите в разделе “Приложение C. Инструменты объяснения SQL” на стр. 591.
- Объяснение SQL. Запускается командой db2expln или dynexpln. Используйте его для просмотра стратегии плана доступа в виде текста. Объяснение SQL не дает информации о затратах, однако вы можете получить генерируемый удаленным оптимизатором план доступа для источников данных, где поддерживается удаленная функция объяснения. Дополнительную информацию смотрите в разделе “Приложение C. Инструменты объяснения SQL” на стр. 591.

Понимание решений оптимизации DB2: В этом разделе перечислены вопросы оптимизации и ключевые области исследования для улучшения производительности. Ключевые вопросы:

- Почему объединение двух псевдонимов одного и того же источника данных не выполняется удаленно?

  Области для исследования включают:

  – Операции объединения. Может ли источник данных поддерживать их?

  – Предикаты объединения. Может ли предикат объединения обрабатываться на удаленном источнике данных? Если ответ отрицателен, проверьте предикаты объединения. Дополнительную информацию смотрите в разделе “Как понять, почему запрос выполняется на источнике данных или в системе объединения” на стр. 216.

  – Число строк в результате объединения (его дает наглядное объяснение). Возможно, в объединении гораздо больше строк, чем в двух псевдонимах, вместе взятых? Правдоподобны ли эти числа? Если ответ отрицателен, рассмотрите возможность обновления статистик псевдонимов вручную (SYSSTAT.TABLES).

- Почему оператор GROUP BY не обрабатывается удаленно?

  Области для исследования включают:

  – Синтаксис операции. Убедитесь, что операция может обрабатываться на удаленном источнике данных.Дополнительную информацию смотрите в разделе “Как понять, почему запрос выполняется на источнике данных или в системе объединения” на стр. 216.

  – Число строк. Проверьте оценки числа строк в входных и выходных данных оператора GROUP BY, используя визуальное объяснение. Не близки ли эти два числа? Если ответ положительный, оптимизатор DB2 посчитает более эффективной локальную обработку этого условия GROUP BY. Правдоподобны ли эти два числа? Если ответ отрицателен, рассмотрите возможность обновления статистик псевдонимов вручную (SYSSTAT.TABLES).

- Почему оператор не полностью выполняется на удаленном источнике данных?

  Оптимизатор DB2 выполняет оптимизацию на основе затрат. Даже если анализ изменения уровня показывает, что каждый оператор может выполняться на удаленном источнике данных, оптимизатор выбирает решение на основе оценки затрат для генерации глобально оптимального плана. В этот план может вносить вклад огромное число факторов. Например, даже если удаленный источник данных может обработать каждую операцию исходного запроса, скорость его процессора может быть меньше скорости процессора DB2, и это может сделать более выгодным выполнение операций в DB2. Если результаты неудовлетворительны, проверьте статистику вашего сервера в SYSCAT.SERVEROPTIONS.

- Почему план, сгенерированный оптимизатором и полностью выполняемый на удаленном источнике данных, имеет гораздо худшую производительность, чем исходный запрос, выполняемый непосредственно на удаленном источнике данных?

  Области для исследования включают:

  – Удаленный оператор SQL, генерируемый оптимизатором DB2. Убедитесь в его идентичности исходному запросу. Сделайте проверку изменения порядка предикатов. Хороший оптимизатор запросов не должен быть чувствителен к порядку предикатов в запросе; однако не все оптимизаторы СУБД идентичны, а следовательно, есть вероятность, что оптимизатор удаленного источника данных может сгенерировать другой план для другого порядка входных предикатов. Если это так, то эта проблема присуща удаленному оптимизатору. Обдумайте модификацию порядка предикатов исходных данных для DB2, либо возможность обращения за содействием в сервисную организацию удаленного источника данных.

    Попробуйте также замену предикатов. Хороший оптимизатор запросов не должен быть чувствителен к эквивалентным заменам предикатов; однако не все оптимизаторы СУБД идентичны, а следовательно, возможно, что оптимизатор удаленного источника данных может сгенерировать другой план на основе входного предиката. Например, некоторые оптимизаторы не могут генерировать операторы транзитивного замыкания для предикатов.

  – Число возвращаемых строк. Это число можно получить при помощи наглядного объяснения (Visual Explain). Если запрос возвращает много строк, сетевой трафик в потенциале является узким местом.

  – Дополнительные функции. Содержит ли удаленный оператор SQL функции, добавочные по сравнению с исходным запросом? Некоторые дополнительные функции могут генерироваться для преобразования типов данных. Проверьте, необходимо ли оно.

Глава 7. Возможность объяснения SQL

Возможность объяснения SQL - это часть компилятора SQL, с помощью которой можно захватить информацию о среде, в которой скомпилирован статический или динамический оператор SQL. Эта информация позволит вам понять структуру и потенциальную производительность операторов SQL, в том числе:

- Последовательность операций по обработке запроса
- Информацию о стоимости
- Предикаты и оценки избирательности
- Статистику для всех объектов, к которым обращается оператор SQL во время работы объяснения.

Эта информация может помочь:

- Понять план выполнения, выбранный для запроса
- Создать прикладную программу
- Определить, когда следует пересвязать программу
- Спроектировать базу данных.

Далее объясняются следующие темы:

Результат объяснения сохраняется в реляционных таблицах и, по вашему желанию, в формате, который может быть отображен графически с помощью инструмента наглядного объяснения - Visual Explain. К этим таблицам объяснения можно применять запросы, чтобы извлечь интересующую вас информацию. Дополнительную информацию о таблицах, используемых при объяснении, и о том, как создавать эти таблицы, смотрите в разделе “Приложение B. Таблицы объяснения и их определения” на стр. 557.

Выбор инструмента объяснения

Из СУБД подобного класса DB2 предоставляет наиболее исчерпывающую утилиту объяснения SQL; она выдает подробную информацию оптимизатора о плане доступа, выбранном для объясненного оператора SQL. Чтобы обеспечить необходимую гибкость при захвате и доступе к информации объяснения, предлагается несколько методов.

Подробная информация оптимизатора, позволяющая производить углубленный анализ плана доступа, сохраняется в таблицах объяснения отдельно от самого плана доступа. Информацию из таблиц объяснения можно извлечь тремя способами:

1. Написать свои собственные запросы (используя описания таблицы объяснения, как показано в разделе “Приложение B. Таблицы объяснения и их определения” на стр. 557)
2. Использовать инструмент *db2exfmt*
3. Использовать инструмент Visual Explain (для просмотра информации снимка объяснения)

Таблицы объяснения доступны на всех поддерживаемых платформах и содержат информацию как о статических, так и о динамических операторах SQL. Обращаться к таблицам объяснения можно с помощью операторов SQL; этот способ позволяет легко манипулировать выходными данными и сравнивать различные запросы или один и тот же запрос в разные моменты времени. Если вы хотите, чтобы информация из таблиц объяснения была представлена в заранее заданном формате, можете использовать инструмент *db2exfmt*. Более подробная информация об этом инструменте приводится в разделе “Приложение D. db2exfmt - Инструмент форматирования таблицы объяснения” на стр. 637. Другой вариант - создать свои собственные операторы для доступа к этим таблицам.

Примечание: Этот инструмент (и другие, такие как *db2batch*, *dynexpln*, *db2vexp* и *db2_all*) находится в подкаталоге *misc* каталога *sqllib*. Если инструмент был перенесен из этого каталога, приведенный выше вызов из командной строки может не работать.

Инструмент наглядного объяснения (Visual Explain) позволяет анализировать план доступа и информацию оптимизатора из таблиц объяснения с помощью графического интерфейса. С помощью этого инструмента можно анализировать как статические, так и динамические операторы SQL. Обычно Visual Explain вызывается из Центра управления. Центр управления можно вызвать из командной строки, введя *db2cc*. Для отдельного оператора SQL Visual Explain можно вызвать и непосредственно из командной строки с помощью команды *db2vexp*. На некоторых платформах Visual Explain можно вызвать через папку, содержащуюся в папке DB2 Universal Database. Инструмент Visual Explain доступен не во всех поддерживаемых платформах. Чтобы узнать,

поддерживается ли у вас Visual Explain, посмотрите руководство *Быстрый старт* для вашей платформы. С помощью Visual Explain можно просматривать снимки, снятые на другой платформе. Например, клиент Windows NT может просматривать снимки, сгенерированные на сервере DB2 for HP-UX. Чтобы это было возможно, на обеих платформах должна использоваться Версия 5 или более новая. Выходные данные Visual Explain не просто использовать для дальнейшего анализа и нельзя передать другим программам. Чтобы получить дополнительную информацию о команде *db2vexp*, введите в командной строке *db2vexp -h* или посмотрите руководство *Command Reference*. Другую информацию о Visual Explain смотрите в электронной справке Центра управления, который вызывается командой *db2cc*.

Информация о планах доступа для статических операторов SQL генерируется и хранится в системном каталоге как часть пакета. Для просмотра имеющейся информации о плане доступа для одного или нескольких пакетов можно использовать инструмент *db2expln*, вызываемый из командной строки. Инструмент *db2expln* показывает фактическую реализацию выбранного плана доступа. Он не показывает информацию оптимизатора.

Инструмент *dynexpln* (который вызывает *db2expln*) позволяет быстро объяснять динамические операторы SQL, не содержащие маркеры параметров. Для вызовов *db2expln* из *dynexpln* входной оператор SQL преобразуется в статический оператор внутри псевдопакета. Из-за этого информация иногда может оказаться не совсем точной. Если требуется абсолютная точность, следует использовать возможность объяснения, как описано в разделе “Использование возможности объяснения SQL” на стр. 228.

Инструмент *db2expln* дает вам относительно компактный и удобочитаемый обзор операций, которые будут происходить во время выполнения; для этого изучается фактически сгенерированный план доступа (дополнительную информацию о том, как генерируется код, смотрите в разделе 158). Подробности о использование *db2expln* и интерпретации выходных данных можно найти в разделе “Приложение C. Инструменты объяснения SQL” на стр. 591.

В Табл. 16 описаны различные инструменты, доступные в утилите объяснения DB2, и их индивидуальные характеристики. Используйте эту таблицу, чтобы выбрать инструмент, наиболее подходящий для вашей среды и вашей задачи.

Таблица 16. Инструменты объяснения

Требуемые характеристики	**Visual Explain**	**db2vexp**	**Таблицы объяснения**	**db2exfmt**	**db2expln**	**dynexpln**
Графический интерфейс пользователя	Да	Да				
Текстовый вывод				Да	Да	Да

Таблица 16. Инструменты объяснения (продолжение)

Требуемые характеристики	Visual Explain	db2vexp	Таблицы объяснения	db2exfmt	db2expln	dynexpln
“Быстрый упрощенный” анализ статического SQL					Да	
Поддержка статического SQL	Да		Да	Да	Да	
Поддержка динамического SQL	Да	Да	Да	Да		Да*
Поддержка прикладных программ CLI	Да		Да	Да		
Доступность для реквестеров прикладных программ DRDA			Да			
Подробная информация оптимизатора	Да	Да	Да	Да		
Пригодность для анализа нескольких операторов			Да	Да	Да	Да
Доступность информации из прикладной программы			Да			

Примечание:

***** Косвенно, с использованием db2expln; есть некоторые ограничения.

Использование возможности объяснения SQL

Различные способы захвата информации объяснения используют:

1. опции EXPLAIN и EXPLSNAP BIND/PREP
2. специальные регистры CURRENT EXPLAIN MODE и CURRENT EXPLAIN SNAPSHOT
3. оператор SQL EXPLAIN
4. инструмент *db2vexp* (для отображения информации непосредственно вызывается также Visual Explain)

Есть три задачи, для решения которых вам может понадобиться собирать и использовать данные объяснения:

1. Для понимания шагов (плана доступа), которые должен выполнить Менеджер баз данных, чтобы удовлетворить ваш запрос. Раздел “Концепции доступа к данным и оптимизация” на стр. 169 содержит информацию, с которой вам, возможно, надо ознакомиться, чтобы понять выходные данные объяснения.
2. Чтобы оценить эффективность ваших действий по настройке производительности. Есть некоторые действия, которые вы можете

предпринять для повышения производительность ваших запросов. Многие из этих действий описаны в подтемах следующих разделов:

- “Глава 3. Особенности прикладного программирования” на стр. 43
- “Глава 4. Факторы среды” на стр. 95
- “Глава 5. Статистика системного каталога” на стр. 117.

После внесения изменений в любую из этих областей с помощью объяснения SQL вы можете определить, повлияло ли изменение на выбранный план доступа, и если да, то как. Например, если вы добавили индекс, исходя из рекомендаций, данных в “Влияние индексов на оптимизацию запросов” на стр. 102, данные объяснения помогут вам определить, используется ли этот индекс на самом деле так, как вы предполагали.

Хотя выходные данные объяснения позволяют определить, какой план доступа выбран и какова его относительная стоимость, но единственный способ точно измерить изменение производительности запроса - использовать методы измерения производительности, описанные в разделе “Глава 12. Измерение производительности” на стр. 335.

3. Чтобы понять причины изменения производительности запроса, надо иметь информацию объяснения как до, так и после внесения изменений для анализа их влияния. Поэтому при компиляции оператора SQL в базу данных надо:
   - Применить возможность объяснения для захвата информации о плане до внесения изменений и сохранить получившиеся таблицы объяснения или выходные данные `db2exfmt`.
   - Если вы не хотите или не можете обратиться к Visual Explain для просмотра этой информации, сохранить и/или напечатать текущую статистику каталога. (Для этой задачи можно использовать инструмент `db2look`, описанный в разделе “Моделирование производственных баз данных” на стр. 147.)
   - Сохранить и/или напечатать операторы языка определения данных (DDL), включая операторы для CREATE TABLE, CREATE VIEW, CREATE INDEX и CREATE TABLESPACE.

   Эта информация даст вам картину *до* изменения, которую можно использовать как исходную точку для последующего анализа. Для динамических операторов SQL эту информацию можно также получить во время первого выполнения вашей программы. Для статических операторов SQL эту информацию можно получить также во время связывания.

   Когда вы захотите проанализировать причину изменения производительности, можно сравнить данные *до* изменения с информацией о запросе и среде, которую вы соберете, начиная анализ (данные *после* изменения).

Простой пример: ваш анализ может показать, что некоторый индекс больше не используется в качестве части пути доступа. Используя информацию статистики каталога в Visual Explain, вы можете заметить, что число уровней индекса (столбец NLEVELS) стало значительно выше, чем при первом связывание запроса с базой данных. Тогда вы можете выбрать, что сделать:

- Реорганизовать индекс
- Собрать новую статистику для вашей таблицы и индексов
- Собрать информацию объяснения при пересвязывании запроса.

Выполнив эти действия, вы можете заметить, что индекс опять используется в плане доступа и больше нет проблем с производительностью запроса.

Начальные понятия объяснения

Вы можете использовать информацию объяснения для анализа плана доступа, выбранного оптимизатором на основе возможностей, описанных в разделе “Концепции доступа к данным и оптимизация” на стр. 169. Например, информация объяснения может показать, что оптимизатор выбрал некоторый способ просмотра индекса (смотрите раздел “Концепции просмотра индекса” на стр. 170). Кроме того, эта информация может помочь вам определить:

- Сколько столбцов индекса используется в качестве критерия поиска, как описано в разделе “Предикаты ограничения интервала и предикаты с аргументом поиска индекса” на стр. 180
- Используется ли доступ только через индекс, как описано в разделе “Доступ только к индексу” на стр. 175
- Будет ли использоваться предварительная выборка списка для чтения страницы, как описано в разделе “Как работает предварительная выборка списка” на стр. 272.

Другой пример: информация объяснения может помочь понять, как объединены две таблицы, а именно:

- Способ объединения
- Порядок, в котором объединены таблицы
- Использование и типы сортировок.

Объяснение можно использовать для операторов SQL SELECT, SELECT INTO, UPDATE, INSERT, VALUES, VALUES INTO, и DELETE, но основное его назначение - отслеживать пути доступа для подоператоров выбора SELECT ваших операторов.

Чтобы удовлетворить некоторый запрос SQL, менеджер баз данных обычно:

- Использует один или несколько объектов данных (таблицу, индекс или и то и другое)

- Выполняет одну или несколько операций (например, просмотр таблицы, просмотр индекса и объединение)
- Возвращает набор результатов вызывающей программе.

Для простого запроса SQL, такого как:

```
SELECT DEPTNO, DEPTNAME
  FROM DEPARTMENT
```

инструмент Visual Explain может показать следующее графическое представление выполненных шагов:

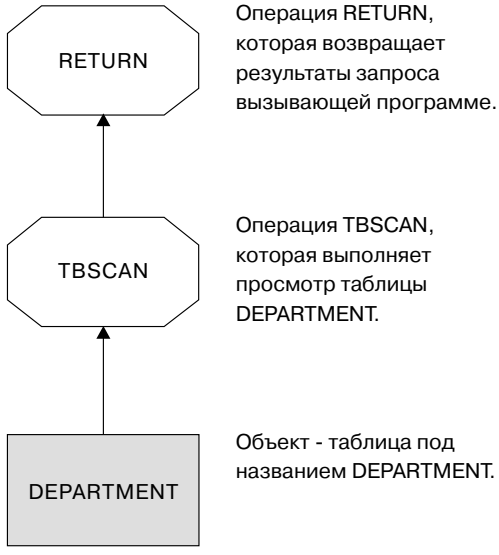


Рисунок 20. Графическое представление результатов объяснения

В следующих темах обсуждается, какие элементы объектов и операций можно просмотреть:

- “Информация объяснения для объектов данных”
- “Информация объяснения для операций данных” на стр. 232

Информация объяснения для объектов данных

В одном плане доступа для выполнения оператора SQL может использоваться один или несколько объектов данных.

Статистика объекта: возможность объяснения записывает об объекте следующую информацию:

- Время создания
- Время последнего сбора статистики для объекта (смотрите раздел “Глава 5. Статистика системного каталога” на стр. 117)

- Информацию о том, упорядочены ли данные в объекте (только для объектов таблиц и индексов)
- Число столбцов в объекте (только для объектов таблиц и индексов)
- Оценку числа строк в объекте (только для объектов таблиц и индексов)
- Количество страниц, занимаемых объектом в пуле буферов
- Оценку полных расходов в миллисекундах на одну произвольную операцию ввода-вывода для табличного пространства, в котором хранится данный объект
- Оценку скорости в миллисекундах чтения страницы размером 4 Кбайта из указанного табличного пространства
- Размеры предварительной выборки и экстента в страницах по 4 Кбайта
- Степень кластеризации данных с индексом
- Число конечных страниц, используемых индексом этого объекта, и количество уровней в дереве
- Число различных значений полного ключа в индексе этого объекта
- Полное число записей переполнения в таблице.

Информация объяснения для операций данных

Для выполнения оператора SQL и возврата результатов в одном плане доступа может производиться несколько операций над данными. Компилятор SQL определяет, какие требуются операции, например операции просмотра таблицы, просмотра индекса, объединения с вложенным циклом или группирования. Подробности о многих из этих операций изложены в разделе “Концепции доступа к данным и оптимизация” на стр. 169.

Кроме вывода различных операций, используемых в плане доступа, приводится также информация объяснения для каждой операции и для совокупных эффектов плана доступа.

Информация о примерной стоимости: для операций могут быть приведены следующие показатели совокупной стоимости. Стоимости даны для выбранного плана доступа до операции, для которой захвачена информация, включительно.

- Полная стоимость (в единицах времени)
- Число страниц ввода-вывода
- Количество команд процессора
- Стоимость выборки первой строки (в единицах времени), включая все необходимые начальные расходы
- Стоимость связи (в кадрах).

Единицы времени - это искусственные, относительные единицы измерения. Единицы времени определяются оптимизатором на основе внутренних значений, таких как статистические показатели, которые меняются в процессе работы с

базой данных. В результате нельзя гарантировать, что изменение в единицах времени для оператора SQL будет постоянным при каждой оценке стоимости.

Свойства операции: эта информация включается в объяснение, чтобы описать свойства каждой операции:

- Набор таблиц, к которым были обращения
- Набор столбцов, к которым были обращения
- Столбцы с упорядоченными данными, если оптимизатор определил, что этот порядок может быть использован в последующих операциях
- Набор примененных предикатов
- Примерное число возвращаемых строк (мощность).

Как организована информация объяснения

Вся информация объяснения построена на основе концепции экземпляров объяснения. Экземпляр объяснения представляет один вызов возможности объяснения для одного или нескольких операторов SQL. Экземпляр объяснения дает информацию объяснения для:

- Всех подходящих операторов SQL в одном пакете - для **статических** операторов SQL
- Одного определенного оператора SQL - для инкрементно связанных операторов SQL
- Одного определенного оператора SQL - для **динамических** операторов SQL
- Для каждого оператора EXPLAIN SQL (динамического или статического).

Информация объяснения, захваченная в одном экземпляре объяснения, описывает как план доступа, выбранный для выполнения скомпилированного оператора SQL, так и среду компиляции SQL. Информация объяснения организована в 3 подгруппы:

Информация экземпляра объяснения
: Информация о среде компиляции, захваченная для каждого экземпляра объяснения.

Информация снимка объяснения
: Информация, используемая Visual Explain.

Информация таблицы объяснения
: Информация, собираемая, если затребована информация таблицы объяснения.

Информация экземпляра объяснения

Информация экземпляра объяснения хранится в таблице EXPLAIN_INSTANCE. Дополнительная конкретная информация о каждом операторе SQL, объясненном в экземпляре объяснения, хранится в таблице EXPLAIN_STATEMENT.

Идентификация экземпляра объяснения: Однозначно идентифицировать каждый экземпляр объяснения и сопоставить информацию об операторах SQL при данном вызове утилиты можно по следующим данным:

- Пользователь, затребовавший информацию объяснения
- Момент начала требования на объяснение
- Имя пакета, из которого пришел объясненный оператор SQL
- Схема пакета, из которого пришел объясненный оператор SQL.
- Информацию, был ли затребован снимок как часть объяснения.

Параметры среды: захватывается информация о среде, касающаяся оптимизации запросов компилятором SQL. Информация о среде включает в себя:

- Номер версии и выпуска для используемого уровня DB2.
- Степень параллелизма, используемую при компиляции запроса. Для определения степени параллелизма можно использовать специальный регистр CURRENT DEGREE, опция связывания DEGREE, параметры конфигурации SET RUNTIME DEGREE API и параметр конфигурации *dft_degree*.
- Являлся оператор SQL динамическим или статическим.
- Класс оптимизации запроса, использованный при компиляции запроса. Дополнительную информацию смотрите в разделе “Настройка класса оптимизации” на стр. 70.
- Тип блокировки указателя, указанный при компиляции запроса. Дополнительную информацию о указателях смотрите в руководстве *SQL Reference*. Дополнительную информацию о блокировке указателей смотрите в разделе “Блокирование строк” на стр. 83.
- Уровень изоляции, использованный при компиляции запроса. Дополнительную информацию смотрите в разделе “Одновременность” на стр. 43.
- Значения различных параметров конфигурации в момент компиляции запроса. Дополнительную информацию о параметрах конфигурации, которые могут влиять на оптимизацию запроса, смотрите в разделе “Параметры конфигурации, влияющие на оптимизацию запросов” на стр. 95; это, в частности, следующие параметры, записываемые при снятии снимка объяснения:
  - “Размер пула буферов (buffpage)” на стр. 366
  - “Размер кучи сортировки (sortheap)” на стр. 381
  - “Среднее число активных программ (avg_appls)” на стр. 420
  - “Куча базы данных (dbheap)” на стр. 369
  - “Максимальная память для списка блокировок (locklist)” на стр. 375
  - “Максимальный процент списка блокировок перед расширением (maxlocks)” на стр. 407
  - “Скорость процессора (cpuspeed)” на стр. 501

– “Пропускная способность связи (comm_bandwidth)” на стр. 500.

Идентификация оператора SQL: в одном экземпляре объяснения может быть объяснено несколько операторов SQL. Вместе с информацией, однозначно идентифицирующей экземпляр объяснения, идентифицировать каждый отдельный оператор SQL помогает следующая информация.

- Тип оператора: SELECT, DELETE, INSERT, UPDATE, позиционированный DELETE, позиционированный UPDATE.
- Номера оператора и раздела для пакета, выдавшего оператор SQL, как записано в производной таблице каталога SYSCAT.STATEMENTS.

Поля QUERYTAG и QUERYNO в таблице EXPLAIN_STATEMENT содержат идентификаторы и задаются в процессе объяснения для вас.

При динамическом объяснении для операторов SQL, переданных во время сеанса CLP или CLI, когда включен режим EXPLAIN MODE или EXPLAIN SNAPSHOT, для поля QUERYTAG задается значение “CLP” или “CLI”. Когда это происходит, в поле QUERYNO по умолчанию устанавливается число, которое с каждым оператором увеличивается на один или на несколько.

Для других динамически объясняемых операторов SQL (не от CLP, CLI, или использующих оператор SQL EXPLAIN) поле QUERYTAG остается пустым, а в поле QUERYNO всегда будет “1”.

Оценка стоимости: Для каждого объясненного оператора записывается оценка относительной стоимости выполнения выбранного плана доступа. Стоимость дается в искусственных относительных единицах измерения, называемых *единицами времени*. Оценка затрачиваемого времени **не** дается по следующим причинам:

- Оптимизатор SQL оценивает потребление ресурсов, а не затрачиваемое время.
- Оптимизатор не моделирует всех факторов, которые могут повлиять на затраты времени, он игнорирует те факторы, которые не влияют на эффективность плана доступа. На затрачиваемое время **влияют** некоторые факторы времени выполнения, в частности: рабочая нагрузка системы, конкуренция за ресурсы, количество параллельных обработок и операций ввода-вывода, стоимость возвращения строк пользователю и время соединения между клиентом и сервером.

Текст оператора: для каждого объясненного оператора записывается два варианта текста оператора SQL. Один вариант - это текст, полученный компилятором SQL. Другой вариант - это текст оператора, полученный обратным переводом из внутреннего представления запроса в компиляторе. Этот перевод, хотя и выглядит похоже на другие операторы SQL, **не** обязательно следует корректному синтаксису SQL и не обязательно отражает реальное содержимое внутреннего представления как целого. Этот перевод дается, чтобы

вы могли понять контекст SQL, исходя из которого, оптимизатор выбирает план доступа. Сравнение написанного пользователем текста оператора с внутренним представлением оператора SQL может помочь понять, как компилятор SQL переписал запрос для лучшей оптимизации. (Смотрите раздел “Перезапись запросов компилятором SQL” на стр. 159.) В обратном переводе приводятся также другие элементы среды, влияющие на ваш оператор, такие как триггеры и ограничения. Вот некоторые ключевые слова, используемые в этом “оптимизированном” тексте:

$Cn	Имя производного столбца, где n - некоторое целое число.
$CONSTRAINT$	Тэг, используемый для обозначения имени ограничения, добавленного к исходному оператору SQL во время компиляции. Появляется в сочетании с префиксом $WITH_CONTEXT$.
$DERIVED.Tn	Имя производной таблицы, где n - это некоторое целое число.
$INTERNAL_FUNC$	Тэг, показывающий наличие функций, примененных компилятором SQL для объясняемого запроса, но не доступных для общего использования.
$INTERNAL_PRED$	Тэг, показывающий наличие предикатов, добавленных компилятором SQL во время компиляции объясняемого запроса. Эти предикаты также не доступны для общего использования. Внутренний предикат используется компилятором, чтобы отразить дополнительный контекст, добавленный в исходный оператор SQL в результате триггеров и ограничений.
RID	Тэг, используемый для идентификации столбца идентификатора строки (RID) для данной строки.
$TRIGGER$	Тэг, используемый для обозначения имени триггера, добавленного к исходному оператору SQL во время компиляции. Появляется в сочетании с префиксом $WITH_CONTEXT$.
$WITH_CONTEXT$(...)	Этот тэг появляется в начале текста, когда в исходный оператор SQL добавляются дополнительные триггеры или ограничения. За этим префиксом появится список имен всех

триггеров и ограничений, влияющих на компиляцию и разрешение оператора SQL.

Информация снимка объяснения

Когда затребован снимок объяснения, записывается дополнительная информация объяснения, описывающая план доступа, который выбран оптимизатором SQL. Эта информация хранится в таблице EXPLAIN_STATEMENT в столбце SNAPSHOT в формате, требуемом инструментом Visual Explain. Этот формат непригоден для других программ.

Дополнительная информация о содержимом информации снимка объяснения доступна в Visual Explain, а также в разделах:

- “Информация объяснения для объектов данных” на стр. 231
- “Информация объяснения для операций данных” на стр. 232.

Информация таблицы объяснения

Когда затребована информация таблицы объяснения, записывается дополнительная информация, описывающая план доступа, выбранный оптимизатором SQL. Эта информация хранится в следующих таблицах объяснения:

- EXPLAIN_ARGUMENT. Эта таблица представляет индивидуальные характеристики каждого отдельного оператора, если они есть.
- EXPLAIN_INSTANCE. Это - главная таблица управления для всей информации объяснения. Каждая строка данных в таблицах объяснения явно связана с одной определенной строкой в этой таблице. В этой таблице хранится основная информация о источнике объясняемых операторов SQL и о среде.
- EXPLAIN_OBJECT. В этой таблице показано, какие объекты данных требуются плану доступа, сгенерированному для данного оператора SQL.
- EXPLAIN_OPERATOR. В этой таблице содержатся все операции, нужные компилятору SQL для оператора SQL.
- EXPLAIN_PREDICATE. В этой таблице показано, какие предикаты применяются данной операцией.
- EXPLAIN_STATEMENT. Эта таблица содержит текст оператора SQL, как он используется для различных уровней информации объяснения. В этой таблице сохраняется исходный оператор SQL, как он был введен пользователем, а также его версия, использованная оптимизатором для выбора плана доступа для этого оператора SQL.
- EXPLAIN_STREAM. Эта таблица представляет входные и выходные потоки данных между отдельными операциями и объектами данных. Сами объекты данных описаны в таблице EXPLAIN_OBJECT. Операции, работающие с потоком данных, описаны в таблице EXPLAIN_OPERATOR.
- ADVISE_WORKLOAD. Эта таблица позволяет пользователям описывать их рабочую нагрузку для базы данных. Каждая строка в рабочей нагрузке

представляет оператор SQL и содержит связанную с ним частоту. Инструмент db2advis и мастер по индексам используют эту таблицу для сбора и хранения рабочей нагрузки и информации.

- ADVISE_INDEX. В этой таблице хранится информация о рекомендуемых индексах. Эта таблица заполняется компилятором SQL, утилитой db2advis, мастером по индексам или пользователем. Она используется в двух целях:
  - Для получения рекомендуемых индексов.
  - Для вычисления индексов на основе данных о предложенных индексах.

Все вышеперечисленные таблицы не создаются по умолчанию. Их можно создать, выполнив сценарий EXPLAIN.DDL, находящийся в подкаталоге misc подкаталога sqllib. Соединитесь с базой данных, для которой нужны таблицы объяснения и советов. Затем введите команду db2 -tf EXPLAIN.DDL, и эти таблицы будут созданы. Если нужно, эти таблицы может также автоматически создать мастер по индексам.

Каждый прямоугольный узел (*объект*) в Наглядном объяснение соответствует строке в таблице EXPLAIN_OBJECT. Каждый восьмиугольный узел (“операция”) в Наглядном объяснение соответствует строке в таблице EXPLAIN_OPERATOR. Каждая связь между операциями или объектами операций соответствует строке в таблице EXPLAIN_STREAM.

Информация таблицы объяснения по содержанию аналогична записываемой для снимка объяснения, но в таблице объяснения информация хранится в обычных реляционных таблицах, к которым можно обращаться с помощью стандартных операторов SQL.

Как и диаграмма плана доступа Наглядного объяснения, таблицы объяснения устроены так, чтобы отражать взаимоотношения между операциями и объектами данных в данном плане доступа. Следующая схема показывает взаимосвязь между этими таблицами.

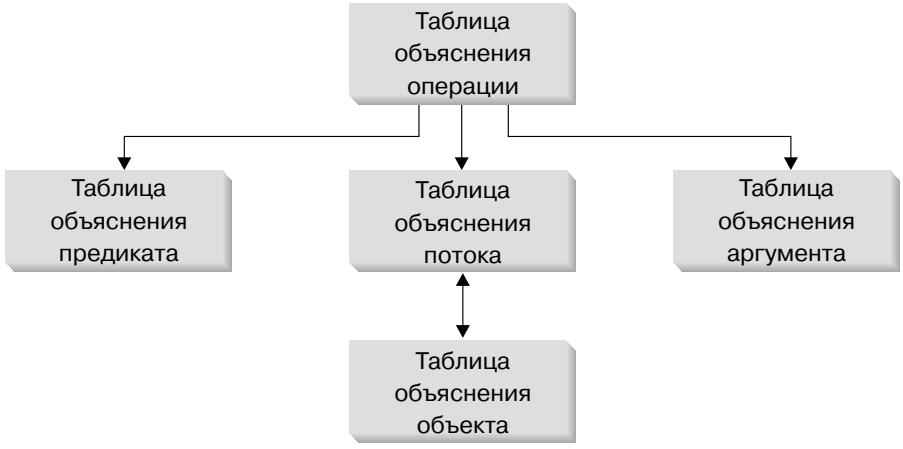


Рисунок 21. Общий вид взаимосвязей между таблицами объяснения (показаны не все таблицы).

Таблицы объяснения могут быть общими для нескольких пользователей. Можно определить таблицы объяснения для одного пользователя, а затем для остальных пользователей задать алиасы с тем же именем, указывающие на определенные таблицы. Каждый пользователь, имеющий в совместном пользовании общие таблицы объяснения, должен иметь разрешение на вставку в эти таблицы.

Дополнительную информацию о таблицах объяснения и способах их создания смотрите в разделе “Приложение C. Инструменты объяснения SQL” на стр. 591. Дополнительную информацию о содержимом информации таблицы объяснения смотрите в разделах:

- “Информация объяснения для объектов данных” на стр. 231
- “Информация объяснения для операций данных” на стр. 232.

Инструмент `db2exfmt`, находящийся в подкаталоге `misc` каталога `sqllib`, позволяет привести содержимое таблиц объяснения к удобному, организованному виду.

Получение данных объяснения

Прежде чем вы сможете получить данные объяснения для оператора SQL, у вас должен быть набор таблиц объяснения, определенных в той же схеме, что и ID авторизации вызывающего утилиту объяснения. Информацию о создании этих таблиц смотрите в разделе “Определения таблиц для таблиц объяснения” на стр. 580.

Захват информации таблицы объяснения

Когда эти таблицы определены, данные объяснения захватываются, если оператор SQL скомпилирован и были затребованы данные объяснения:

- Для **статических** или операторов с **инкрементным связыванием** информация таблицы объяснения захватывается, если указаны опции EXPLAIN ALL или EXPLAIN YES в командах BIND или PREP, или в исходной программе используется статический оператор SQL EXPLAIN.

  Примечание: Когда операторы SQL с инкрементным связыванием компилируются во время выполнения, в таблицы объяснения они помещаются тоже во время выполнения, а не во время связывания. Для вставки в таблицы объяснения используется спецификатор таблицы объяснения и ID авторизации владельца пакета, а не пользователя, выполняющего пакет.

- Для **динамических** операторов SQL информация таблицы объяснения захватывается в следующих ситуациях:
  - Оператор SQL EXPLAIN. Вся информация объяснения захватывается и помещается в таблицы объяснения, если не используется условие FOR SNAPSHOT.

    Пример оператора SQL EXPLAIN:

    ```
    EXPLAIN PLAN FOR <любой допустимый оператор SQL DELETE, INSERT, SELECT,
    SELECT INTO, UPDATE, VALUES, или VALUES INTO>
    ```

  - Специальный регистр CURRENT EXPLAIN MODE имеет значение YES. Это значение указывает компилятору SQL захватить данные объяснения, и позволяет выполнить оператор SQL с возвратом результатов запроса.
  - Специальный регистр CURRENT EXPLAIN MODE имеет значение EXPLAIN. Это значение указывает компилятору SQL захватить данные объяснения, но не выполнять оператор SQL.
  - Специальный регистр CURRENT EXPLAIN MODE имеет значение RECOMMEND INDEXES. Это значение указывает компилятору SQL захватить данные объяснения и разместить рекомендуемые индексы в таблице ADVISE_INDEX, оператор SQL не выполняется.
  - Специальный регистр CURRENT EXPLAIN MODE имеет значение EVALUATE INDEXES. Эта установка указывает компилятору SQL использовать индексы, помещенные пользователем в таблицу ADVISE_INDEX. Пользователь вставляет новую строку для каждого индекса, который надо оценить. Информация, требуемая для каждого индекса: имя индекса, имя таблицы и имена столбцов, составляющих оцениваемый индекс. Как уже говорилось, специальный регистр CURRENT EXPLAIN MODE должен иметь значение EVALUATE INDEXES. Тогда компилятор SQL находит в таблице ADVISE_INDEX индексы, у которых поле USE_INDEX имеет значение “Y” (они называются виртуальными индексами). Все динамические операторы, выполняемые в режиме EVALUATE INDEXES, объясняются так, как если бы эти виртуальные

индексы были доступны. Если виртуальные индексы улучшают производительность операторов, компилятор SQL потом выбирает использование этих индексов. Иначе эти индексы игнорируются. Просматривая результаты EXPLAIN, вы можете увидеть, использовал ли компилятор SQL индексы, предложенные пользователем. Следует рассмотреть возможность применения индексов, использованных компилятором, для улучшения доступа.

– В команде BIND или PREP была указана опция EXPLAIN ALL. Эта установка указывает компилятору SQL захватить данные объяснения для динамического SQL во время выполнения, даже если специальный регистр CURRENT EXPLAIN MODE имеет значение NO. Оператор SQL тоже выполняется с возвратом результатов запроса.

Примечание: Информация объяснения захватывается, только когда оператор SQL скомпилирован. После начальной компиляции динамические операторы SQL перекомпилируются, только когда этого требуют изменения среды.Если один и тот же оператор PREPARE выдается последовательно для одинаковых операторов SQL, оператор SQL будет скомпилирован, а данные объяснения - захвачены только в первый раз, когда выдан оператор PREPARE (предполагается, что среда не меняется).

Дополнительную информацию о использование оператора SQL EXPLAIN и специального регистра CURRENT EXPLAIN MODE смотрите в руководстве *SQL Reference*. Дополнительную информацию о командах BIND и PREP смотрите в руководстве *Command Reference*.

Захват информации снимка объяснения

Информация снимка объяснения захватывается, когда оператор SQL скомпилирован и была затребована информация объяснения:

- Для **статических** операторов SQL или операторов SQL с **инкрементным связыванием** информация таблицы объяснения захватывается, если указаны опции EXPLSNAP ALL или EXPLSNAP YES в командах BIND или PREP, или в исходной программе используется статический оператор SQL EXPLAIN.

  Примечание: Когда операторы SQL с инкрементным связыванием компилируются во время выполнения, в таблицы объяснения они помещаются тоже во время выполнения, а не во время связывания. Для вставки в таблицы объяснения используется спецификатор таблицы объяснения и ID авторизации владельца пакета, а не пользователя, выполняющего пакет.

- Для **динамических** операторов SQL снимок объяснения захватывается в следующих ситуациях:
  – Оператор SQL EXPLAIN использует условие FOR SNAPSHOT или WITH SNAPSHOT. Условие FOR SNAPSHOT не приводит к захвату информации

таблицы объяснения, кроме информации, связанной со снимком объяснения. Условие WITH SNAPSHOT приводит к захвату всей информации таблицы объяснения, а также информации, связанной со снимком объяснения.

Пример снимка объяснения, использующего оператор SQL EXPLAIN:

```
EXPLAIN PLAN FOR SNAPSHOT FOR <любой допустимый оператор SQL DELETE,
INSERT, SELECT, SELECT INTO, UPDATE, VALUES или VALUES INTO>
```

Генерируется только снимок объяснения, захваченная информация помещается в таблицы EXPLAIN_INSTANCE и EXPLAIN_STATEMENT.

- Специальный регистр CURRENT EXPLAIN SNAPSHOT имеет значение YES. Это значение указывает компилятору SQL сделать снимок данных объяснения и позволяет выполнить оператор SQL с возвратом результатов запроса.
- Специальный регистр CURRENT EXPLAIN SNAPSHOT имеет значение EXPLAIN. Это значение указывает компилятору SQL снять снимок данных объяснения, но не выполнять оператор SQL.
- В команде BIND или PREP была указана опция EXPLSNAP ALL. Эта опция указывает компилятору SQL снять снимок данных объяснения во время выполнения, даже если специальный регистр CURRENT EXPLAIN SNAPSHOT имеет значение NO. Оператор SQL тоже выполняется с возвратом результатов запроса.

Примечание: Информация объяснения захватывается, только когда оператор SQL скомпилирован. После начальной компиляции динамические операторы SQL перекомпилируются, только когда этого требуют изменения среды.Если один и тот же оператор PREPARE выдается последовательно для одинаковых операторов SQL, оператор SQL будет скомпилирован, а данные объяснения - захвачены только в первый раз, когда выдан оператор PREPARE (предполагается, что среда не меняется).

Дополнительную информацию об использовании оператора SQL EXPLAIN и условиях FOR SNAPSHOT и WITH SNAPSHOT, а также об использовании специального регистра CURRENT EXPLAIN SNAPSHOT смотрите в руководстве *SQL Reference*. Дополнительную информацию о командах BIND и PREP смотрите в руководстве *Command Reference*.

Указания по использованию выходных данных объяснения

Есть несколько способов настройки ваших запросов и среды с помощью анализа данных объяснения. Например:

- **Используются ли индексы?**

Как говорилось в разделе “Влияние индексов на оптимизацию запросов” на стр. 102 , выбор подходящих индексов может значительно улучшить производительность. С помощью выходных данных объяснения можно определить, используются ли индексы, созданные вами, для повышения эффективности определенного набора запросов. Использование индексов отражается в следующих областях выходных данных объяснения:

- Объединенные предикаты
- Локальные предикаты
- Условие GROUP BY
- Условие ORDER BY
- Список выбора.

Возможность объяснения можно также использовать, чтобы оценить, можно ли использовать другой индекс вместо существующего или вообще не использовать индекс. После создания нового индекса соберите для него статистику (используя команду RUNSTATS) и перекомпилируйте ваш запрос. Пользуясь данными объяснения, вы можете заметить, что теперь вместо некоторого просмотра индекса используется просмотр таблицы. Так может получиться из-за изменения кластеризации данных таблицы. Если у ранее используемого индекса теперь упало отношение кластеризации, вы можете:

- Перестроить таблицу, чтобы кластеризовать данные в соответствии с этим индексом
- Использовать команду RUNSTATS, чтобы обновить статистику каталога для таблицы и индекса
- Перекомпилировать ваш запрос
- Пересмотреть выходные данные объяснения, чтобы определить, улучшила ли перестройка таблицы план доступа.

- **Подходит ли тип доступа для вашей программы?**

  При анализе выходных данных объяснения вы можете найти такие типы доступа к данным, которые, как правило, не оптимальны для типа выполняемой программы. Например:

  - **Запросы OLTP (Online Transaction Processing - оперативная обработка транзакций)**

    Программы OLTP - это первые претенденты на использование просмотров индекса с ограничивающими диапазон предикатами, они обычно возвращают всего несколько строк, определяемых с помощью предиката эквивалентности для ключевого столбца. Если запросы OLTP используют просмотр таблицы, вы можете проанализировать данные объяснения и определить, по каким причинам не использовался просмотр индекса.

  - **Обзорные запросы**

    Критерии поиска для запроса “обзорного” типа могут быть очень нечеткими, что приводит к большому числу подходящих строк. Если

пользователь обычно просматривает только несколько первых страниц выходных данных, можно попытаться обеспечить возвращение первых данных до вычисления полного набора ответа. В этом случае цели пользователя отличаются от предполагаемых оптимизатором, который пытается минимизировать потребление ресурсов для всего запроса, а не для нескольких первых страниц данных.

Например, если результат объяснения показывает, что в плане доступа использовались операции объединения с просмотром слияния и сортировки, тогда весь набор ответа будет помещен во временную таблицу, прежде чем какие-либо строки будут возвращены программе. В этом случае можно попытаться изменить план доступа с помощью условия OPTIMIZE FOR оператора SELECT. (Дополнительную информацию об условии OPTIMIZE FOR смотрите в разделе “Условие OPTIMIZE FOR n ROWS” на стр. 80.) В результате оптимизатор может выбрать план доступа, который не собирает весь набор ответа во временную таблицу перед возвратом программе первых строк.

- **Какой тип метода объединения используется?**

  Если запрос объединяет две таблицы, вы можете узнать, какой тип объединения используется. Объединения большого числа строк, такие, как в запросах по поддержке принятия решений, обычно выполняются быстрее с помощью объединения слиянием. Объединения всего нескольких строк, такие, как в запросах OLTP, как правило, выполняются быстрее с помощью объединений с вложенным циклом. Однако в любом случае могут сложиться обстоятельства, такие как использование локальных предикатов или индексов, которые повлияют на работу этих типичных объединений. (Информацию о том, как работают эти два метода объединения, смотрите в разделах “Объединение со вложенным циклом” на стр. 183 и “Объединение слияния” на стр. 184.)

Наглядное объяснение

Инструмент наглядного объяснения Visual Explain можно использовать для более детального, по сравнению с другими методами, изучения запросов, особенно запросов со сложной последовательностью операций. Инструмент Visual Explain доступен не во всех поддерживаемых платформах. Чтобы узнать, поддерживается ли у вас Visual Explain, посмотрите руководство *Быстрый старт* для вашей платформы.

Наглядное объяснение позволяет просмотреть план доступа с графическим представлением объясненных операторов SQL. Полученную информацию можно использовать для настройки производительности запросов SQL. Visual Explain также позволяет динамически объяснять оператор SQL и просматривать диаграмму получающегося плана доступа.

Оптимизатор выбирает план доступа и Visual Explain показывает информацию в виде диаграммы плана доступа, в которой таблицы, индексы и каждая операция над ними обозначены как узлы, а поток данных представлен связями между узлами.

Прежде чем просмотреть диаграмму плана доступа, вы должны создать снимок объяснения. На диаграмме плана доступа можно увидеть подробности о:

- Таблицах и индексах (и связанных с ними столбцах)
- Операторах (например, просмотра, сортировки и объединения таблиц)
- Табличных пространствах и функциях.

Инструмент Visual Explain можно также использовать, чтобы:

- Просмотреть статистику, использованную во время оптимизации. Можно затем сравнить эту статистику с текущей статистикой каталога, чтобы понять, можно ли улучшить производительность, повторно связав пакет.
- Выяснить, использовался ли для доступа к таблице индекс. Если индекс не использовался, Visual Explain может помочь определить, индексация каких столбцов принесет пользу.
- Увидеть эффект различных вариантов настройки, сравнивая версии диаграммы плана доступа для указанного запроса до и после настройки.
- Получать информацию о каждой операции в плане доступа, включая общую примерную стоимость и количество полученных строк.

Дополнительные подробности о Visual Explain можно найти в электронной справке Центра управления. Центр управления можно вызвать, введя в командной строке `db2cc`.

Советчики по SQL

Советчик по индексам - это инструмент управления, облегчающий создание и определение подходящих для ваших данных индексов.

Советчик по индексам удобен для:

- Поиска лучших индексов для сложных запросов.
- Поиска лучших индексов для набора запросов (рабочей нагрузки), возможно, при ограниченных ресурсах.
- Тестирования индекса на рабочей нагрузке без его создания.

Изложим основные понятия, связанные с советчиком по SQL. Основное понятие - это *рабочая нагрузка*. Рабочая нагрузка - набор операторов SQL, который менеджер базы данных должен обработать за определенный промежуток времени. Среди этих операторов SQL могут быть операторы SELECT, INSERT, UPDATE и DELETE. Например, за один месяц менеджер базы данных может

быть должен обработать 1000 операторов INSERT, 10000 операторов UPDATE, 10000 операторов SELECT и 1000 операторов DELETE. Информация в рабочей нагрузке касается типа и частоты использования операторов SQL за данный промежуток времени. Механизм советчика использует эту информацию рабочей нагрузки в сочетание с информацией базы данных, чтобы рекомендовать индексы. Механизм советчика стремится минимизировать полную стоимость рабочей нагрузки.

Во-вторых, используется понятие *виртуальный индекс*. Виртуальные индексы - это индексы, которые не существуют в текущей схеме базы данных. Это могут быть индексы, рекомендуемые вам советчиком, или индексы, которые вы хотите оценить с помощью советчика. Это также могут быть индексы, которые советчик сначала рассматривал как часть процесса, но потом отбросил, потому что решил не рекомендовать их. Виртуальные индексы передаются от вас советчику и обратно через таблицу ADVISE_INDEX.

Чтобы генерировать рекомендуемые индексы, советчик использует рабочую нагрузку и статистику базы данных.

Советчик использует две таблицы EXPLAIN:

- ADVISE_WORKLOAD

  В этой таблице вы описываете рабочую нагрузку, которую надо рассмотреть. Каждая строка в этой таблице представляет оператор SQL и содержит связанную с ним частоту. Для каждой рабочей нагрузки есть идентификатор - это поле WORKLOAD_NAME (имя рабочей нагрузки) таблицы. Все операторы SQL, принадлежащие одной рабочей нагрузке, должны иметь одинаковое значение WORKLOAD_NAME.

  Мастер по индексам и инструмент `db2advis` используют эту таблицу для получения и хранения информации о рабочей нагрузке.

- ADVISE_INDEX

  В этой таблице хранится информация о рекомендуемых индексах. Информацию в эту таблицу помещает компилятор SQL, мастер по индексам, инструмент `db2advis` или пользователь.

  Эта таблица используется в двух целях:

  – Для получения рекомендованных индексов от советчика
  – Для оценки индексов.

Примечание: Чтобы создать эти таблицы, выполните сценарий EXPLAIN.DDL, находящийся в подкаталоге `misc` подкаталога `sqllib`. Эти таблицы может также создать мастер по индексам, если они еще не созданы.

Использование советчика по индексам включает в себя получение входных данных, вызов советчика, обработку выходных данных и некоторые специальные случаи.

Есть три способа подготовить входные данные для советчика по индексам:

- Захват рабочей нагрузки.

  Для создания операторов SQL, которые надо проверить, используйте одно из следующих средств:

  – Монитор для получения динамического SQL.
  – Производная таблица каталога SYSSTMT для получения статического SQL.
  – Добавление операторов и частот путем вырезания и вставки значений в таблицу ADVISE_INDEX.
- Изменение частот рабочей нагрузки, чтобы увеличить или уменьшить важность запросов.
- Задание ограничений на данные, если есть.

Есть четыре способа вызвать советчик по индексам:

- Используя Центр управления.

  Это рекомендуемый способ использования советчика по индексам. В Центре управления раскрывайте дерево объектов, пока не найдете папку **Индексы**. Щелкните правой кнопкой мыши по папке **Индексы** и выберите во всплывающем меню **Создать–>Индекс при помощи мастера**. Откроется мастер по индексам. У мастера по индексам есть подробная справка и им легко пользоваться. В мастере по индексам рабочую нагрузку можно составлять по последним выполненным операторам SQL, по недавно использовавшимся пакетам или же добавляя операторы SQL собственноручно.
- Используя процессор командной строки.

  В командной строке введите `db2advis`. Запустится инструмент `db2advis`, который считает рабочую нагрузку из одного из трех мест:

  – Из командной строки
  – Из операторов в текстовом файле
  – Из таблицы ADVISE_WORKLOAD после того, как вы вставили туда строки с предполагаемой рабочей нагрузкой (операторы SQL и частоты).

  Потом инструмент использует регистр CURRENT EXPLAIN MODE, чтобы получить рекомендуемые индексы в сочетании с внутренним алгоритмом оптимизации для отбора наилучших индексов. Вывод идет на экран вашего терминала, в таблицу ADVISE_INDEX и, по требованию, в выходной файл.

  Например, этот инструмент может порекомендовать индексы для простого запроса “SELECT COUNT(*) FROM SALES WHERE REGION = ’Quebec’”

```
$ db2advis -d SAMPLE \
 -s "SELECT COUNT(*) FROM SALES WHERE REGION = 'Quebec'" \
 -t 1
performing auto-bind

Bind is successful. Used bindfile: /home3/valentin/sqllib/bnd/db2advis.bnd

Calculating initial cost (without recommended indexes) [31.198040] timerons
Initial set of proposed indexes is ready.
Found maximum set of [1] recommended indexes
Cost of workload with all indexes included [2.177133] timerons
cost without index [0] is [31.198040] timerons.  Derived benefit is
[29.020907]
total disk space needed for initial set [1] MB
total disk space constrained to         [-1] MB
  1  indexes in current solution
 [31.198040] timerons  (without indexes)
 [2.177133] timerons  (with current solution)
 [%93.02] improvement

Trying variations of the solution set.
Time elapsed.
LIST OF RECOMMENDED INDEXES
===========================
index[1], 1MB CREATE INDEX WIZ689 ON VALENTIN.SALES (REGION DESC)
===========================
Index Advisor tool is finished.
```

Инструмент db2advis может также рекомендовать индексы для рабочей нагрузки. Можно создать входной файл под названием “sample.sql”:

```
--#SET FREQUENCY 100
select count(*) from sales where region = ?;
--#SET FREQUENCY 3
select projno, sum(comm) tot_comm  from employee, emp_act
where employee.empno = emp_act.empno and
     employee.job='DESIGNER'
group by projno
order by tot_comm desc;
--#SET FREQUENCY 50
select * from sales where sales_date = ?;
```

и выполнить следующую команду:

```
$ db2advis -d sample -i sample.sql -t 0
  found [3] SQL statements from the input file

Calculating initial cost (without recommmended indexes) [62.331280] timerons
Initial set of proposed indexes is ready.
Found maximum set of [2] recommended indexes
Cost of workload with all indexes included [29.795755] timerons
cost without index [0] is [58.816662] timerons.  Derived benefit is
[29.020907]
cost without index [1] is [33.310373] timerons.  Derived benefit is
[3.514618]
```

```
total disk space needed for initial set [2] MB
total disk space constrained to         [-1] MB
  2  indexes in current solution
 [62.331280] timerons  (without indexes)
 [29.795755] timerons  (with current solution)
 [%52.20] improvement

Trying variations of the solution set.
Time elapsed.
LIST OF RECOMMENDED INDEXES
===========================
index[1], 1MB CREATE INDEX WIZ119 ON VALENTIN.SALES (SALES_DATE DESC,
SALES_PERSON DESC)
index[2], 1MB CREATE INDEX WIZ63 ON VALENTIN.SALES (REGION DESC)
===========================
Index Advisor tool is finished.
```

- Используя автоматические методы с режимами EXPLAIN и опциями команды PREP.

  Например, специальный регистр CURRENT EXPLAIN MODE имеет значение RECOMMEND INDEXES. Это значение укажет компилятору SQL захватить данные объяснения и разместить рекомендуемые индексы в таблице ADVISE_INDEX, оператор SQL не выполняется.

  Другой вариант: специальный регистр CURRENT EXPLAIN MODE имеет значение EVALUATE INDEXES. Это значение укажет компилятору SQL использовать индексы, помещенные пользователем в таблицу ADVISE_INDEX. Пользователь вставляет новую строку для каждого индекса, который надо оценить. Информация, требуемая для каждого индекса: имя индекса, имя таблицы и имена столбцов, составляющих оцениваемый индекс. Как уже говорилось, специальный регистр CURRENT EXPLAIN MODE должен иметь значение EVALUATE INDEXES. Тогда компилятор SQL находит в таблице ADVISE_INDEX индексы, у которых поле USE_INDEX имеет значение “Y” (они называются виртуальными индексами). Все динамические операторы, выполняемые в режиме EVALUATE INDEXES, объясняются так, как если бы эти виртуальные индексы были доступны. Если виртуальные индексы улучшают производительность операторов, компилятор SQL потом выбирает использование этих индексов. Иначе эти индексы игнорируются. Просматривая результаты EXPLAIN, вы можете увидеть, использовал ли компилятор SQL индексы, предложенные пользователем. Следует рассмотреть возможность применения индексов, использованных компилятором, для улучшения доступа.
- Используя интерфейс уровня вызовов (CLI).

  Если вы пользуетесь этим интерфейсом при написании программ, вы также можете использовать советчик.

Результаты работы советчика можно использовать по-разному:

- Интерпретировать вывод советчика по индексам.

Чтобы увидеть, какие индексы были рекомендованы советчиком, можно воспользоваться следующим запросом:

```
SELECT CAST(CREATION_TEXT as CHAR(200))
   FROM ADVISE_INDEX
```

- Применить рекомендации Советчика по индексам.
- Определить, когда надо отбросить индекс.

Чтобы получить лучшие рекомендации для определенного запроса, предлагается получать совет только для этого запроса. Можно использовать мастер по индексам, чтобы определить рекомендуемые индексы для отдельного запроса, построив рабочую нагрузку, содержащую только этот запрос.

Образец рабочей нагрузки можно получить из выходных данных Монитора событий. С помощью Монитора событий можно собрать данные о выполнении динамического SQL. Потом эти операторы можно передать обратно на вход советчика.

Мастер по индексам - это простой, понятный, легко применимый и наглядный интерфейс, дающий отличный способ доступа к советчику.

Часть 3. Настройка и конфигурирование вашей системы

Глава 8. Производительность работы

В следующих темах описано, как повысить производительность обработки запросов SQL:

- Как DB2 использует память
- Управление пулом буферов базы данных
- Управление несколькими пулами буферов базы данных
- Предварительная выборка данных в пул буферов
- Настройка серверов ввода/вывода для предварительной выборки и параллельного ввода/вывода
- Сортировка
- Реорганизация каталогов и пользовательских таблиц
- Особенности производительности для устройств DMS
- Управление расходами на инициализацию
- Агенты базы данных
- Использование системного монитора базы данных
- Расширение памяти.

Вопросы производительности рассматриваются также в следующих главах:

- “Глава 3. Особенности прикладного программирования” на стр. 43
- “Глава 4. Факторы среды” на стр. 95
- “Глава 5. Статистика системного каталога” на стр. 117.

В руководстве *Administration Guide: Planning* можно прочесть об особенностях физической структуры баз данных.

Как DB2 использует память

Многие параметры конфигурации DB2 влияют на использование памяти в системе. Некоторые из них затрагивают память на сервере, другие - на клиенте, третьи - и там, и там. Кроме того, отведение памяти, как и освобождение, производится в разное время и из разных областей системы.

Системный администратор, помимо прочего, должен учитывать общий баланс использования памяти в системе. Выполняемые в операционной системе прикладные программы могут по-разному использовать память. Например, некоторые прикладные программы могут использовать кэш файловой системы, тогда как менеджер баз данных использует для кэширования данных не

возможности операционной системы, а собственный пул буферов. Другие особенности смотрите в разделе “Задание параметров, влияющих на использование памяти” на стр. 260.

В разделе рис. 22 описано использование менеджером баз данных различных типов памяти. В данной иллюстрации использования памяти предполагается, что среда - не Enterprise – Extended Edition и не среда с несколькими логическими узлами. В среде Enterprise – Extended Edition и в среде с несколькими логическими узлами используется несколько наборов совместно используемой памяти менеджера баз данных (по одному на узел).

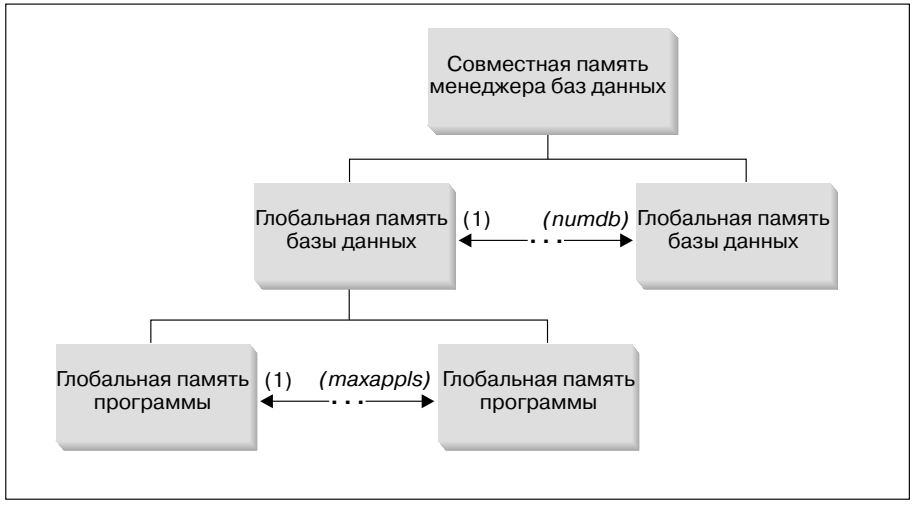


Рисунок 22. Типы памяти, используемые менеджером баз данных

Память отводится каждому экземпляру менеджера баз данных в следующие моменты времени:

- При запуске менеджера баз данных (`db2start`) отводится область, обозначенная “Совместно используемая память менеджера баз данных”; эта область не освобождается до остановки менеджера баз данных (`db2stop`). Эта область содержит информацию, которая нужна менеджеру баз данных для управления работой всех соединений баз данных. С момента, когда с базой данных соединится первая прикладная программа, отводятся глобальные и собственные области памяти.
- Когда база данных активируется или с ней устанавливается первое соединение, размещается “Глобальная память базы данных”. Глобальная память базы данных используется всеми прикладными программами, соединяющимися с базой данных, и содержит такие области памяти, как пулы буферов, список блокировок, куча базы и куча утилит.

- Когда прикладная программа соединяется с базой данных, размешается "Глобальная память программы" (это происходит только в среде распределенных баз данных или при включенном параметре конфигурации *intra_parallel*). Эта память используется агентами прикладной программы для совместного доступа к данным и взаимной координации действий.
- (Не показано на предыдущей диаграмме:) Когда агенту назначается работа для определенной программы (в результате запроса соединения или, в параллельной среде, нового запроса SQL), для этого агента отводится "Собственная память агента". Область собственной памяти агента отводится для этого агента и содержит области памяти, которые будут использоваться только этим конкретным агентом, например, кучу сортировки и кучу программы.

  После того, как одна программа начнет использовать базу данных, всем программам, подключившемся позже, отводится только собственная память агента и глобальная совместно используемая память программы.

В разделе рис. 22 на стр. 254 описано, как могут повлиять на память параметры конфигурации. В частности, параметры из следующего списка могут ограничивать объем памяти, отводимый для определенных целей. (В среде распределенных баз данных эта память требуется в каждом разделе базы данных.)

- Параметр *numdb* определяет максимальное число одновременно активных баз данных (используемых разными программами). Поскольку у каждой базы данных есть своя область глобальной памяти, при увеличении значения этого параметра потенциальный объем отводимой памяти возрастает.
- Параметр *maxappls* определяет максимальное число программ, которые могут одновременно соединиться с одной базой данных. Оно влияет на потенциальный объем памяти, отводимый для "Собственной памяти агента" и "Глобальной памяти программы" этой базы данных. (Обратите внимание на то, что этот параметр можно задать разным для разных баз данных.)
- (Не показан на предыдущей диаграмме:) Параметр *maxagents* (как и параметр *max_coordagents* для параллельной обработки) ограничивает число агентов Менеджера баз данных, которые могут одновременно существовать на всех активных базах данных экземпляра. Как и *maxappls*, эти параметры ограничивают объем памяти, отводимый для "Собственной памяти агента" и "Глобальной памяти программы". (Сведения об агентах смотрите в разделе "Агенты базы данных" на стр. 287.)

В разделе рис. 23 на стр. 256 описан общий объем памяти, используемый для поддержки программ. Следующие параметры конфигурации позволяют управлять размером этой памяти, ограничивая число и размер ″сегментов памяти″ (порций логической памяти).

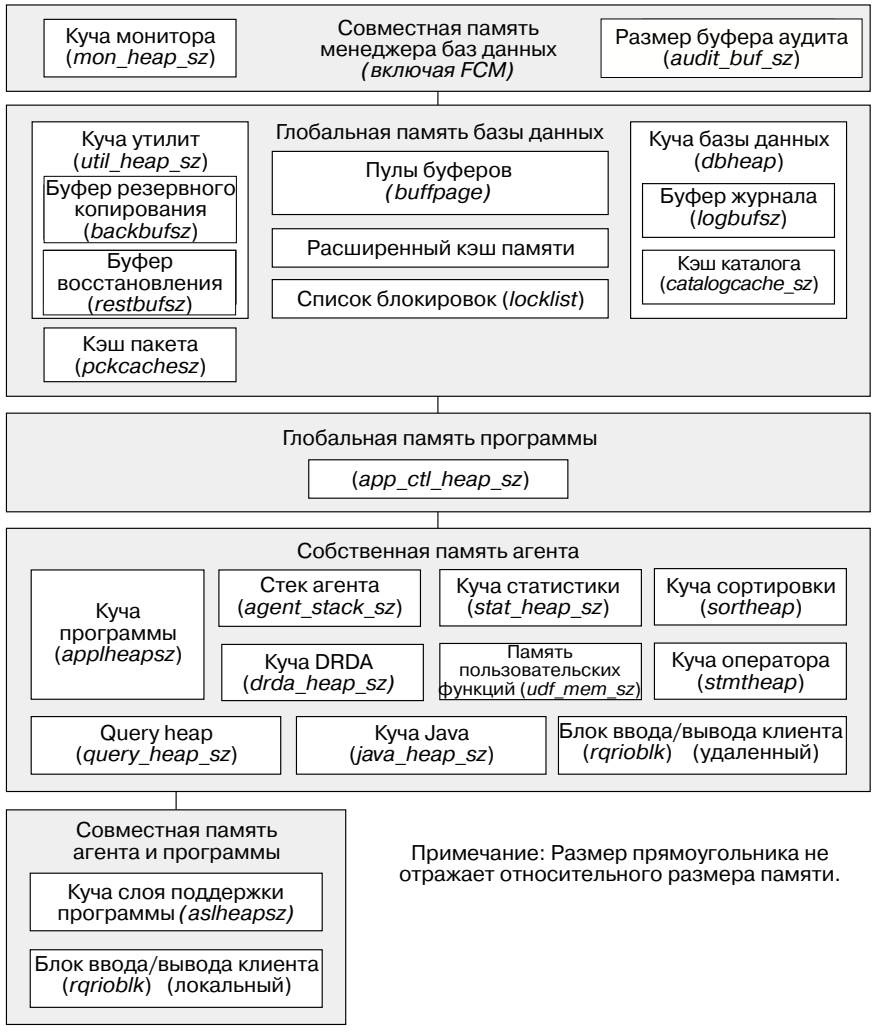


Рисунок 23. Как память используется Менеджером баз данных

Совместно используемая память Менеджера баз данных

Для работы менеджера баз данных требуется память. Объем этой памяти может быть весьма значительным, особенно в средах внутриразделного и межраздельного параллелизма. Оценка этого объема и управление им описаны в следующих разделах:

- “Агенты базы данных” на стр. 287. Агенты, запускаемые для программ, требуют много памяти, особенно если не задано подходящее значение *maxagents*.

- “Требования FCM” на стр. 261. В системах распределенных баз данных много памяти требует менеджер быстрой связи (FCM, fast communications manager), особенно если не задано подходящее значение *fcm_num_buffers*.

  Память, требуемая FCM, отводится либо только из пула буферов FCM, либо из Совместно используемой памяти Менеджера баз данных и пула буферов FCM, если система распределенных баз данных использует несколько логических узлов. Подробности смотрите в приведенном ниже описании пула буферов FCM.

Пул буферов FCM

Если в вашей системе распределенных баз данных нет нескольких логических узлов, Совместно используемая память Менеджера баз данных и пул буферов FCM будут такими, как описано в разделе рис. 24.

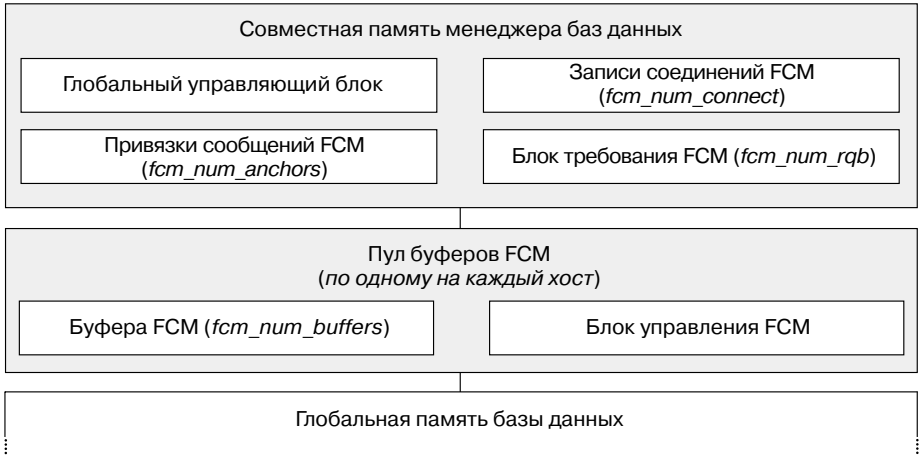


Рисунок 24. Пул буферов FCM, когда не используется несколько логических узлов

Если в вашей системе распределенных баз данных используется несколько логических узлов, Совместно используемая память Менеджера баз данных и пул буферов FCM будут такими, как описано в разделе рис. 25 на стр. 258.

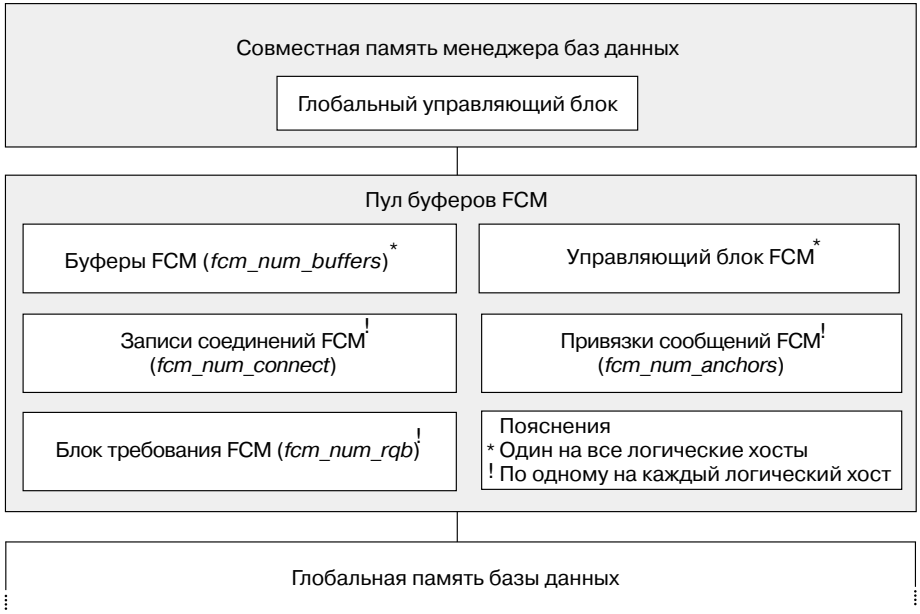


Рисунок 25. Пул буферов FCM, когда используется несколько логических узлов

Глобальная память базы данных

На Глобальную память базы данных влияют следующие параметры конфигурации:

- Число сегментов памяти ограничивает параметр *numdb* (смотрите раздел "Максимальное число одновременно активных баз данных (numdb)" на стр. 502).
- Максимальный размер сегментов памяти определяется значениями следующих параметров:
  - "Размер пула буферов (buffpage)" на стр. 366 (если размер пула буферов задан как -1), или явные размеры, заданные при создании или изменении пулов буферов
  - "Максимальная память для списка блокировок (locklist)" на стр. 375
  - "Куча базы данных (dbheap)" на стр. 369
  - "Размер кучи утилит (util_heap_sz)" на стр. 372
  - "Размер сегмента расширения памяти (estore_seg_sz)" на стр. 417
  - "Число сегментов расширения памяти (num_estore_segs)" на стр. 418.
  - "Размер кэша пакета (pckcachesz)" на стр. 378.

Глобальная память программы

На Глобальную память программы влияет следующий параметр

конфигурации: “Размер кучи управления прикладной программы (app_ctl_heap_sz)” на стр. 379. В параллельных системах требуется также пространство для кучи управления программой, совместно используемой агентами, работающими для одной и той же программы в разделе базы данных. Отведение кучи происходит, когда требует соединения агент, первым получивший запрос от прикладной программы. Агент может быть либо координирующим агентом, либо подагентом (смотрите раздел “Агенты базы данных” на стр. 287).

Собственная память агента

- Число сегментов памяти ограничивает наименьшее из следующих значений:
  - Общее число *maxappls* для всех активных баз данных (смотрите раздел “Максимальное число активных прикладных программ (maxappls)” на стр. 419)
  - Значение *maxagents* (смотрите раздел “Максимальное число агентов (maxagents)” на стр. 425).
- Максимальный размер сегментов памяти определяется значениями следующих параметров:
  - “Размер кучи прикладных программ (applheapsz)” на стр. 385
  - “Размер кучи сортировки (sortheap)” на стр. 381
  - “Размер кучи операторов (stmtheap)” на стр. 385
  - “Размер кучи статистики (stat_heap_sz)” на стр. 386
  - “Размер кучи запросов (query_heap_sz)” на стр. 387
  - “Параметр drda_heap_sz (размер кучи DRDA)” на стр. 388
  - “Размер совместной памяти пользовательских функций (udf_mem_sz)” на стр. 389
  - “Размер стека агента (agent_stack_sz)” на стр. 390.

Совместно используемая память агента/программы

- Общее число сегментов совместно используемой памяти агента/программы (для локальных клиентов) ограничено наименьшим из следующих значений:
  - Общее число *maxappls* для всех активных баз данных (смотрите раздел “Максимальное число активных прикладных программ (maxappls)” на стр. 419)
  - Значение *maxagents* (смотрите раздел “Максимальное число агентов (maxagents)” на стр. 425) или (в параллельных системах) *max_coordagents* (смотрите раздел “Максимальное число координирующих агентов (max_coordagents)” на стр. 427).
- На Совместно используемую память агента/программы влияют также следующие параметры:
  - “Размер кучи слоя поддержки программ (aslheapsz)” на стр. 395.

– “Размер блока ввода-вывода клиента (rqrioblk)” на стр. 398.

Задание параметров, влияющих на использование памяти

Даже в системах, на которых установлен максимальный объем памяти, *не следует* задавать максимальные значения параметрам выделения памяти без тщательной предварительной оценки. Многие их этих параметров могут позволить менеджеру баз данных с легкостью забрать всю доступную память системы. Кроме того, управление большим объемом памяти может потребовать значительной дополнительной работы от менеджера баз данных, что увеличит расходы на служебные операции.

Некоторые операционные системы на основе UNIX размещают пространство подкачки при отведении памяти для процесса, а не при его выгрузке из памяти в пространство подкачки. В этих случаях нужно убедиться, что в пространстве подкачки доступно место для общего размера совместно используемой памяти.

Для большинства параметров конфигурации память выделяется только тогда, когда она требуется. Эти параметры отражают максимальный размер конкретной кучи памяти. Важное исключение из этого правила - параметры, для которых память выделяется непосредственно при задании значения параметра:

- “Размер пула буферов (buffpage)” на стр. 366 (если размер пула буферов задан как -1), или явные размеры, заданные при создании или изменении пулов буферов
- “Порог кучи сортировки (sheapthres)” на стр. 382
- “Максимальная память для списка блокировок (locklist)” на стр. 375
- “Размер кучи слоя поддержки программ (aslheapsz)” на стр. 395
- “Число привязок сообщений FCM (fcm_num_anchors)” на стр. 488
- “Число буферов FCM (fcm_num_buffers)” на стр. 488
- “Число записей соединений FCM (fcm_num_connect)” на стр. 490
- “Число блоков требований FCM (fcm_num_rqb)” на стр. 490.

Подходящие значения для этих типов параметров рекомендуется определять в ходе измерений производительности, при которых на сервере выполняются типичные и наиболее трудоемкие операторы SQL, и значения параметров изменяют до обнаружения максимального показателя производительности. Если изобразить на графике зависимость производительности от значения параметра объема памяти, та точка, где кривая выходит на плато или начинает идти вниз, покажет, когда дополнительное выделение памяти не дает улучшения для программы и, следовательно, является бесполезной тратой. (Смотрите раздел “Глава 12. Измерение производительности” на стр. 335.)

Верхние пределы выделения памяти для нескольких параметров могут превышать возможности существующих компьютеров и операционных систем. Эти пределы выбраны в расчете на будущий прогресс техники.

Описание допустимых диапазонов параметров смотрите в разделе “Глава 13. Конфигурирование DB2” на стр. 349.

Требования FCM

Начните со значений по умолчанию при настройке следующих параметров конфигурации менеджера быстрой связи (FCM, fast communications manager):

- “Число буферов FCM (fcm_num_buffers)” на стр. 488
- “Число блоков требований FCM (fcm_num_rqb)” на стр. 490
- “Число записей соединений FCM (fcm_num_connect)” на стр. 490
- “Число привязок сообщений FCM (fcm_num_anchors)” на стр. 488.

Чтобы настроить эти параметры, найдите при помощи системного монитора баз данных низший уровень для свободных буферов, свободных привязок сообщений, свободных записей соединений и свободных блоков запросов. Если нижний уровень окажется меньше 10 процентов числа соответствующего свободного элемента данных, увеличьте значение соответствующего параметра. Информацию о системном мониторе баз данных смотрите в разделе “Использование системного монитора базы данных” на стр. 293.

Информацию о включении FCM смотрите в руководстве *Administration Guide: Planning*.

Управление пулом буферов базы данных

Пул буферов - это область памяти, где временно хранятся и изменяются страницы баз данных (содержащие строки таблиц или записи индексов). Пул буферов предназначен для улучшения производительности системы баз данных. Доступ к данным из памяти намного быстрее, чем с диска. Это значит, что чем реже менеджеру баз данных придется читать с диска или писать на диск, тем выше будет производительность.

Конфигурация одного или нескольких пулов буферов - единая и важнейшая область настройки, поскольку именно здесь производится большинство операций с данными для программ, подключенных к базе данных (за исключением данных больших объектов и длинных полей).

Когда прикладная программа обращается к строке таблицы в первый раз, менеджер баз данных помещает страницу, содержащую эту строку, в пул буферов. В следующий раз, когда какая-либо программа запросит данные, сначала будет проверен пул буферов. Если требуемые данные будут найдены на страницах в пуле буферов, менеджеру баз данных не придется читать эти данные с диска. Устранение необходимости считывания данных с диска повышает производительность.

Память, связанная с пулом буферов, отводится при активации базы данных или при первом соединении программы с базой данных. Пулы буферов используются в первую очередь в интересах прикладных программ: когда все программы отсоединятся, память, связанная с пулом буферов, освобождается.

Страницы сохраняются в пуле буферов, пока не будет закрыта база данных или пока пространство, занимаемое страницей, не потребуется другой странице. При выборе пространства в пуле буферов для размещения новой страницы используются следующие критерии:

- Последнее обращение к странице
- Вероятность повторного обращения к странице последнего агента, обратившегося к этой странице
- Тип данных на странице
- Не была ли страница изменена и не записана после этого на диск. (Измененные страницы всегда записывают на диск перед тем, как переписать их.)

Примечание: После записи на диск измененные страницы не удаляются из пула буферов, пока занимаемое ими пространство не понадобится для других страниц. Пока эти страницы не переписаны, к их данным сохраняется доступ.

При создании пула буферов размер страницы по умолчанию равен 4 Кбайтам. Можно при создании пула буферов задать размер страницы 4 Кбайта, 8 Кбайт, 16 Кбайт или 32 Кбайта. Если пулы буферов создаются с использованием одного размера страницы, с ними могут связываться только табличные пространства, созданные с тем же размером страницы. Нельзя изменить размер страницы пула буферов после его создания. Надо создать новый пул буферов с требуемым размером страницы.

Работа с большой памятью в системах Windows

При работе с Windows 2000 поддерживаются размеры пула буферов до 64 Гбайт минус размер DB2 и размер операционной системы. (Предполагается, что DB2 - основная программ в системе.) Такая поддержка доступной при помощи Microsoft Address Windowing Extensions (AWE).

AWE может использоваться с пулами буферов любого размера, но если вам нужно использовать AWE на пулах буферов большего размера, существуют другие рекомендуемые продукты Windows. Windows 2000 Advanced Server обеспечивает поддержку до 8 Гбайт памяти. Сервер Windows 2000 Data Center обеспечивает поддержку до 64 Гбайт памяти.

Для поддержки пулов буферов AWE нужно правильно сконфигурировать DB2 и Windows 2000. Пул буферов, который будет использовать преимущества AWE, должен существовать в базе данных.

Для выделения 3 Гбайт пользовательского пространства используйте опцию загрузки /3GB Windows 2000. Она допускает использование окна AWE больших размеров. Для доступа к объему памяти более 4 Гбайт через интерфейс памяти AWE используйте опцию загрузки /PAE Windows 2000. Чтобы убедиться, что опция загрузки выбрана правильно, выберите Управление -> Система а затем "Запуск и восстановление". В выпадающем списке можно увидеть доступные опции загрузки. Если нужная вам опция загрузки (/3GB или /PAE) выбрана, можно приступить к следующей задаче по установке поддержки AWE. Если желаемая вами опция загрузки недоступна для выбора, надо добавить эту опцию в файл boot.ini на системном диске. Файл boot.ini содержит список действий, которые нужно выполнить при запуске операционной системы. Добавьте /3GB или /PAE или обе опции (через пробел) в конце списка существующих параметров. Сохранив измененный файл, можно проверить и выбрать правильную опцию загрузки, как сказано выше.

Кроме того, следует в Windows 2000 задать для пользователя, под именем которого установлена DB2, опцию"lock pages in memory"-right. Для этого надо зарегистрироваться в Windows 2000 как пользователь, установивший DB2, в меню Start (Пуск) в Windows 2000 выбрать папку "Administrative Tools" (Инструменты управления), а затем программу "Local Security Policy" (Локальная политика защиты). В этой программе можно выбрать назначение прав пользователей, установив "lock pages in memory"-right.

Для DB2 требуется задать переменную реестра DB2_AWE. Чтобы правильно задать эту переменную реестра, нужно знать ID пула буферов, для которого вы хотите разрешить поддержку AWE. Этот ID можно найти в столбце BUFFERPOOLID производной таблицы системного каталога SYSCAT.BUFFERPOOLS. Кроме того, надо знать выделяемое число физических страниц и число страниц окна адресации. Число выделяемых физических страниц должно быть несколько меньше общего числа физических страниц. Фактическое выбранное число будет зависеть от рабочей среды. Например, для среды, где в системе используются только DB2 и программы баз данных, в качестве значения, используемого с переменной DB2_AWE, можно выбрать объем на 0,5 - 1 Гбайт меньше общего объема физических страниц. В случае среды, где системой используются программы, не относящиеся к базе данных, следует увеличить значение, вычитаемое из общего объема, предоставив большее количество физических страниц для этих других программ. Число, используемое в переменной реестра DB2_AWE - это число физических страниц, которые будут использоваться для поддержки AWE и DB2. Верхний предел для страниц окна адресации - 1,5 Гбайта (2,5 Гбайта, если используется опция загрузки Windows 2000 /3GB).

Информацию о задании переменной реестра DB2 DB2_AWE смотрите далее в этой книге в "Приложении A. Переменные среды и реестра DB2", в описании переменных реестра.

Работа со страницами пула буферов

У страниц в пуле буферов могут быть различные атрибуты:

- Используемые страницы - те, которые сейчас читаются или изменяются. Другие агенты могут их читать, но не изменять.
- “Грязные” страницы - те, в которых были изменены данные и которые после этого еще не были записаны на диск. После записи страницы на диск она считается “чистой” и остается в пуле буферов. Пространство, занимаемое чистыми страницами, может быть использовано для новых страниц и доступно для переноса в соответствующий кэш расширенной памяти (если он определен).

Запись страниц из пула буферов на диск возможна в моменты, когда процент занятого измененными страницами пространства в пуле буферов превышает значение, заданное параметром конфигурации *chngpgs_thresh*. Кроме того, вам может потребоваться задать в конфигурации базы данных несколько агентов очистки страниц. Эти агенты записывают страницы на диск, чтобы агенты базы данных смогли найти доступное пространство в пуле буферов.

Агенты очистки страниц выполняют операции ввода/вывода, которые иначе пришлось бы выполнять агентам базы данных. В результате транзакциям не приходится ждать, пока их агенты базы данных запишут страницы на диск, и поэтому прикладные программы выполняются быстрее. (Агенты очистки страниц иногда называют *асинхронными чистильщиками страниц* или *асинхронными процессами записи буферов*, поскольку они могут выполнять свою работу одновременно с агентами базы данных.)

Чтобы изменить число агентов очистки страниц, используйте параметр конфигурации *num_iocleaners* (по умолчанию создается один агент очистки страниц). Подробности смотрите в разделе “Число асинхронных чистильщиков страниц (num_iocleaners)” на стр. 412. Задайте для этого параметра значение между единицей и числом физических дисков базы данных. Чем больше это число, тем выше производительность при нагрузке, связанной с интенсивными изменениями. Это верно также, если число процессов записи данных и индексов велико по отношению к числу асинхронных процессов записи данных и индексов.

Благодаря записи страниц на диск ускоряется также восстановление базы данных в случае аварии системы, поскольку увеличивается часть пула буферов, которую менеджер баз данных может восстановить с диска без необходимости использовать файлы журнала базы данных. В результате очистка страниц будет затребована, если размер журнала, который придется прочитать при восстановлении, превышает значение:

```
logfilsiz * softmax
```

где:

- *logfilsiz* - размер файлов журнала (смотрите раздел “Размер файлов журнала (logfilsiz)” на стр. 435)
- *softmax* - процент файлов журнала, восстанавливаемый после аварии базы данных (смотрите “Диапазон восстановления и интервал мягких контрольных точек (softmax)” на стр. 443).

  Например, если значение softmax - 250, то 2,5 файла журнала будут содержать изменения, требующие восстановления в случае аварии.

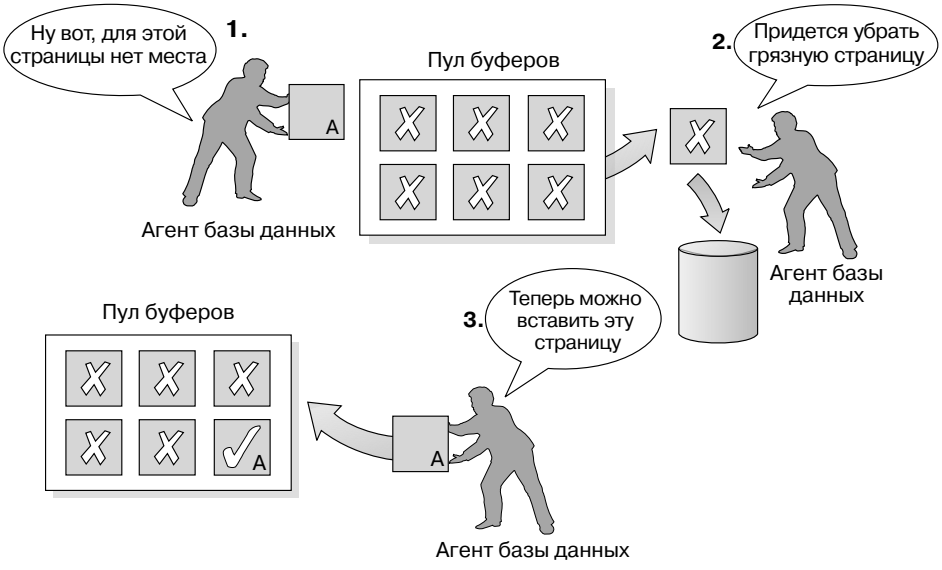
С помощью системного монитора баз данных можно отследить число требований очистки страниц, чтобы минимизировать время чтения журнала при восстановлении. Дополнительную информацию смотрите в описании элемента монитора *pool_lsn_gap_clns (переключаемые чистильщики пространства журнала пула буферов)* в руководстве *System Monitor Guide and Reference*.

Часть журнала, которую потребуется прочитать при восстановлении, заключена между следующими записями в журнале:

- Последняя сделанная в журнале запись
- Запись в журнале, описывающая самое старое изменение данных в пуле буферов.

На следующем рисунке ситуация, когда работа по управлению пулом буферов разделена между агентами очистки страниц и агентами базы данных, сравнивается с ситуацией, когда все операции ввода/вывода производятся агентами базы данных.

Без чистильщиков страниц



С чистильщиками страниц

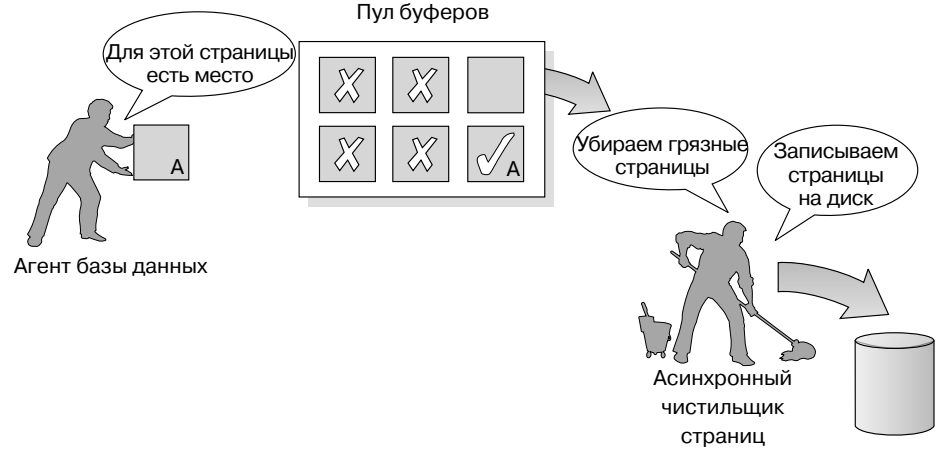


Рисунок 26. Асинхронный чистильщик страниц. “Грязные” страницы записываются на диск.

Управление несколькими пулами буферов базы данных

Каждой базе данных требуется по меньшей мере один пул буферов. Однако иногда выгоднее создать для одной базы данных несколько пулов буферов разного размера. Для создания, изменения и удаления пула буферов служат соответственно операторы CREATE, ALTER и DROP BUFFERPOOL. Задать, какие данные будут кэшироваться в пуле буферов, можно с помощью операторов CREATE TABLESPACE и ALTER TABLESPACE.

Параметр конфигурации *buffpage* задает размер всех пулов буферов, если в производной таблице каталога SYSCAT.BUFFERPOOLS для этого размера задано значение -1. (В противном случае этот параметр игнорируется.) Размер одного пула буферов можно задать с помощью операторов DDL ALTER BUFFERPOOL или CREATE BUFFERPOOL.

У новой база данных по умолчанию есть пул буферов под именем IBMDEFAULTBP, размер которого определяется платформой. После создания или перенастройки базы данных для нее можно создать другие пулы буферов.

Разрабатывая базу данных, вы могли предпочесть размер страниц табличных пространств 8 Кбайт. В таком случае вы должны создать пул буферов с размером страниц 8 Кбайт (а также одно или несколько табличных пространств с тем же размером страниц).

В среде многораздельных баз данных у всех пулов буферов одной базы данных по умолчанию одно и тоже определение во всех разделах базы данных (если другое не задано в операторе CREATE BUFFERPOOL и размер пула буферов для одного раздела базы данных не изменялся при помощи оператора ALTER BUFFERPOOL).

Если вы создаете табличное пространство с размером страниц 4 Кбайта и не назначаете его конкретному пулу буферов, это табличное пространство будет назначено пулу буферов по умолчанию. Если вы создаете табличное пространство с размером страниц более 4 Кбайт (8 КБайт, 16 КБайт, 32 КБайта), нужно назначить его пулу буферов, использующему тот же размер страницы. Если этот пул буферов сейчас не активен, DB2 попытается назначить табличное пространство активному пулу буферов, использующему тот же размер страницы (если такой пул доступен). Это назначение, если оно будет сделано - временное. Если при новой активации базы данных первоначально заданный пул буферов окажется активен, DB2 назначит табличное пространство этому пулу буферов.

Нельзя добавить оператором ALTER TABLESPACE табличное пространство к пулу буферов, использующему другой размер страницы.

При создании или изменении пулов буферов общий объем памяти, требуемый для всех пулов буферов, должен быть доступен менеджеру баз данных, чтобы при запуске базы данных можно было разместить все пулы буферов. Если эта память окажется недоступна при запуске базы данных, менеджер баз данных попытается запустить пул буферов по умолчанию (IBMDEFAULTBP) и по одному пулу буферов, определенных для других размеров страницы, но с минимальными размерами (по 16 страниц). Размер такого минимального пула буферов можно переопределить при помощи переменной реестра DB2_OVERRIDE_BPF. Дополнительную информацию об этой и других переменных реестра и среды смотрите в разделе “Приложение A. Переменные реестра и среды DB2” на стр. 521. При каждой неудачной попытке запустить пул буферов выдается предупреждение; база данных продолжает оставаться в этом состоянии до изменения ее конфигурации и полного перезапуска базы данных.

Менеджеру баз данных разрешено запускаться с минимальными значениями размеров для того, чтобы вы могли соединиться с базой данных. После этого можно изменить конфигурацию размеров пула буферов или выполнить другие срочные задачи, чтобы перезапустить базу данных с правильными размерами пула буферов. Не стоит длительное время использовать базу данных в таком состоянии.

Примечание: Можно изменять размер и атрибуты пула буферов по умолчанию, но нельзя отбросить этот пул. Кроме того, для каждого пула буферов существует минимальный размер, зависящий от используемой платформы.

Есть преимущества в отведении для пулов буферов большого объема памяти. Так, большие размеры пулов буферов:

- Позволяют сохранять в пуле буферов часто запрашиваемые страницы, что ускоряет доступ. Уменьшение числа операций ввода/вывода может снизить частоту конфликтов ввода-вывода и следовательно, время ответа и ресурсы процессора, требуемые для операций ввода/вывода.
- Обеспечивают возможность достичь более высоких скоростей транзакций при том же времени ответа.
- Предотвращают конфликты ввода/вывода для часто используемых устройств дискового хранения, таких как таблицы каталога и часто запрашиваемые пользовательские таблицы и индексы. Запрашиваемые сортировки также выигрывают от уменьшения частоты конфликтов ввода/вывода на устройствах дискового хранения, содержащих временные табличные пространства.

Выбор одного или нескольких пулов буферов

Если для вашей системы выполняется хотя бы одно из следующих условий, следует использовать только один пул буферов:

- Общее пространство буферов меньше 10000 страниц по 4 Кбайта.

- Нет возможности привлечь для специальной настройки лиц, знакомых с прикладными программами.
- Вы работаете на тестовой системе.

Если ни одно из этих условий не имеет места, вы можете использовать несколько пулов буферов ради потенциального повышения производительности в следующих аспектах:

- Можно поместить временные табличные пространства в отдельный пул буферов, чтобы обеспечить более высокую производительность для запросов, требующих временного хранения, особенно для интенсивных сортировок.
- Если у вас есть данные, к которым требуется частый и быстрый доступ со стороны многих коротких прикладных программ, выполняющих транзакции изменения, можно вынести табличное пространство с этими данными в отдельный пул буферов. При правильном выборе размера этого пула его страницы будет легче найти, что вносит вклад в уменьшение времени ответа и снижение стоимости транзакции.
- Можно выносить данные в отдельные пулы буферов, чтобы отдать предпочтение некоторым программам, данным и индексам. Например, вы можете пожелать поместить в отдельный пул буферов те таблицы и индексы, которые изменяются часто, обособив их от тех таблиц и индексов, которые часто запрашиваются, но редко изменяются. Такая перемена снизит влияние частых изменений (в первом наборе таблиц) на частые запросы (во втором наборе таблиц).
- Можно использовать малые пулы буферов для данных, к которым обращаются редко используемые программы, особенно в том случае, когда программе весьма редко требуется доступ к весьма большой таблице. В этом случае нет нужды хранить данные в памяти пула буферов дольше, чем в течение одного запроса. Выгодно оставить для этих данных небольшой пул буферов и освободить дополнительную память для других целей (например, для других пулов буферов).
- Разделив разные работы и данные по разным пулам буферов, можно получить хорошие и относительно недорогие диагностические данные производительности от трассировок статистики и учета.

Предварительная выборка данных в пул буферов

Предварительная выборка страниц индексов и данных в пул буферов может повысить производительность, сократив время ожидания завершения операции ввода/вывода. *Предварительная выборка* страниц - это чтение с диска одной или нескольких страниц, использование которых только прогнозируется. Есть две категории предварительной выборки:

- *Последовательная предварительная выборка* - это механизм, который читает в пул буферов последовательные страницы до того, как эти страницы

запрашиваются прикладной программой. (Смотрите раздел “Анализ последовательной предварительной выборки”.)

- *Предварительная выборка списка*, или предварительная выборка последовательности списка - это способ эффективного доступа к страницам данных, даже если требуемые страницы данных не являются последовательными. (Смотрите раздел “Как работает предварительная выборка списка” на стр. 272.)

Оба эти метода чтения страниц данных дополняют обычное чтение. Обычное чтение используется при извлечении одной или небольшого числа последовательных страниц. При обычном чтении передается одна страница.

Дальнейшую информацию об использовании предварительной выборки смотрите также в разделе “Настройка серверов ввода/вывода для предварительной выборки и параллельного ввода/вывода” на стр. 273.

Анализ последовательной предварительной выборки

Чтение в пул буферов нескольких последовательных страниц за одну операцию ввода/вывода может значительно снизить расходы на служебные операции, связанные с выполнением прикладной программы. Кроме того, параллельное выполнение нескольких операций ввода/вывода для одновременного считывания в пул буферов нескольких диапазонов страниц может сократить время ожидания программой завершения операций ввода/вывода.

Предварительная выборка запускается, когда менеджер баз данных определяет, что есть условия для последовательного ввода/вывода и предварительная выборка может повысить производительность. В таких случаях, как просмотр таблиц и сортировка таблиц, менеджер баз данных легко определяет, что последовательная предварительная выборка повысит производительность ввода/вывода. В этих случаях менеджер баз данных запускает последовательную предварительную выборку автоматически. В следующем примере может потребоваться просмотр таблицы, и вероятно применение последовательной предварительной выборки:

```
SELECT NAME FROM EMPLOYEE
```

Количество страниц в предварительной выборке можно задать для каждого табличного пространства с помощью условия PREFETCHSIZE в операторе CREATE TABLESPACE или ALTER TABLESPACE. Его значение сохраняется в столбце PREFETCHSIZE таблицы системного каталога SYSCAT.TABLESPACES.

Хорошо задавать значение PREFETCHSIZE в явном виде, как произведение значения EXTENTSIZE для табличного пространства и числа контейнеров в табличном пространстве. (Размер экстента - это число страниц, которые менеджер баз данных записывает в контейнер до того, как перейти к другому контейнеру; смотрите раздел “Проектирование и выбор табличных пространств” в руководстве *Administration Guide: Planning*.) Например, если

размер экстента - 16 страниц, а в табличном пространстве два контейнера, можно задать размер предварительной выборки 32 страницы.

Менеджер баз данных наблюдает за использованием пула буферов и не допускает, чтобы при предварительной выборке данных в пуле буферов стирались страницы, необходимые другой единице работы. Чтобы избежать проблем, менеджер баз данных может сократить число страниц предварительной выборки по сравнению с заданным вами для табличного пространства.

Значение объема предварительной выборки может существенно влиять на производительность, особенно при просмотре больших таблиц. Для настройки параметра PREFETCHSIZE табличных пространств можно использовать системный монитор базы данных и другие системные инструменты мониторинга. Можно, например, собрать информацию и выяснить:

- Имели ли место ожидания ввода/вывода при обработке вашего запроса - при помощи инструментов мониторинга, доступных в вашей операционной системе.
- Происходила ли предварительная выборка - изучив элемент данных *pool_async_data_reads (асинхронные чтения данных пула буферов)*, поддерживаемый системным монитором баз данных. Дополнительную информацию смотрите в справочнике *System Monitor Guide and Reference*.

Если ожидания ввода/вывода имели место, и для запроса делалась предварительная выборка, можно попробовать увеличить значение PREFETCHSIZE. Может оказаться, что не процесс предварительной выборки вызывал ожидания ввода/вывода; в таком случае увеличение значения PREFETCHSIZE не повысит производительность для вашего запроса.

Во всех типах предварительной выборки параллельное выполнение нескольких операций ввода/вывода возможно, когда размер предварительной выборки является кратным размера экстента для табличного пространства, а экстенты табличного пространства находятся в отдельных контейнерах. Для лучшей производительности контейнеры нужно настроить на использование разных физических устройств. Дополнительную информацию о параллельной предварительной выборке смотрите в разделе “Настройка серверов ввода/вывода для предварительной выборки и параллельного ввода/вывода” на стр. 273.

Как работает обнаружение последовательного чтения

В некоторых случаях не очевидно, повысит ли производительность последовательная предварительная выборка. В этих случаях менеджер баз данных может отслеживать операции ввода/вывода и, обнаружив чтение последовательных страниц, активировать предварительную выборку. В этом случае предварительная выборка активируется и деактивируется менеджером баз данных по собственному усмотрению. Этот тип последовательной

предварительной выборки называется *обнаружением последовательного чтения* и применяется как для страниц индексов, так и для страниц данных. При помощи параметра конфигурации *seqdetect* (смотрите раздел “Флаг обнаружения последовательного чтения (seqdetect)” на стр. 415) можно управлять применением менеджером баз данных обнаружения последовательного чтения. Если обнаружение последовательного чтения включено, с помощью этого параметра можно определить использование последовательной предварительной выборки для следующего оператора SQL:

```
SELECT NAME FROM EMPLOYEE
WHERE EMPNO BETWEEN 100 AND 3000
```

В этом примере программа оптимизации может выбрать просмотр таблицы при помощи индекса для столбца EMPNO. Если таблица существенно кластеризована по отношению к этому индексу, чтение страниц данных окажется почти последовательным, и предварительная выборка может повысить производительность. В этом случае будет производиться предварительная выборка страниц данных.

В этом примере может также производиться предварительная выборка страниц индекса. Если приходится просматривать много страниц индекса, и менеджер баз данных обнаруживает последовательное чтение страниц индекса, будет производиться предварительная выборка страниц индекса.

Как работает предварительная выборка списка

Предварительная выборка списка, или предварительная выборка последовательности списка - способ эффективного доступа к страницам данных, даже если требуемые страницы данных не идут последовательно. Предварительная выборка списка может использоваться и при одноиндексном, и при многоиндексном доступе.

Предварительная выборка и внутриразделный параллелизм

Предварительная выборка имеет большое значение для производительности внутриразделного параллелизма, использующего несколько подагентов при просмотре индекса или таблицы. Эти параллельные просмотры расширяют масштабы обработки данных, что требует более мощной предварительной выборки.

Стоимость неадекватной предварительной выборки при параллельных просмотрах окажется выше, чем при последовательных просмотрах. Если предварительная выборка при выполнении последовательного просмотра не производится, замедление выполнения запроса будет связано с тем, что агенту придется каждый раз ожидать ввода/вывода. Если предварительная выборка не производится при выполнении параллельного просмотра, всем подагентам придется ожидать, пока один подагент ожидает ввода/вывода.

В связи с особой важностью при предварительной выборке для внутриразделного параллелизма применяется более агрессивная стратегия. Механизм обнаружения последовательного чтения позволяет считать последовательными даже такие соседние страницы, между которыми имеется значительный пропуск. Ширина допустимых пропусков растет вместе с числом подагентов, участвующих в просмотре.

Настройка серверов ввода/вывода для предварительной выборки и параллельного ввода/вывода

Чтобы обеспечить возможность предварительной выборки, менеджер баз данных запускает отдельные потоки управления - *серверы ввода/вывода*, выполняющие чтение страниц. В результате обработка запроса разделяется на две параллельных работы: обработку данных (процессор) и ввод/вывод страниц данных. Серверы ввода/вывода ожидают запросов на предварительную выборку, исходящих от обработки данных процессором. Эти запросы на предварительную выборку содержат описания операций ввода/вывода, которые должны удовлетворить ожидаемую потребность в данных. В зависимости от цели предварительной выборки менеджер баз данных решает, когда и какие запросы на предварительную выборку генерировать. (Дополнительную информацию смотрите в разделах “Анализ последовательной предварительной выборки” на стр. 270 и “Как работает предварительная выборка списка” на стр. 272.)

На следующем рисунке показано, как серверы ввода/вывода используются для предварительной выборки данных в пул буферов.

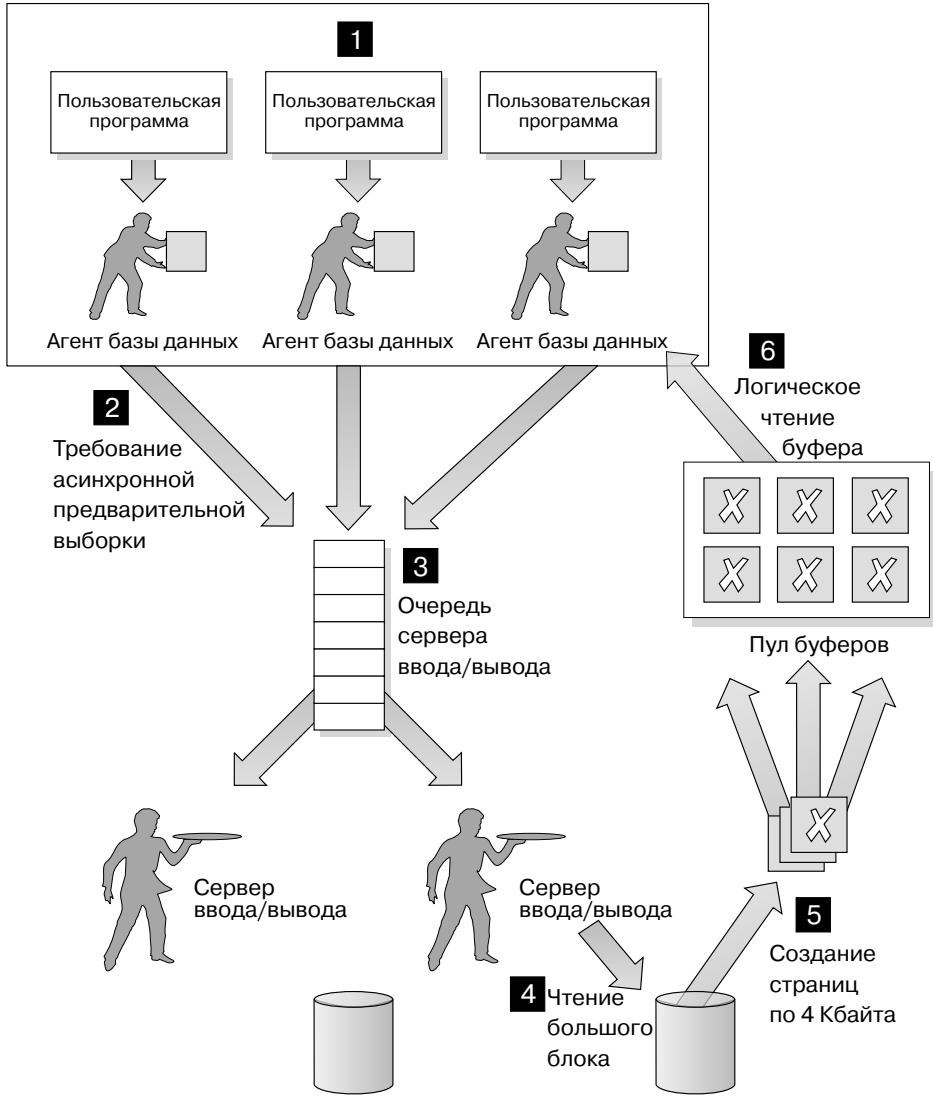


Рисунок 27. Предварительная выборка данных при помощи серверов ввода/вывода

В разделе рис. 27 описаны следующие шаги:

1 Пользовательская программа передает требование SQL агенту базы данных, который назначен ей менеджером баз данных.

2, **3** Агент базы данных определяет, что для удовлетворения запроса SQL нужно при получении требуемых данных использовать предварительную выборку, после чего записывает требование предварительной выборки в очередь серверов ввода/вывода.

4 , **5**

Первый доступный сервер ввода-вывода берет из очереди требование предварительной выборки и считывает данные из табличного пространства в пул буферов. В зависимости от числа требований предварительной выборки в очереди и числа серверов ввода/вывода, заданного параметром конфигурации *num_ioservers*, несколько серверов ввода/вывода могут одновременно выполнять предварительную выборку данных из табличного пространства.

6 Агент базы данных выполняет необходимые действия над страницами данных в пуле буферов, чтобы вернуть пользовательской программе результат требования SQL.

Задание достаточного числа серверов ввода/вывода при помощи параметра конфигурации *num_ioservers* может существенно повысить производительность запросов, допускающих применение предварительной выборки данных. Некоторое число избыточных серверов ввода/вывода не снизит производительности, поскольку избыточные серверы ввода/вывода не будут использоваться, и их страницы памяти будут выгружены. Все процессы серверов ввода/вывода нумеруются, и менеджер баз данных всегда использует доступный процесс с наименьшим номером; в результате возможно, что некоторые из процессов со старшими номерами остаются ни разу не использованными.

Чтобы определить, какое задать число серверов ввода/вывода, учтите следующие факторы:

- Количество одновременно активных процессов для базы данных. Речь идет о числе агентов базы данных, которые могут в одно и то же время записывать требования предварительной выборки в очередь серверов ввода/вывода.
- Наивысший уровень параллелизма при работе серверов ввода/вывода. Дополнительную информацию смотрите в разделе “Включение параллельного ввода/вывода”.

Чтобы максимизировать преимущества параллельного ввода/вывода, задайте значение *num_ioservers* не менее числа физических дисков базы данных.

Включение параллельного ввода/вывода

В ситуациях, когда у табличного пространства есть несколько контейнеров, менеджер баз данных может инициировать *параллельный ввод/вывод*. Параллельный ввод/вывод - это способность менеджера баз данных использовать несколько серверов ввода/вывода для обработки требований ввода/вывода из единой очереди. Каждый сервер обрабатывает рабочую нагрузку ввода/вывода для отдельного контейнера, что позволяет параллельно производить чтение нескольких контейнеров. Параллельная работа может существенно повысить пропускную способность ввода/вывода.

Пока рабочая нагрузка для каждого контейнера обрабатывается отдельным сервером, количество серверов, осуществляющих параллельный ввод/вывод, ограничено числом физических устройств, по которым распределены требуемые данные. Это означает также, что серверов ввода/вывода должно быть столько же, сколько физических устройств.

Способ инициации и использования параллельного ввода/вывода зависит от цели выполнения ввода/вывода:

- **Последовательная предварительная выборка**

  В случае последовательной предварительной выборки параллельный ввод/вывод инициируется, когда размер предварительной выборки кратен размеру экстента табличного пространства. В этом случае каждое требование на выполнение предварительной выборки разбивается на несколько меньших требований по границам экстентов. Затем эти меньшие требования исполняются разными серверами ввода/вывода.

- **Предварительная выборка списка**

  В случае предварительной выборки списка каждый список страниц разделяется на меньшие списки в соответствии с контейнером, в котором хранятся страницы данных. Эти меньшие списки затем поручаются отдельным серверам ввода/вывода.

- **Резервное копирование и восстановление базы данных и табличного пространства**

  При резервном копировании и восстановлении данных число параллельных требований ввода/вывода может изменяться от отношения размера буфера резервного копирования к размеру экстента (минимальное значение) до числа контейнеров, (максимальное значение).

- **Восстановление базы данных и табличного пространства**

  При восстановлении данных параллельные требования ввода/вывода инициируются и распределяются тем же способом, что и при последовательной предварительной выборке. Вместо восстановления данных с чтением их в пул буферов данные непосредственно переносятся из буфера восстановления на диск.

- **Загрузка**

  При загрузке данных задать уровень параллелизма ввода/вывода можно при помощи опции DISK_PARALLELISM команды LOAD. (Если она не задана, значение по умолчанию базируется на суммарном числе контейнеров всех табличных пространств, связанных с таблицей.)

Чтобы оптимизировать параллельный ввод/вывод и добиться максимальной производительности, убедитесь, что:

- Есть достаточное число серверов ввода/вывода. Число серверов ввода/вывода следует задать несколько большим, чем число контейнеров, используемых для всех табличных пространств базы данных.

- Для табличного пространства важны размер экстента и размер предварительной выборки. Размер предварительной выборки не должен быть слишком большим, чтобы не занимать пул буферов сверх необходимого. (Идеальный размер - произведение размера экстента на число контейнеров табличного пространства.) Размер экстента должен быть достаточно мал, разумно задавать значение от 8 до 32 страниц.
- Контейнеры настраивают так, чтобы они располагались на разных физических дисках.
- Все контейнеры должны быть одного размера, чтобы не нарушать согласованность при параллелизме.

  Если один или несколько контейнеров окажутся меньше, чем другие, это нарушит оптимальность параллельной предварительной выборки. Например:

  – После того, как меньший контейнер заполнится, новые данные будут записываться в остающиеся контейнеры, из-за чего контейнеры окажутся несбалансированны. Несбалансированные контейнеры снижают производительность параллельной предварительной выборки, поскольку число контейнеров, из которых возможна предварительная выборка данных, может оказаться меньше общего числа контейнеров.
  – Если меньший контейнер добавлен позже и баланс данных восстановлен, в меньшем контейнере будет меньше данных, чем в других контейнерах. Этот малый по сравнению с другими контейнерами объем данных не позволит оптимизировать параллельную предварительную выборку.
  – Если один из контейнеров имеет больший размер, и все остальные контейнеры заполнены, он останется единственным контейнером для новых данных. Менеджер баз данных не сможет использовать параллельную предварительную выборку для доступа к этим данным.
- Существует мощность ввода/вывода, адекватная внутриразделному параллелизму. На компьютерах SMP внутриразделный параллелизм позволяет уменьшить время обработки запроса за счет выполнения запроса на нескольких процессорах. Чтобы обеспечить работой все процессоры, нужна достаточная мощность ввода/вывода; обычно для обеспечения требуемой мощности ввода/вывода требуются дополнительные физические диски.

  Для эффективного использования мощности ввода/вывода нужно ускорить предварительную выборку. Чтобы ускорить предварительную выборку, нужно увеличить размер предварительной выборки. Размер предварительной выборки должен быть произведением размера экстента на число контейнеров табличного пространства. В идеале контейнеры нужно настроить так, чтобы они располагались на разных физических дисках.

  Требуемое число физических дисков может зависеть от скорости и мощности дисков и шины ввода/вывода и от мощности процессоров.

Размещение нескольких страниц за раз

Табличные пространства SMS расширяются по требованию. По умолчанию это расширение производится по одной странице за раз. Однако при некоторых

видах работы (например, при вставке большого объема данных) можно повысить производительность, если при помощи инструмента *db2empfa* сообщить DB2, что надо расширять табличное пространство группами страниц или экстентами. Инструмент *db2empfa* находится в подкаталоге *bin* каталога *sqllib*. Его запуск задает для параметра конфигурации базы данных *multipage_alloc* значение Yes. Дополнительную информацию об этом инструменте смотрите в руководстве *Command Reference*.

Еще один способ оптимального использования доступной памяти описан в разделе “Расширение памяти” на стр. 296.

Сортировка

Многие запросы требуют сортировки, и правильная настройка областей кучи сортировки имеет большое значение для эффективного выполнения запроса. Сортировка требуется, когда:

- Нет индекса, удовлетворяющего требуемому упорядочиванию (например, оператору SELECT с условием ORDER BY)
- Индекс есть, но сортировка эффективнее, чем его использование
- Выполняется создание индекса (если для параметра конфигурации *indexsort* задано значение yes).

Разные типы сортировки

Сортировка состоит из двух этапов:

1. Фаза сортировки
2. Возврат результатов фазы сортировки.

Можно выделить несколько категорий, или типов сортировки, исходя из методов, используемых на двух этапах сортировки. На фазе сортировки различают категории сортировок “с переполнением” и “без переполнения”. На фазе возврата результата сортировки различают категории сортировок “конвейерных” и “неконвейерных”.

С переполнением и без переполнения

Если сортируемая информация не уместилась в куче сортировки (блоке памяти, отводимом при каждом выполнении сортировки), она попадает во временные таблицы базы данных. У сортировок, выполняемых без переполнения памяти, производительность всегда выше, чем у сортировок с переполнением.

Конвейерные и неконвейерные

Сортировку, при которой возможен прямой возврат отсортированной информации и не требуется сохранения конечного, отсортированного списка данных во временной таблице, называют “конвейерной сортировкой”. Сортировку, для возврата отсортированной информации

которой требуется временная таблица, называют “неконвейерной сортировкой”. Производительность конвейерной сортировки всегда выше, чем у неконвейерной.

Настройка параметров, влияющих на сортировку

На производительность сортировки влияют следующие факторы:

- Настройка следующих параметров конфигурации:

  “Размер кучи сортировки (sortheap)” на стр. 381
  Задает объем памяти для каждой сортировки

  “Порог кучи сортировки (sheapthres)” на стр. 382
  Управляет общим объемом памяти для сортировки, доступным в экземпляре для всех сортировок.

- Операторы, предполагающие большой объем сортировки
- Отсутствие индексов, которые могли бы предотвратить лишнюю сортировку
- Логика прикладной программы, не минимизирующая сортировку
- Параллельная сортировка, которая повышает производительность сортировок, но возможна только при использовании внутриразделного параллелизма (смотрите раздел “Включение параллельного ввода/вывода” на стр. 275).

Индикаторы проблем с производительностью сортировки

Чтобы понять, есть ли у вас вообще проблемы с сортировкой, посмотрите, каково общее время, затраченное процессором на сортировку, по сравнению с временем, затраченным на всю прикладную программу. Здесь может помочь системный монитор базы данных (смотрите раздел “Использование системного монитора базы данных” на стр. 293). В частности, Монитор производительности (который состоит из “монитора снимков” и “монитора событий” и доступен из Центра управления), показывает по умолчанию *общее время сортировки*, а также время *ввода/вывода* и *ожидания блокировок*.

Если общее время сортировки значительно по сравнению с другими затратами времени, просмотрите следующие значения, которые также выводятся по умолчанию:

Процент сортировок с переполнением
Эта переменная (в окне подробностей производительности Монитора снимков) показывает процент сортировок с переполнением памяти. Если процент сортировок с переполнениями велик, увеличьте параметры конфигурации *sortheap* и/или *sheapthres*, если были постпороговые сортировки. (Чтобы определить, были ли постпороговые сортировки, используйте Монитор снимков.)

Послепороговые сортировки
Если постпороговых сортировок много, увеличьте *sheapthres* и/или уменьшите *sortheap*.

В общем случае доступную в экземпляре общую память сортировки (*sheapthres*) следует сделать как можно большей, но без лишних страниц. Можно добиться, чтобы вся сортировка происходила в памяти сортировки. Но если операционная система, выполняя требование к памяти сортировки, слишком часто начнет записывать страницы подкачки, преимущество большой кучи сортировки может быть потеряно. Поэтому при каждом изменении параметров конфигурации сортировки нужно при помощи монитора операционной системы проследить, как изменилась системная подкачка страниц.

Примечание: После усовершенствования метода DB2 для двоичной сортировки частичных ключей, который допускает теперь нецелочисленные типы ключей, возникла потребность в дополнительной памяти при сортировке длинных ключей. Если вы полагаете, что используются длинные ключи, увеличьте параметр конфигурации *sortheap*.

Заметьте также, что при конвейерной сортировке куча сортировки не освобождается, пока прикладная программа не закроет указатель, связанный с этой сортировкой. Поэтому конвейерная сортировка может продолжать использование памяти до закрытия указателя.

Методы управления производительностью сортировки

При подборе параметров конфигурации *sortheap* и *sheapthres* можно использовать системный монитор базы данных и методы тестовых измерений. Выполните следующие действия для каждого менеджера баз данных и его баз данных:

- Установите и запустите характерную типовую работу.
- Для каждой нужной базы данных соберите средние значения следующих переменных производительности за период характерной тестовой нагрузки:
  - Вся куча сортировки в использовании
  - Число активных сортировок

  Эти переменные производительности выводятся в окне подробный вид производительности Монитора снимков.
- Для каждой базы данных задайте параметр *sortheap* равным среднему *общему использованному объему кучи сортировки*.
- Настройте параметр *sheapthres*, для чего:
  1. Определите, у какой из баз данных экземпляра наибольшее значение параметра *sortheap*.
  2. Определите для этой базы данных средний размер кучи сортировки.

     Если этот размер определить трудно, примите 80% от максимальной кучи сортировки
  3. Задайте параметр *sheapthres* равным среднему числу активных сортировок, умноженному на рассчитанный выше средний размер кучи сортировок.

Это рекомендуемое начальное значение. В дальнейшем с помощью техники тестовых измерений можно уточнить его.

Кроме того, можно выявить конкретные прикладные программы и операторы, где сортировка существенно влияет на производительность:

- Задайте мониторы событий на уровне программ и операторов, чтобы выявить программы с наибольшим общим временем сортировки.
- В каждой из этих программ найдите операторы с наибольшим *общим временем сортировки*.
- Настройте эти операторы, используя такие инструменты, как Visual Explain.
- Убедитесь, что построены нужные индексы. При помощи Visual Explain можно выявить все операции сортировки для данного оператора. Затем выясните, существует ли нужный индекс для каждой таблицы, запрашиваемой этим оператором.

Примечание: Можно просмотреть таблицы объяснения и выяснить, в каких запросах выполняются операции сортировки. (Смотрите раздел “Приложение C. Инструменты объяснения SQL” на стр. 591.)

Реорганизация каталогов и пользовательских таблиц

Производительность операторов SQL, использующих индексы, снижается после большого числа изменений, удалений и вставок. В общем случае вновь вставляемые строки не могут попасть физически в то место в последовательности строк, которое они занимают логически согласно индексу (кроме случая кластеризованных индексов). Это значит, что менеджер баз данных для доступа к данным должен выполнять дополнительные операции чтения, так как логически последовательные данные могут оказаться на разных физических страницах данных, вовсе не последовательных.

В общем случае реорганизация таблицы занимает больше времени, чем запуск статистики. Возможно, достаточного повышения производительности удастся добиться, получив текущую статистику для данных и повторно связывав прикладных программ, так что попробуйте начать с этого. Если это не повысит производительности, возможно, данные в таблицах и индексах упорядочены не эффективно, и стоит попробовать реорганизацию. Это относится не только к вашим собственным таблицам, но и к таблицам системного каталога, которым тоже может потребоваться реорганизация.

Для типизированных таблиц надо задавать имя корневой таблицы иерархии.

Команда REORGCHK возвращает информацию о физических характеристиках таблицы и сообщает, будет ли выигрыш от реорганизации этой таблицы. Эту

команду можно ввести через процессор командной строки. Дополнительную информацию о команде, включая интерпретацию результатов, смотрите в руководстве *Command Reference*.

Примечание: Команда REORGCHK не выдает никакой информации для расширенных индексов и для объявленных временных таблиц.

Утилита REORG может упорядочивать данные в физическую последовательность согласно заданному индексу. У REORG есть опция, задающая порядок строк в таблице по индексу, что кластеризует данные таблицы согласно индексу и улучшает статистические показатели CLUSTERRATIO или CLUSTERFACTOR, собираемые утилитой RUNSTATS. В результате операторы SQL, которым строки требуются в порядке индекса, могут обрабатываться более эффективно. REORG также переписывает таблицы в более компактном виде, удаляя неиспользуемое пустое пространство (хотя это пространство останется неиспользованным, если при использовании ALTER TABLE задавалось PCTFREE).

Не используйте в командах REORG и REORGCHK псевдонимы.

Утилита REORG требует отключения всех других программ, которые обычно работают с данными и индексами реорганизуемых таблиц. Возможно, в вашей среде желательно свести к минимуму время, в течение которого программы не могут работать с данными. В такой среде может быть предпочтительна утилита фоновой реорганизации индекса.

Пространство журнала, необходимое для перестроения индекса во время реорганизации, рассчитывается по формуле:

```
2 * (10500 + ((число страниц индекса / размер экстента) * 110) +
(число страниц индекса * 45) +
(число страниц индекса / 16000) * 64 ))
```

Различные слагаемые в этой формуле соответствуют различным типам накладных расходов, связанных с созданием данных и их записью в журнал при перестроении индексов.

Выбирая момент для реорганизации данных таблицы, вы можете учесть следующие возможные поводы:

- Большой объем вставок, изменений и удалений
- Существенное изменение производительности запросов, использующих индекс с высоким кластерным отношением
- То, что запуск статистики (RUNSTATS) не повысил производительности запросов
- То, что команда REORGCHK указала на необходимость реорганизовать таблицу

- Стоимость реорганизации таблицы, включая время процессора, время обработки и снижение текущей результативности из-за блокировки таблицы утилитой REORG до завершения реорганизации.

Для выполнения утилиты REORG вам потребуются полномочия SYSADM, SYSMAINT, SYSCTRL или DBADM или привилегия CONTROL для таблицы.

Утилита REORG использует временные таблицы, которые могут оказаться существенно больше исходной таблицы, если к ней добавлялись столбцы или в ней имелись столбцы больших объектов. Если эти временные таблицы больше, чем исходная, то полученная постоянная таблица, созданная утилитой REORG, также окажется больше.

Примечание: Нельзя использовать утилиту REORG для реорганизации объявленных временных таблиц.

При вызове утилиты REORG можно указать временное табличное пространство для временной таблицы REORG. Если не задать временное табличное пространство, утилита REORG будет создавать временные таблицы REORG в том табличном пространстве, где находится реорганизуемая таблица. Следующие указания могут помочь определить, стоит ли использовать временное табличное пространство:

- Если вы задаете временное табличное пространство, обычно рекомендуется задать временное табличное пространство SMS. Не рекомендуется использовать временное табличное пространство DMS, поскольку с помощью этого типа табличного пространства можно выполнять только один процесс REORG.
- Обычно рекомендуется задать временное табличное пространство SMS. Использование того же табличного пространства быстрее, однако приводит к большему объему записи в журнал и требует достаточного места для реорганизованной таблицы. Если вы задаете временное табличное пространство, обычно рекомендуется задать временное табличное пространство SMS. Не рекомендуется использовать временное табличное пространство DMS, поскольку с помощью этого типа табличного пространства можно выполнять только один процесс REORG.

Утилита REORG неявно закрывает все открытые указатели.

Помните, что в реорганизуемой таблице может использоваться размер страниц больше 4 Кбайт (8, 16 или 32 Кбайта). Временное табличное пространство, используемое при реорганизации, должно иметь тот же размер страниц, что и исходное табличное пространство.

Если утилита REORG не завершилась успешно, **не удаляйте** временные файлы, таблицы и табличные пространства. Эти файлы и таблицы используются

менеджером баз данных для отката сделанных утилитой REORG изменений или для завершения реорганизации, если процесс успел значительно продвинуться до сбоя.

В распределенной базе данных утилита REORG реорганизует данные в каждом разделе. При сбое утилиты откат затронет только тот раздел, в котором произошел сбой. Если вы зададите каталог для хранения временных таблиц, в указанном путь менеджер баз данных создаст подкаталоги для каждого раздела базы данных. Таким образом, если вы зададите каталог, общий для разделов базы данных, временные файлы будут храниться в разных подкаталогах (различаемых по имени узла) указанного каталога.

Фоновая реорганизация индекса

Фоновая реорганизация возможна за счет задания пользовательского порога для максимального объема свободного пространства на конечной странице индекса. Если при удалении индексного ключа на последней странице оказывается превышен порог, проверяются соседние конечные страницы индекса и выясняется, можно ли слить две конечные страницы. Если на странице оказывается достаточно пространства для слияния двух соседних страниц, слияние происходит без отключения базы данных.

Фоновая реорганизация индекса возможна только для индексов, созданных в Версии 6 и последующих выпусках. Если надо обеспечить такую возможность для существующих индексов, надо отбросить их и воссоздать снова, чтобы произвести необходимые внутренние изменения на конечных страницах индекса. Чтобы включить фоновую реорганизацию для конкретного индекса, при создании индекса задайте значение `MINPCTUSED`. Значение `MINPCTUSED` должно быть меньше ста (100). Это значение - порог реорганизации, равный проценту используемого на индексной странице пространства - минимальному приемлемому значению, ниже которого надо предпринимать попытки слить конечную страницу индекса с соседней. Рекомендуется задавать `MINPCTUSED` меньше 50 процентов, поскольку целью является слияние двух соседних конечных страниц. Нулевое значение `MINPCTUSED`, принимаемое по умолчанию, отключает фоновую реорганизацию.

Конечные страницы индекса, освобождаемые в ходе фоновой реорганизации индекса, становятся доступными для повторного использования. Но пользоваться освобожденными страницами могут только другие индексы той же таблицы. Полная реорганизация таблицы освободит страницы для других объектов при работе с моделью памяти DMS и освободит пространство диска при работе с моделью памяти SMS.

Промежуточные страницы индекса при фоновой реорганизации индекса не освобождаются. Однако при полной реорганизации таблицы индекс уменьшится до минимально возможного размера. Уменьшается число конечных и промежуточных страниц, а также число уровней индекса.

Ограничение потребности в реорганизации таблицы

Чтобы уменьшить потребность в реорганизации таблицы, выполните после создания таблицы следующие действия:

- Измените таблицу, добавив PCTFREE
- Создайте индекс кластеризации с PCTFREE для индекса
- Отсортируйте данные
- Загрузите данные.

После одной из этих операций над существующей таблицей у вас будет таблица с индексом кластеризации. Индекс кластеризации с PCTFREE для таблицы будет поддерживать первоначальный отсортированный порядок. При достаточном пространстве на страницах возможна вставка новых данных на правильные страницы, сохраняющая характеристики кластеризации индекса кластеризации. Если по мере вставки все новых данных страницы таблицы начнут заполняться, новые записи будут добавляться в конец таблицы, и таблица будет становиться все менее кластеризованной.

Рекомендуется после создания индекса кластеризации выполнить REORG или сортировку. Индекс кластеризации пытается поддерживать определенный порядок данных, что улучшает статистику CLUSTERRATIO и CLUSTERFACTOR, собираемую утилитой RUNSTATS.

Объем свободного пространства, который будет оставлен на каждой странице при выполнении REORG, определяется значением PCTFREE таблицы. Если это значение не задано, REORG заполнит страницы по мере реорганизации данных.

Особенности производительности для устройств DMS

Если для табличных пространств вы используете контейнеры устройства памяти, управляемой базой данных (Database Managed Storage, DMS), для эффективного управления средой надо знать следующее:

- **Кэширование файловой системы**

  Кэширование файловой системы работает так:

  – Для контейнеров файла DMS (и всех контейнеров SMS) операционная система может кэшировать страницы в кэше файловой системы

  – Для табличных пространств устройства DMS операционная система не кэширует страницы в кэше файловой системы.

  Примечание: При работе в Windows NT переменная реестра DB2NTNOCACHE задает, будет ли DB2 открывать файлы базы данных с опцией NOCACHE. Если DB2NTNOCACHE=ON, кэширование файловой системы исключается. Если DB2NTNOCACHE=OFF, операционная система кэширует файлы DB2. Это относится ко всем

данным, кроме файлов, содержащих длинные поля и большие объекты. Устранение системного кэширования освобождает больше памяти для базы данных, что позволяет увеличить пул буферов или кучу сортировки.

- **Буферизация данных**

  Табличные данные, прочитанные с диска, обычно доступны в пуле буферов базы данных (смотрите раздел “Управление пулом буферов базы данных” на стр. 261). В некоторых случаях страница данных в пуле буферов может быть освобождена до того, как ей воспользовалась прикладная программа. (Это может случиться, если пространство в пуле буферов потребовалось для других страниц данных.) Для табличных пространств, использующих контейнеры файлов SMS или DMS, смотрите приведенное выше описание кэширования файловой системы. Это кэширование способно устранять лишние операции ввода/вывода.

  Табличные пространства, использующие контейнеры устройств DMS, **не используют** файловую систему и ее кэш. В результате вы можете пожелать увеличить размер пула буферов базы данных и уменьшить размер кэша файловой системы, чтобы компенсировать то обстоятельство, что двойная буферизация не делается с помощью табличных пространств DMS, использующих контейнеры устройств.

  Как вы могли заметить при помощи инструментов мониторинга системного уровня, ввод/вывод интенсивнее в случае табличного пространства DMS, использующего контейнеры устройств, чем в случае эквивалентного табличного пространства SMS, и это различие может быть связано с описанной выше двойной буферизацией.

- **Использование данных типа LOB и LONG**

  Когда прикладная программа извлекает данные типа LOB или LONG, менеджер баз данных не использует свои буферы для кэширования данных. Каждый раз, когда программе понадобится одна из страниц таких данных, менеджер баз данных должен извлекать ее с диска.

  Но если данные LOB или LONG хранятся в файловых контейнерах SMS или DMS, кэширование файловой системы может обеспечить буферизацию и, в результате, лучшую производительность.

  Поскольку несколько столбцов LOB содержится в системных каталогах, рекомендуется хранить их в табличных пространствах SMS (или в файле DMS).

Управление расходами на инициализацию

Команда ACTIVATE DATABASE запускает выбранные базы данных. Применение этой команды для распределенной базы данных приводит к попытке активировать выбранную базу данных во всех ее разделах. Благодаря этой команде прикладная программа не тратит времени на инициализацию или запуск баз данных.

Базы данных, инициализированные при помощи команды ACTIVATE DATABASE, должны быть закрыты при помощи команды DEACTIVATE DATABASE; база данных не закроется сама при отключении последней программы. Дополнительную информацию о командах ACTIVATE и DEACTIVATE смотрите в руководстве *Command Reference*.

Если база данных не была запущена, а в программе встретился оператор CONNECT TO (или неявное требование соединения), прежде чем программа сможет работать с требуемой базой данных, ей придется ожидать, пока менеджер баз данных запустит эту базу данных. Это стоимость запуска, которую платит программа, первой обратившаяся к конкретной базе данных. В распределенной базе данных все это происходит в каждом из разделов. После запуска базы данных остальные прикладные программы могут соединяться и работать с базой данных без затрат времени на запуск базы данных.

Агенты базы данных

Серверы DB2 должны упрощать связь между менеджером баз данных и клиентом и локальными программами. Среды на основе UNIX используют архитектуру на базе *процессов*. Например, принимающие программы связи DB2 создаются как процессы. Операционные системы Intel, такие как OS/2 и Windows NT, используют архитектуру, базирующуюся на *потоках*, чтобы повысить производительность. Например, принимающие программы связи DB2 создаются как потоки внутри процесса системного контроллера сервера DB2. Для каждой запрашиваемой базы данных запускаются различные процессы/потоки, выполняющие разные работы с базой данных (например, предварительную выборку, связь, ведение журнала).

Важные процессы (потоки) - агенты базы данных, облегчающие прикладным программам операции с базами данных.

Логический агент представляет подключенную программу в менеджере баз данных. Логический агент содержит все информационные и управляющие блоки, необходимые прикладной программе. Максимальное число логических агентов определяется параметром конфигурации *max_logicagents* менеджера баз данных. Поскольку для каждой прикладной программы используется один логический агент, этот параметр определяет максимальное число прикладных программ, которые могут соединиться с экземпляром.

Рабочий агент выполняет запросы прикладной программы, но не приписан постоянно к конкретной программе. Рабочий агент содержит все информационные и управляющие блоки, необходимые для выполнения в менеджере баз данных действий, которые запрашивает прикладная программа.

Существует четыре типа рабочих агентов: *агенты активного координатора*, *подагенты*, *неактивные агенты* и *свободные агенты*.

Свободный агент - простейший вид рабочего агента. Он не связан с логическим агентом, у него нет исходящего соединения, подключения к локальной базе данных или подсоединения к экземпляру.

Неактивный агент - это рабочий агент, не участвующий в активной транзакции, не связанный с логическим агентом, не имеющий исходящего соединения и не имеющий подключения к локальной базе данных или подсоединения к экземпляру. Неактивный агент свободен связаться с другим логическим агентом и начать работать на прикладную программу, представляемую этим логическим агентом.

У каждого процесса (потока) клиентской программы есть один *агент активного координатора*, работающий с базой данных. После того, как агент координатора будет создан, он начинает выполнять все требования к базе данных для своей прикладной программы и связываться с другими агентами при помощи межпроцессовой связи (inter-process communications, IPC) или протоколов удаленной связи. Каждый агент работает в своей памяти, и все агенты совместно используют глобальные ресурсы менеджера баз данных и базы данных, в частности, пул буферов. По завершении транзакции активный агент координатора может отсоединиться от логического агента и тем самым стать неактивным агентом.

В средах распределенных баз данных и средах с разрешенным внутриразделным параллелизмом агент координатора распределяет запросы к базе данных между *подагентами*, а те выполняют запросы для программ. После того, как агент координатора будет создан, он начинает обрабатывать все требования к базе данных для своей прикладной программы и координировать работу подагентов, выполняющих требования к базе данных.

Когда клиент отключится от базы данных или отсоединится от экземпляра, координирующий агент перейдет в одно из следующих состояний:

- Активный агент. Если другие логические агенты находятся в состоянии ожидания, рабочий агент станет активным агентом координатора.
- Свободный агент, если нет других ожидающих логических агентов и не достигнуто максимальное число агентов пула.
- Будет завершен с освобождением памяти, если нет других ожидающих логических агентов и достигнуто максимальное число агентов пула.

Агенты, не выполняющие работу для каких-либо прикладных программ и ожидающие назначения, считаются свободными агентами и находятся в *пуле агентов*. Эти агенты доступны для запросов со стороны агентов координатора, работающих для клиентских программ, или для подагентов, работающих для существующих агентов координатора. Число доступных агентов зависит от параметров конфигурации менеджера баз данных *maxagents* и *num_poolagents*.

Агенты из пула агентов (*num_poolagents*) повторно используются как агенты координатора:

- Для удаленных программ на основе TCP/IP, или
- Для локальных программ в операционных системах на основе UNIX или
- Для локальных и удаленных программ в операционных системах Windows NT и OS/2.

В остальных случаях удаленные программы создают новый агент.

Если агент требуется, когда нет ни одного свободного агента, новый агент должен быть создан динамически. Создание нового агента сопряжено с некоторыми накладными расходами, так что производительность CONNECT и ATTACH заметно выше, если существует свободный агент, который может быть активирован для клиента.

Когда подагент работает для прикладной программы, он считается *ассоциированным* с этой программой. По завершении назначенной работы агент может быть возвращен в пул агентов, но остается ассоциированным с первоначальной прикладной программой. Если прикладной программе потребуется дополнительная работа, менеджер баз данных при подборе агента вначале проверяет, есть ли в пуле агентов ассоциированные агенты.

Возможность отдельно регулировать число подключенных программ (при помощи числа логических агентов, определяемого параметром *max_logicagents*) и число обрабатываемых запросов программ (при помощи числа активных агентов координатора, определяемого параметром *max_coordagents*) повышает гибкость управления нагрузкой на базу данных. В типичном случае работы программ с базой данных есть взаимно-однозначное соответствие между числом подключенных программ и числом обрабатываемых запросов. Однако бывает, что в рабочей среде на каждый обрабатываемый запрос программ приходится по несколько подключенных программ.

Поскольку служебные расходы глобальных ресурсов базы данных связаны с активными агентами координатора, получается, что чем больше число этих агентов, тем больше вероятность достижения верхних пределов допустимых глобальных ресурсов базы данных. Может оказаться желательным иметь больше подключенных программ, чем активных агентов координатора, чтобы не достигать верхних пределов доступных глобальных ресурсов базы данных. Задавая значение *max_logicagents* больше значения *max_coordagents*, вы концентрируете работу базы данных.

Дополнительную информацию и примеры использования DB2 Connect в качестве концентратора поддержки транзакций XA смотрите в руководстве *DB2 Connect. Руководство пользователя*.

При работе в среде, требующей использования DB2 Connect для связи с удаленными системами, создается *пул исходящих соединений*. Этот пул связи уменьшает время соединения (когда первое соединение уже выполнено) с хостом. При требовании отключиться от хоста DB2 Connect завершает входящее соединение, но сохраняет исходящее соединение с хостом в пуле. При поступлении нового требования на соединение с хостом DB2 Connect повторно использует существующее исходящее соединение из пула (если оно доступно).

Примечание: При использовании пула связи DB2 Connect ограничен входящими соединениями TCP/IP и исходящими соединениями TCP/IP и SNA. При работе с SNA тип защиты должен быть NONE, чтобы соединение было помещено в пул.

При использовании пула связи активный агент не закрывает свое исходящее соединение после отключения, а переходит в пул агентов с активным соединением с удаленным хостом. Такой агент называется *неактивным агентом DRDA*. Пул неактивных агентов DRDA - это синоним пула исходящих соединений. “DRDA” - это “Distributed Relational Database Architecture” (архитектура распределенных реляционных баз данных).

Рассмотрим следующие примеры, основанные на четырех различных применениях и требованиях рабочей нагрузки:

1. В первом примере в среднем 40 пользователей одновременно соединяются с удаленными базами данных хоста через DB2 Connect. Временами число одновременных соединений поднимается примерно до 50, но заведомо не превышает 55. Транзакции длятся недолго, и пользователи часто подключаются и отключаются.

   В таких условиях системный администратор должен задать параметр *max_coordagents* равным 55, поскольку известно, что число пользователей, которые одновременно попытаются соединиться при помощи DB2 Connect, составит максимум 55. Размер пула агентов *num_poolagents* следует задать равным 40 - среднему числу одновременно подключенных или пытающихся подключиться пользователей. Такой размер пула гарантирует, что существующего числа соединений с удаленной базой данных достаточно для удовлетворения всех входящих клиентов без необходимости устанавливать новые, кроме моментов пиковой нагрузки.
2. Во втором примере нагрузка намного выше, около 1000 входящих клиентов. Соединения пользователей также кратковременные. Системный администратор не хочет допустить большего числа одновременных соединений. Поэтому он задает для обоих параметров *max_coordagents* и *num_poolagents* значение 1000. Это означает, что максимальное число входящих клиентов, которые смогут одновременно соединиться с удаленной базой данных, будет равно 1000. Когда все клиенты отключатся, в пуле будет ровно 1000 подключенных агентов, готовых обслужить новых входящих клиентов.

3. В третьем примере имеется одна прикладная программа, соединяющаяся при помощи DB2 Connect с одной удаленной базой данных. Программа остается подключенной в течение длительного времени. В этом сценарии оптимальная конфигурация пула агентов и пула связи - задать *max_coordagents* равным 1, поскольку соединяться будет заведомо только один клиент. *num_poolagents* в этом случае можно задать нулевым, поскольку не происходит частых подключений и отключений в отношении удаленного хоста. Задание *num_poolagents* равным нулю фактически выключает пул связи, поскольку в этом пуле не хранится ни одного агента с активным соединением с удаленной базой данных. Для каждого нового входящего клиента, требующего подключения, создается новый агент и устанавливается новое удаленное соединение для его обслуживания.
4. Четвертый пример - вариант, сочетающий черты трех предыдущих сценариев нагрузки. В этом примере системный администратор желает ограничить одновременный доступ к удаленным базам данных на уровне ровно 100. Поэтому *max_coordagents* задается равным 100 и, чтобы максимизировать производительность связи, *num_poolagents* также задается равным 100. Однако в дальнейшем может понадобиться локальное соединение для мониторинга рабочей нагрузки в системе, где установлена DB2 Connect. Ожидается, что одновременно будет делаться не более 5 снимков монитора, поэтому *max_coordagents* задается равным 105. Это новое значение параметра конфигурации допускает повышение максимального числа одновременных программ сверх прежнего верхнего предела 100, чтобы учесть эпизодические снимки монитора и/или подключение экземпляра.

В средах распределенных баз данных и средах, где разрешен внутриразделный параллелизм, у каждого раздела (то есть каждого сервера баз данных или узла) есть свой пул агентов, из которого извлекаются подагенты. Благодаря этому пулу не приходится создавать и уничтожать подагенты всякий раз, когда потребуется подагент или когда он завершит работу. Подагенты могут оставаться в пуле ассоциированными агентами и менеджер баз данных может использовать их для новых запросов со стороны прикладной программы, с которой они ассоциированы.

На число агентов базы данных влияют следующие параметры конфигурации менеджера баз данных:

- “Максимальное число агентов (maxagents)” на стр. 425. После того, как число рабочих агентов достигнет этого значения, все последующие требования, нуждающиеся в новом агенте, будут отклоняться, пока число агентов не упадет ниже этого значения. Это значение применяется к общему числу агентов, включая агенты координатора , подагенты, неактивные агенты и свободные агенты, работающие для всех прикладных программ.
- “Размер пула агентов (num_poolagents)” на стр. 428. Число неактивных агентов, свободных агентов и ассоциированных подагентов в пуле агентов не может превысить этого значения.

- “Начальное число агентов в пуле (num_initagents)” на стр. 430. При запуске менеджера баз данных пул рабочих агентов создается на основе этого значения. Это повышает производительность для начальных запросов. Рабочие агенты все начинают как свободные агенты.
- “Максимальное число логических агентов (max_logicagents)” на стр. 428. Максимальное число логических агентов. Поскольку для каждой прикладной программы используется один логический агент, этот параметр определяет максимальное число прикладных программ, которые могут соединиться с экземпляром.
- “Максимальное число координирующих агентов (max_coordagents)” на стр. 427. В средах с разделенными базами данных и средах с разрешенным внутриразделльном параллелизмом это значение ограничивает число координирующих агентов.
- “Максимальное число одновременных агентов (maxcagents)” на стр. 426. Это значение управляет числом *маркеров*, допускаемым менеджером баз данных. Для каждой транзакции (единицы работы) с базой данных, производимой в то время, когда клиент подключен к базе данных, координирующий агент должен получить от менеджера баз данных разрешение на обработку транзакции (называемое также маркером обработки). Только агенты с маркером обработки допускаются менеджером баз данных к выполнению единицы работы для базы данных. Если маркер недоступен, агент будет ожидать, пока не появится доступный маркер, и только тогда начнется обработка требуемой единицы работы.

  Этот параметр может быть полезен в среде, в которой запросы пиковой нагрузки превышают системные ресурсы (память, процессор и диск). В такой среде пиковая нагрузка может вызывать дополнительное ухудшение производительности, например, из-за подкачки страниц. Этот параметр может ограничить нагрузку и предотвратить падение производительности, хотя это может повлиять на одновременность и на время ожидания.

В средах с разделенными базами данных и средах с разрешенным внутриразделльным параллелизмом влияние на производительность и стоимость памяти в системе сильно зависит от настройки пула агентов:

- Такой параметр конфигурации менеджера баз данных, как размер пула агентов (*num_poolagents*), влияет на общее число подагентов, которые могут оставаться ассоциированными с прикладными программами в разделе (то есть на узле). Если размер пула окажется слишком мал (и пул будет заполнен), подагент отсоединится от программы, для которой работал, и завершится. Это ведет к ухудшению производительности, поскольку приходится все время создавать подагенты и повторно ассоциировать их с программами.

  Кроме того, если значение *num_poolagents* слишком мало, одна-единственная прикладная программа может заполнить пул своими ассоциированными подагентами. В результате, когда другой программе потребуется новый

подагент, и у нее не окажется ни одного подагента в ее ассоциированном пуле агентов, она “захватит” подагентов пулов агентов других прикладных программ. Такая ситуация приводит к высоким расходам и к снижению производительности.

- Описанные ситуации следует сопоставить со стоимостью ресурсов для поддержания большого числа агентов, активных одновременно.

  Так, если значение *num_poolagents* слишком велико, ассоциированные агенты могут подолгу находиться в пуле, не работая. Эти подагенты будут занимать ресурсы менеджера баз данных, не давая использовать их для других задач.

Помимо агентов базы данных, менеджер баз данных выполняет и другие асинхронные работы, запускающие собственный процесс (или поток), включая:

- Серверы ввода/вывода баз данных (занимаются предварительной выборкой ввода/вывода) (смотрите раздел “Предварительная выборка данных в пул буферов” на стр. 269)
- Асинхронные чистильщики страниц базы данных (смотрите раздел “Управление пулом буферов базы данных” на стр. 261)
- Регистраторы базы данных
- Детекторы тупиковых ситуаций
- Мониторы событий
- Принимающие программы связи и IPC
- Балансировщики контейнеров табличного пространства

Дополнительную информацию об особенностях различных процессов DB2 смотрите в руководстве *Troubleshooting Guide*.

Использование системного монитора базы данных

Менеджер баз данных DB2 поддерживает информацию о своей работе, производительности и программах, которые его используют. Эти данные поддерживаются при запусках менеджера баз данных и содержат ценную информацию о производительности и устранении неисправностей. Можно, например, определить:

- Число прикладных программ, подключенных к базе данных, состояние каждой программы и оператор SQL, который она выполняет - если она выполняет оператор SQL.
- Информацию, показывающую, хорошо ли настроены менеджер баз данных и база данных, и помогающую настроить их лучше.
- Если вокруг некоторой базы данных возникали тупиковые ситуации - какие в них участвовали прикладные программы и какие блокировки.
- Список блокировок, удерживаемых прикладной программой или базой данных. Если программа не может продолжать выполнение из-за ожидания

блокировки, предоставляется дополнительная информация о блокировке, включая и сведения о том, какая программа эту блокировку удерживает.

Поскольку сбор некоторых из перечисленных видов данных связан с дополнительными расходами при работе DB2, **переключатели монитора** позволяют ограничить собираемую информацию. Чтобы явно задать переключатели монитора, используйте команду UPDATE MONITOR SWITCHES или интерфейс API sqlmon(). (Вам потребуются полномочия SYSADM, SYSCTRL или SYSMAINT.)

Есть два способа доступа к информации монитора баз данных:

- **Снимок**

  Снимки можно получать тремя способами. Можно ввести команду GET SNAPSHOT в командной строке, использовать графический интерфейс Центра управления в операционных системах OS/2 или Windows или же написать собственную прикладную программу с вызовом API sqlmonss().

  Центр управления, который можно вызвать из папки DB2 или командой `db2cc`, содержит инструмент монитора производительности, обеспечивающий выборку данных мониторинга через регулярные интервалы при помощи снимков. Этот графический интерфейс обеспечивает просмотр данных снимка в виде диаграмм или текста, подробно или в виде сводки. Кроме того, вы можете сами определять переменные производительности на основе элементов данных, возвращаемых монитором баз данных.

  Инструмент Монитор снимков Центра управления позволяет вам также определять условия исключительных ситуаций, задавая пороговые значения переменных производительности. При достижении порогового значения выполнится одно из следующих заранее определенных вами действий: сообщение при помощи окна или звукового сигнала и/или запуск сценария или прикладной программы.

  Если вы делаете снимки объекта базы данных (например, экземпляра или базы данных) или любого из его дочерних объектов из Центра управления, нельзя выполнить действие, которое изменит, заменит или удалит этот объект. (Кроме того, при мониторинге системы с распределенной базой данных нельзя обновлять вид объектов распределенной базы данных.) Например, нельзя следить за базой данных A, если вы хотите удалить ее экземпляр. Однако если вы наблюдаете только за этим экземпляром, вы можете изменить базу данных A.

  Чтобы прекратить весь мониторинг в отношении экземпляра (включая все его дочерние объекты), выберите **Остановить все мониторы** из всплывающего меню экземпляра. Мониторинг рекомендуется прекращать на уровне экземпляра, поскольку это гарантирует снятие всех блокировок, удерживаемых монитором производительности.

- **Использование монитора событий**

Монитор события захватывает информацию системного монитора при определенных событиях, например, завершения транзакции, завершения оператора или обнаружения тупиковой ситуации. Эта информация может помещаться в файлы или на именованные конвейеры.

Чтобы использовать монитор событий:

1. Создайте его определение при помощи Центра управления или оператора SQL CREATE EVENT MONITOR. Этот оператор сохраняет определение в системных каталогах базы данных.
2. Активируйте монитор событий при помощи Центра управления или оператора SQL:

   ```
   SET EVENT MONITOR имя_события STATE 1
   ```

   В случае записи в именованный конвейер программу, читающую именованный конвейер, надо запускать до активации монитора событий. Для этого можно написать свою прикладную программу или использовать **db2evmon**. После того, как монитор событий был активирован и начал запись событий в именованный конвейер, **db2evmon** будет считывать их по мере генерации и записывать в стандартное устройство вывода.
3. Прочитайте данные трассировки. При использовании файлового монитора событий создаваемую им двоичную трассировку можно просмотреть одним из следующих способов:
   - При помощи инструмента **db2evmon** отформатировать и перенаправить трассировку на стандартное устройство вывода.
   - Щелкнуть по значку **Анализатор событий** в Центре управления (в операционных системах Windows и OS/2) и использовать графический интерфейс, чтобы просмотреть трассировку, провести поиск ключевых слов и отсеять ненужную информацию.

     Примечание: Если наблюдаемая система базы данных работает на другом компьютере по отношению к Центру управления, для просмотра трассировки нужно скопировать файл монитора событий на тот компьютер, где находится Центр управления. Альтернативный метод - поместить этот файл в систему совместного использования файлов, доступную обеим компьютерам.

Информацию о системном мониторе баз данных и мониторе событий смотрите в руководстве *System Monitor Guide and Reference*.

Расширение памяти

На вашей машине может быть установлено больше реальной адресуемой памяти, чем максимальный объем виртуальной адресуемой памяти (так, обычно виртуальная адресуемая память на большинстве платформ составляет от 2 до 4 Гбайт). Дополнительную реальную адресуемую память, выходящую за пределы виртуальной адресуемой памяти, можно сконфигурировать как *кэш расширенной памяти*. Такой кэш расширенной памяти можно использоваться любым из определяемых пулов буферов и должен повысить производительность менеджера баз данных. Кэш расширенной памяти определяется в терминах сегментов памяти.

Следует знать, что решение использовать некоторую часть реальной адресуемой памяти как кэш расширенной памяти исключает ее использование на компьютере для других целей, например, в качестве JFS-кэша или собственного адресного пространства процесса. Передача дополнительной реальной адресуемой памяти под кэш расширенной памяти может увеличить системную подкачку страниц.

DB2 использует адресуемую память на вашем компьютере для пулов буферов (смотрите раздел “Управление пулом буферов базы данных” на стр. 261). Кэш расширенной памяти используется пулами буферов в качестве уровня вторичного кэширования (сами пулы буферов выполняют первый уровень кэширования). В идеальном случае пулы буферов могут удерживать наиболее часто запрашиваемые данные, а кэш расширенной памяти может удерживать данные, запрашиваемые несколько реже.

При выделении пула буферов Windows 2000 Address Windowing Extensions (AWE) с использованием переменной реестра DB2_AWE кэш расширенной памяти использоваться не может.

Следующие параметры конфигурации базы данных влияют на объем и размер памяти, доступной для расширения памяти:

- *num_estore_segs* задает число сегментов расширения памяти. По умолчанию этот параметр конфигурации имеет значение ноль, так что кэш расширенной памяти не существует. (Смотрите раздел “Число сегментов расширения памяти (num_estore_segs)” на стр. 418.)
- *estore_seg_sz* задает размер каждого сегмента расширения памяти. Этот размер ограничен платформой, на которой используется кэш расширенной памяти. (Смотрите раздел “Размер сегмента расширения памяти (estore_seg_sz)” на стр. 417.)

Поскольку кэш расширенной памяти является расширением пула буферов, он всегда должен быть ассоциирован с одним или несколькими конкретными пулами буферов. Таким образом, необходимо объявить, какие пулы буферов смогут воспользоваться кэшем после его создания. Операторы CREATE и

ALTER BUFFERPOOL имеют атрибуты NOT EXTENDED STORAGE и EXTENDED STORAGE, управляющие использованием кэша. По умолчанию ни IBMDEFAULTBP, ни вновь создаваемые пулы буферов не будут использовать расширенную память.

Примечание: Можно использовать пулы буферов, определенные с другим размером страниц. В определении некоторых или всех этих пулов буферов может быть задано использование расширенной памяти. Размер страницы, используемый для поддержки расширенной памяти - максимальный из определенных.

Менеджер баз данных не может напрямую работать с данными, находящимися в кэше расширенной памяти. Однако он может передавать данные из кэша расширенной памяти в пул буферов намного быстрее, чем из дисковой памяти.

Когда требуется строка данных со страницы в кэше расширенной памяти, вся страница считывается в соответствующий пул буферов.

Пул буферов и ассоциированный с ним кэш расширенной памяти, если он определен, выделяются в момент активации базы данных или первого соединения с ней.

Глава 9. Использование утилиты ограничения ресурсов

Утилита ограничения ресурсов служит для наблюдения за прикладными программами, выполняемыми на базе данных, и для изменения их поведения.

Эта утилита состоит из двух частей:
- Пользовательский интерфейс
- Демон

Запуская утилиту ограничения ресурсов, вы вводите команду запуска через пользовательский интерфейс, и он запускает демон ограничения ресурсов. По умолчанию демон запускается в каждом разделе базы данных, хотя можно также запустить один демон в заданном разделе, чтобы наблюдать за активностью этого раздела. Кроме того, демон может наблюдать за активностью в одноразделльной базе данных. Подробности смотрите в разделе “Запуск и остановка утилиты ограничения ресурсов” на стр. 300.

Каждый демон ограничения ресурсов собирает статистическую информацию о прикладных программах, выполняемых в базе данных. Затем он сверяет эту статистику с теми правилами, заданными в файле конфигурации утилиты ограничения ресурсов, которые относятся к этой конкретной базе данных. (Подробности смотрите в разделе “Создание файла конфигурации утилиты ограничения ресурсов” на стр. 303.) После этого утилита ограничения ресурсов выполняет действие, предусмотренное этими правилами. Например, может оказаться, что, согласно одному из правил, некоторая прикладная программа использует слишком много ресурсов. В такой ситуации утилита ограничения ресурсов может изменить приоритет прикладной программы или отключить ее от базы данных - согласно инструкциям, которые вы зададите в файле конфигурации утилиты ограничения ресурсов.

Если предусмотренное правилом действие - изменить приоритет прикладной программы, утилита ограничения ресурсов изменяет приоритет агентов в том разделе базы данных, где она отметила превышение расхода ресурсов. Если предусмотренное правилом действие - отключить прикладную программу, отключение состоится даже в том случае, когда утилита ограничения ресурсов, отметивший превышение расхода ресурса, запущена на узле-координаторе этой прикладной программы или в среде многораздельной базы данных.

Утилита ограничения ресурсов также ведет журнал всех своих действий. Запросив информацию из файлов журнала, можно просмотреть предпринятые утилитой действия. Подробности смотрите в разделах “Файлы журналов утилиты ограничения ресурсов” на стр. 312 и “Запрос информации из файлов журналов утилиты ограничения ресурсов” на стр. 313.

Запуск и остановка утилиты ограничения ресурсов

Команда db2gov утилиты ограничения ресурсов служит для запуска и остановки утилиты ограничения ресурсов (во всех разделах или в одном разделе базы данных). Для использования этой команды требуются полномочия SYSADM или SYSCTRL.

Синтаксис db2gov:

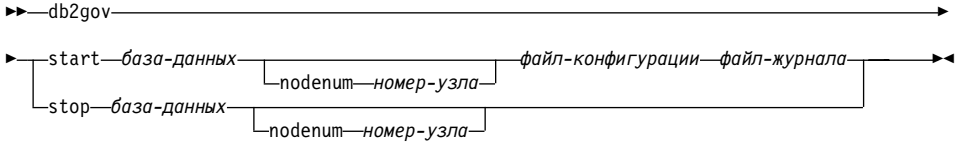


Рисунок 28. Синтаксис db2gov

Используются следующие параметры:

start **база-данных**
: Запускает демон для наблюдения за указанной базой данных. В качестве *базы-данных* можно ввести имя или алиас базы данных.

  Указываемое имя базы данных должно совпадать с тем именем, которое задано в файле конфигурации утилиты ограничения ресурсов. Утилита ограничения ресурсов сверяет эти два имени, чтобы убедиться, что вы используете правильный файл конфигурации. Если утилита запущена с одним алиасом, а в файле конфигурации утилиты ограничения ресурсов - другой алиас, выдается сообщение об ошибке, потому что утилита ограничения ресурсов не может определить, являются ли эти имена алиасами одной и той же базы данных.

  В среде многораздельной базы данных при запуске утилиты ограничения ресурсов во всех разделах при вводе команды сначала проверяется, что файл конфигурации не содержит ошибок. Затем утилита читает файл конфигурации узла и посылает команду в каждый раздел базы данных, чтобы во всех разделах базы данных запустить выполнить команду утилиты ограничения ресурсов с опцией start (что, в свою очередь, запускает демоны во всех разделах базы данных).

  Примечание: Поскольку утилита ограничения ресурсов ведет наблюдение на уровне базы данных, по одному демону выполняется на каждой базе данных, за которой ведется наблюдение. (В среде многораздельной базы данных по одному демону выполняется в каждом разделе базы данных.) Если утилита ограничения ресурсов запущена для нескольких баз данных, на сервере баз данных выполняется несколько демонов.

nodenum **номер-узла**

Задает раздел базы данных, в котором запускается демон ограничения ресурсов. Это тот же номер, что задан в файле конфигурации узла.

Когда утилита ограничения ресурсов запускается в одном разделе базы данных, создается демон для проверки файла конфигурации утилиты ограничения ресурсов. Демон ограничения ресурсов проверяет, что на этом разделе нет уже запущенного другого демона.

файл-конфигурации

Задает файл конфигурации, используемый при слежении за базой данных.

По умолчанию предполагается, что файл конфигурации находится в каталоге `sqllib`. Если заданного файла там нет, утилита предполагает, что заданное имя является полным именем файла.

файл-журнала

Задает основу имен файлов, где утилита ограничения ресурсов делает журнальные записи. Файл журнала сохраняется в подкаталоге `log` каталога `sqllib`. (В Windows NT подкаталог `log` находится в каталоге экземпляра.) Номер раздела базы данных, в котором запускается утилита ограничения ресурсов, автоматически добавляется к имени файла журнала (например, `mylog.0`, `mylog.1`, `mylog.2`).

stop **база-данных**

Останавливает демон ограничения ресурсов, наблюдающий за указанной базой данных.

В среде многораздельных баз данных утилита ограничения ресурсов читает файл конфигурации узла и затем посылает в каждый раздел базы данных команду утилиты ограничения ресурсов с параметром stop. В результате демоны прекращают работу во всех разделах базы данных.

nodenum **номер-узла**

Задает раздел базы данных, на котором останавливается демон ограничения ресурсов. Это тот же номер, что задан в файле конфигурации узла.

Когда утилита останавливает демон утилиты ограничения ресурсов в одном разделе базы данных, она передает информацию демону в этом разделе базы данных путем создания, перемещения и удаления файлов в подкаталоге `tmp` каталога `sqllib`. Не пытайтесь удалять или редактировать эти файлы.

Демон ограничения ресурсов

При запуске демона ограничения ресурсов (командой db2gov или активацией) он выполняет цикл. Первая его задача - проверить, изменился ли его файл конфигурации утилиты ограничения ресурсов и не читал ли его демон раньше. Если файл еще не прочитан или же изменился, демон читает правила в этом файле. Это позволяет изменять поведение демона ограничения ресурсов во время работы.

Затем демон ограничения ресурсов заказывает снимок, чтобы получить статистику по каждой программе и агенту, работающему в базе данных.

Примечание: На некоторых платформах статистика процессора недоступна из Монитора DB2. В этом случае будут недоступны правило account и предел CPU.

Затем утилита ограничения ресурсов сверяет статистику каждой программы с правилами из файла конфигурации утилиты ограничения ресурсов. Если некоторое правило относится к некоторой прикладной программе, утилита ограничения ресурсов может, в зависимости от заданного правилом действия: отключить программу; изменить приоритет программы, что косвенно изменит все приоритеты агентов и подагентов, работающих для этой программы на данном разделе базы данных; наконец, изменить план программы, что косвенно изменит приоритеты агентов, работающих для этой программы. Утилита ограничения ресурсов регистрирует все свои действия в файле журнала.

Примечание: Утилита ограничения ресурсов не сможет играть роль альтернативного средства для регулировки приоритетов агентов, если параметр *agentpri* конфигурации менеджера баз данных отличается от системного значения по умолчанию. (Это примечание не относится к платформам Windows NT.)

Закончив проверку всех прикладных программ, утилита ограничения ресурсов выключается на время, заданное в файле конфигурации. По истечении заданного времени утилита ограничения ресурсов включается и приступает к очередному выполнению цикла.

Получив ошибку или сигнал остановки, утилита ограничения ресурсов перед окончанием работы проводит уборку. Процесс уборки сбрасывает все приоритеты агентов прикладных программ (с помощью списка прикладных программ, имеющих заданные приоритеты). Затем он сбрасывает приоритеты всех агентов, уже не работающих для программ. Этим он предотвращает сохранение новых приоритетов у работающих агентов. Если произошла ошибка, в файл db2diag.log записывается сообщение о ненормальном завершении работы утилиты ограничения ресурсов.

Примечание: Демон ограничения ресурсов не является прикладной программой базы данных и поэтому не подключается к базе данных (хотя и имеет подключение к экземпляру.) Демон утилиты ограничения ресурсов может заметить, что завершился менеджер баз данных, потому что может заказывать снимки.

Создание файла конфигурации утилиты ограничения ресурсов

Запуская утилиты ограничения ресурсов, вы задаете имя файла конфигурации, который содержит правила управления прикладными программами, выполняемыми в базе данных. Утилита ограничения ресурсов действует в соответствии с этими правилами.

Если ваши требования к регулированию работы базы данных изменятся, вы можете отредактировать файл конфигурации, не останавливая утилиту ограничения ресурсов. Все демоны ограничения ресурсов обнаружат, что файл конфигурации изменился, и перечитают его.

Файл конфигурации нужно создавать в каталоге, смонтированном на всех разделах базы данных, поскольку на всех разделах демоны ограничения ресурсов должны иметь возможность читать один и тот же файл конфигурации.

Файл конфигурации состоит из правил и комментариев. Большинство записей могут вводиться в верхнем регистре, нижнем регистре и смешанном регистре. Исключение - `applname`, которое регистрозависимо.

Комментарии заключаются в фигурные скобки { }. В правилах указывается:

- База данных, к которой относятся правила.
- На какое время утилита ограничения ресурсов отключается между периодами активной проверки прикладных программ.
- Правила управления прикладными программами. Эти правила состоят из компонентов, называемых условиями правил.

Каждое правило в файле должно оканчиваться точкой с запятой (;).

Следующие правила задают базу данных, за которой ведется наблюдение, и интервалы, через которые демон периодически включается (они описаны в разделе “Демон ограничения ресурсов” на стр. 302). Каждое из этих правил задается в файле только один раз.

dbname
Имя или алиас базы данных, за которой должно вестись наблюдение.

account *nnn*
В отчет записывается статистика использования процессора по каждому соединению через заданное число минут.

Примечание: Эта опция недоступна в среде Windows NT.

Если сеанс связи окажется короче интервала отчета, никаких записей в отчет сделано не будет. В запись попадает статистика процессора, отражающая его использование с момента предыдущей записи для соединения. При остановке или перезапуске утилиты ограничения ресурсов информация об использовании процессора может попасть сразу в две записи; это можно обнаружить по ID программ, указанным в записях. Дополнительную информацию о файлах журнала утилиты ограничения ресурсов смотрите в разделе “Файлы журналов утилиты ограничения ресурсов” на стр. 312.

interval

Интервал (в секундах), через который включается демон. Если интервал не задан, используется 120 секунд.

Для построения правила комбинируются следующие условия (имейте в виду, что точка с запятой оканчивает полное правило, а не отдельное условие). Условия задают время действия правила, предел использования ресурса и, как необязательный параметр - конкретных пользователей или конкретные программы и любые действия утилиты ограничения ресурсов в случае превышения заданного предела. Условия нельзя задавать несколько раз в одном правиле, но можно задавать во многих правилах. Условия нужно задавать в том порядке, в каком они здесь показаны. В следующем описании [ ] указывают необязательное условие.

[desc] Задает текстовое описание правила. Описание можно заключить в одиночные или двойные кавычки.

[time] Задает период времени, в течение которого проверяется соблюдение правила.

Период времени нужно задать в формате `time чч:мм чч:мм`, например, `time 8:00 18:00`. Если это условие не задано, правило будет действовать 24 часа в сутки.

[authid]

Задает один или несколько ID авторизации (authid), под которыми выполняются прикладные программы. Несколько authids следует разделять запятыми (,), например, `authid gene, michael, james`. Если этого условия не будет в правиле, правило будет применяться ко всем authid.

[applname]

Задает имя выполняемого (объектного) файла, который устанавливает соединение с базой данных.

Несколько имен прикладных программ нужно разделять запятыми (,), например, `applname db2bp, batch, geneprog`. Если этого условия не будет в правиле, правило будет применяться ко всем именам прикладных программ.

Примечания:

1. Имена прикладных программ регистрозависимы.
2. Менеджер баз данных усекает все имена прикладных программ до 20 символов. Нужно убедиться, что в идентификаторе прикладной программы, для которой вы запускаете утилиту ограничения ресурсов, первые 20 символов уникальны; в противном случае утилита может быть применена не только к нужной программе.

   Имена прикладных программ, задаваемые в файле конфигурации утилиты ограничения ресурсов, усекаются до 20 символов, чтобы привести их в соответствие с внутренним представлением.

setlimit

Задает один или несколько пределов, проверяемых утилитой. Можно использовать только -1 и положительные числа (например, `cpu -1 locks 1000 rowssel 10000`). Должен быть задан по меньшей мере один предел (cpu, locks, rowsread, uowtime); незаданные пределы не будут ограничиваться этим правилом. Утилита ограничения ресурсов может проверять соблюдение следующих пределов:

cpu *nnn*
: Задает число секунд процессора, которое может израсходовать прикладная программа. Если задать `-1`, утилита ограничения ресурсов не будет ограничивать использование процессора программой.

  Примечание: Эта опция недоступна в среде Windows NT.

locks *nnn*
: Задает число блокировок, которое может удерживать прикладная программа. Если задать `-1`, утилита ограничения ресурсов не будет ограничивать число блокировок, удерживаемых программой.

rowssel *nnn*
: Задает число строк, возвращаемых программе. Это значение будет отлично от нуля только на узле-координаторе. Если задать `-1`, утилита ограничения ресурсов не будет ограничивать число выбираемых строк.

uowtime *nnn*
: Задает число секунд, которое может пройти с момента, когда единица работы (UOW) активируется впервые. Если задать `-1`, время обработки не ограничивается.

Примечание: Если вы с помощью интерфейса API `sqlmon` (Монитор систем баз данных Switch) деактивируете переключатель единицы работы, это повлияет на способность утилиты ограничения ресурсов регулировать прикладные программы на основе затраченного единицей работы времени. Для сбора информации о системе утилита ограничения ресурсов использует монитор. Если вы выключите переключатели в файле конфигурации менеджера баз данных, он отключится во всем экземпляре, и утилита ограничения ресурсов не получит этой информации. Дополнительную информацию смотрите в разделе “Параметры монитора баз данных” на стр. 498.

idle *nnn*

Задает допустимое число секунд простоя соединения до выполнения заданного действия. Если задать `-1`, время простоя соединения не ограничивается.

rowsread *nnn*

Задает число строк, которое может выбрать прикладная программа. Если задать `-1`, число выбираемых прикладной программой строк не будет ограничиваться.

Примечание: Этот предел отличается от rowssel. Различие в том, что rowsread - это число строк, которые надо будет прочитать, чтобы вернуть результат. В число прочитанных строк входят чтение таблицы каталога ядром; число может уменьшиться при использовании индексов.

[действие]

Задает действие при превышении одного или нескольких заданных пределов. Можно задать следующие действия:

Примечание: Если условие action не задано, при превышении предела утилита ограничения ресурсов уменьшает приоритет агентов, работающих для прикладной программы, на 10.

priority *nnn*

Задает изменение приоритета агентов, работающих для прикладной программы. Допустимые значения - от –20 до +20.

Чтобы этот параметр действовал:

- На платформах на основе UNIX параметр *agentpri* менеджера баз данных должен иметь значение по умолчанию; в противном случае он отменит условие приоритета.
- На платформах OS/2 и Windows NT параметр *agentpri* менеджера баз данных и действие *priority* могут использоваться вместе.

force Задает отключение агента, обслуживающего прикладную программу. (Для этого используется команда FORCE APPLICATION, которая завершает агент-координатор.)

schedule [class]

Планирование улучшает приоритеты агентов, работающих для прикладной программы, чтобы минимизировать средние времена ответа, соблюдая справедливость по отношению ко всем программам.

Утилита ограничения ресурсов навязывает собственный план, задавая приоритеты для агентов, работающих для прикладных программ, используя оценки стоимости запросов от внутреннего компилятора запросов DB2. Если задана опция class, все прикладные программы, выбранные правилом, подвергаются планированию только в отношении друг друга. Если эта опция не задана, утилита ограничения ресурсов использует один или несколько классов, причем планирование осуществляется в пределах каждого класса.

Назначение приоритетов в пределах каждого класса учитывает:

- Число блокировок, удерживаемых прикладной программой в данном классе. (Высокий приоритет получает прикладная программа, задерживающая своими блокировками много других прикладных программ.)
- Возраст прикладной программы. (Высокий приоритет получает прикладная программа, долгое время находящаяся в системе.)
- Оценка оставшегося времени выполнения прикладной программы. (Высокий приоритет получает прикладная программа, близкая к завершению.)

Прикладные программы, не охваченные планированием, выполняются с наивысшим приоритетом.

Примечание: Если вы с помощью интерфейса API `sqlmon` (Монитор систем баз данных Switch) деактивируете переключатель оператора, это повлияет на способность утилиты ограничения ресурсов регулировать прикладные программы на

основе затраченного оператором времени. Для сбора информации о системе утилита ограничения ресурсов использует монитор. Если вы выключите переключатели в файле конфигурации менеджера баз данных, он отключится во всем экземпляре, и утилита ограничения ресурсов не получит этой информации.

Действие планирования может:

- Обеспечить выделение времени программам во всех группах без равного разделения времени между всеми программами.

  Например, если 12 прикладных программ (три коротких, пять средних и шесть длинных) выполняются одновременно, у всех может быть плохое время ответа, поскольку они делят между собой процессор. Администратор базы данных может задать две группы - программ средней длины и программ большой длины. При помощи приоритетов утилита ограничения ресурсов разрешает выполняться коротким прикладным программам, следя, чтобы одновременно выполнялось не более трех средних и трех длинных прикладных программ. Для этого в файле конфигурации утилиты ограничения ресурсов должно быть одно правило для прикладных программ средней длины и еще одно - для длинных. В следующем примере показан фрагмент файла конфигурации утилиты ограничения ресурсов, иллюстрирующий это:

  ```
  desc "Сгруппировать средние программы в один класс планирования"
  applname medq1, medq2, medq3, medq4, medq5
  setlimit cpu -1
  action schedule class;

  desc "Сгруппировать длинные программы в один класс планирования"
  applname longq1, longq2, longq3, longq4, longq5, longq6
  setlimit cpu -1
  action schedule class;
  ```

- Обеспечить равные приоритеты всем группам пользователей (например, всем отделам организации).

  Если одна из групп запускает большое число прикладных программ, администратор может устроить так, что другие группы все равно смогут выполнять свои программы с разумным временем ответа. Например, для трех отделов (финансового, инвентаризации и планового) всех пользователей из финансового отдела можно собрать в одну группу, всех пользователей из отдела инвентаризации - в другую, а из планового отдела - в третью. Мощность процессора будет приблизительно поровну поделена между

тремя отделами. В следующем примере показан фрагмент файла конфигурации утилиты ограничения ресурсов, иллюстрирующий это:

```
desc "Сгруппировать пользователей финансового отдела"
authid tom, dick, harry, mo, larry, curly
setlimit cpu -1
action schedule class;

desc "Сгруппировать пользователей отдела инвентаризации"
authid pat, chris, jack, jill
setlimit cpu -1
action schedule class;

desc "Сгруппировать пользователей планового отдела"
authid tara, dianne, henrietta, maureen, linda, candy
setlimit cpu -1
action schedule class;
```

- Разрешить утилите ограничения ресурсов заниматься планированием прикладных программ.

  Если в условии action нет опции класса, утилита ограничения ресурсов создает собственные классы с учетом числа прикладных программ, подпадающих под действие планирования, и помещает прикладные программы в разные классы с учетом оцененной компилятором стоимости запроса DB2, выполняемого сейчас прикладной программой. Администратор может выбрать планирование для всех программ, не задавая спецификаций для выбора программ. Это значит, что условия *applname* и *authid* не задаются, а условие *setlimit* не накладывает ограничений.

Примечание: Если условие action не задано, при превышении предела утилита ограничения ресурсов уменьшает приоритет агентов, работающих для прикладной программы.

Если к программе применимы несколько правил, применяются все правила. В зависимости от правила и устанавливаемых пределов, первым применяется действие, связанное с первым встреченным пределом. Исключение - случай, когда в правиле для условия задано -1. В этой ситуации значение, заданное для условия в следующем правиле, может отменить только значение, заданное ранее для *того же* условия: другие условия в предыдущем правиле остаются рабочими. Например, одно правило указывает, что приоритет программы должен быть уменьшен, если время обработки превысит 1 час или если она выберет более 100000 строк (это правило `rowssel 100000 uowtime 3600`). Следующее правило указывает, что та же программа может иметь неограниченное время обработки (это правило `uowtime -1`). Если в этой ситуации программа будет выполняться более 1 часа, ее приоритет не

уменьшится (это значит, что `uowtime -1` отменит `uowtime 3600`), но если она выберет более 100000 строк, ее приоритет будет снижен (поскольку `rowssel 100000` продолжает действовать).

На рис. 29 на стр. 311 приведен пример файла конфигурации.

```
{ Активировать раз в секунду, имя базы данных ibmsampl
     генерировать отчет каждые 30 минут.  }
interval 1; dbname ibmsampl; account 30;

desc "ограничения процессора действуют 24 в сутки для всех"
setlimit cpu 600 rowssel 1000000 rowsread 5000000;

desc "Не разрешать единицам работы выполняться дольше часа"
setlimit uowtime 3600 action force;

desc 'Замедлить выполнение некоторых прикладных программ'
applname jointA, jointB, jointC, quryA
setlimit cpu 3 locks 1000 rowssel 500 rowsread 5000;

desc "утилита ограничения ресурсов соберет в один класс по приоритету следующие 6 длинных
программ"
applname longq1, longq2, longq3, longq4, longq5, longq6
setlimit cpu -1
action schedule class;

desc "Планировать все программы, запускаемые плановым отделом"
authid planid1, planid2, planid3, planid4, planid5
setlimit cpu -1
action schedule;

desc "Все программы, интенсивно использующие процессор, планировать как один класс управления
использованием"
setlimit cpu 3600
action schedule class;

desc "Замедлить обработку командной строки db2 для пользователя novice"
authid novice
applname db2bp.exe
setlimit cpu 5 locks 100 rowssel 250;

desc "В дневные часы никому не разрешать выполнять запросы дольше 10 секунд"
time 8:30 17:00 setlimit cpu 10 action force;

desc "Разрешить пользователям, занимающимся настройкой производительности, выполнять
     некоторые программы в обеденный перерыв"
time 12:00 13:00 authid ming, geoffrey, john, bill
applname tpcc1, tpcc2, tpcA, tpvG setlimit cpu 600 rowssel 120000 action force;

desc "Не применять ограничения к некоторым пользователям -- администратору базы данных
  и некоторым другим. Поскольку это последняя спецификация в
  файле, она отменяет все предшествующие."
authid gene, hershel, janet setlimit cpu -1 locks -1 rowssel -1 uowtime -1;

desc "Повысить приоритет важной программы, чтобы она всегда
      завершалась быстро"
applname V1app setlimit cpu 1 locks 1 rowssel 1 action priority -20;
```

Рисунок 29. Пример файла конфигурации утилиты ограничения ресурсов

Файлы журналов утилиты ограничения ресурсов

Когда демон ограничения ресурсов отключает прикладную программу, читает файл конфигурации утилиты ограничения ресурсов, изменяет приоритет программы, обнаруживает ошибку или предупреждение, запускается и завершается, он делает запись в файле журнала. У каждого демона утилиты ограничения ресурсов свой файл журнала. Это предотвращает блокировки файла, в который много демонов утилиты ограничения ресурсов одновременно пытались бы сделать записи. При помощи утилиты `db2govlg` можно сливать файлы журнала и запрашивать информацию из них. Эта утилита описана в разделе “Запрос информации из файлов журналов утилиты ограничения ресурсов” на стр. 313.

Файлы журналов сохраняются в подкаталоге `log` каталога `sqllib`. (В Windows NT подкаталог `log` находится в каталоге экземпляра.) Основа имен файлов журнала задается в команде `db2gov`. Убедитесь, что имя файла журнала содержит имя базы данных, поскольку на каждом узле каждой базы данных, для которой применяется утилита ограничения ресурсов, будет свой файл журнала. В среде многораздельной базы данных номер узла раздела, в котором выполняется утилита ограничения ресурсов, автоматически добавляется к имени файла журнала, обеспечивая уникальность имен файлов всех утилит ограничения ресурсов.

Формат записей в файле журнала:

```
Date Time NodeNum RecType Message
```

Поле *Date* и *Time* имеет формат гггг-мм-дд чч.мм.сс, что позволяет при слиянии файлов журналов всех разделов базы данных использовать сортировку по этому полю.

Поле *NodeNum* содержит номер раздела базы данных, в котором запускается утилита ограничения ресурсов.

Значения в поле *RecType* зависят от типа сделанной записи журнала. Возможны следующие типы записи:

- `START` - утилита ограничения ресурсов запущена
- `FORCE` - прикладная программа отключена
- `PRIORITY` - изменен приоритет прикладной программы
- `ERROR` - ошибка
- `WARNING` - предупреждение
- `READCFG` - утилита ограничения ресурсов прочитала файл конфигурации
- `STOP` - утилита ограничения ресурсов остановлена
- `ACCOUNT` - статистический отчет о прикладной программе.

  В этом отчете есть следующие поля:

- authid
- appl_id
- written_usr_cpu
- written_sys_cpu
- appl_con_time

• SCHEDULE - произведено изменение приоритетов агентов.

Благодаря записи стандартных значений из файлов журнала можно запрашивать сведения о разных типах действий. Остальная, нестандартная информация находится в поле *Message*; содержание сообщения меняется в зависимости от значения в поле *RecType*. Например, запись FORCE или NICE сопровождается информацией о прикладной программе в поле *Message*, а запись ERROR - сообщением об ошибке.

Вот пример файла журнала:

```
1995-12-11 14.54.52    0 START      Database = TQTEST
1995-12-11 14.54.52    0 READCFG    Config = /u/db2instance/sqllib/tqtest.cfg
1995-12-11 14.54.53    0 ERROR      SQLMON Error: SQLCode = -1032
1995-12-11 14.54.54    0 ERROR      SQLMONSZ Error: SQLCode = -1032
```

Запрос информации из файлов журналов утилиты ограничения ресурсов

Каждый демон ограничения ресурсов ведет собственный файл журнала. Для запроса информации из журнала можно использовать утилиту db2govlg. Можно получить список файлов журналов в заданном разделе или во всех разделах базы данных, отсортированный по дате и времени. Можно также делать запросы с учетом поля журнала *RecType*. Синтаксис db2govlg:

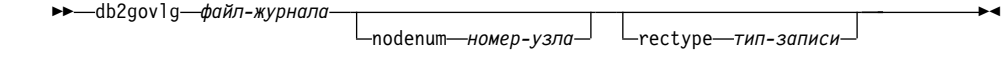


Рисунок 30. Синтаксис db2govlg

Используются следующие параметры:

файл-журнала
Основа имен файлов журнала, из которого вы запрашиваете информацию.

nodenum **номер-узла**
Номер узла раздела базы данных, на котором запускается утилита ограничения ресурсов.

rectype **тип-записи**
Тип запрашиваемой записи. Возможные типы записи:
- START
- READCFG

- STOP
- FORCE
- NICE
- ERROR
- WARNING
- ACCOUNT

На использование этой утилиты нет ограничений авторизации. Благодаря этому все пользователи могут выяснять, повлияла ли утилита ограничения ресурсов на их программу. Если вы хотите ограничить доступ к этой утилите, вы можете изменить права группы для файла `db2govlg`.

Запуск утилиты ограничения ресурсов и производительность менеджера баз данных

Утилита ограничения ресурсов может повлиять на производительность менеджера баз данных, поскольку она заказывает снимки менеджера баз данных. Если утилита ограничения ресурсов потребляет слишком много ресурсов процессора, можно увеличить интервал ее активации и тем сократить использование процессора.

Глава 10. Увеличение числа процессоров в конфигурации

Возможна ситуация, когда характеристики конфигурации не удовлетворяют текущим и планируемым потребностям. В таком случае придется принимать меры к повышению емкости конфигурации, или ее производительности, или и того, и другого. Например, добавление к конфигурации контейнеров повышает емкость, то есть возможный объем хранимых данных; оно может также повысить производительность утилит (таких как загрузка данных). Другие пути повышения емкости и производительности: добавление памяти и добавление процессоров в симметрической многопроцессорной среде или в среде многораздельных баз данных.

Главная тема этой главы - повышение производительности путем увеличения числа процессоров в конфигурации.

Увеличение конфигурации, которое описывается далее, имеет смысл, если:

- У вас была одноразделная конфигурация с одним процессором, который использовался на полную мощность. Вы решили изменить эту конфигурацию, причем:
  - Определили, что лучше всего в новой среде использовать симметрическую многопроцессорную конфигурацию. Возможно, вы приняли это решение, надеясь использовать большую вычислительную мощность нескольких процессоров. Все процессоры совместно используют системные ресурсы оперативной памяти и дисковой памяти. Поскольку эти процессоры работают в одной системе, не приходится решать такие вопросы, как линии связи между системами, дополнительные сотрудники-администраторы для работы со всеми новыми системами и координация выполнения заданий на этих системах. DB2 Universal Database поддерживает такую среду.
  - Определили, что в новой среде лучше всего использовать конфигурацию многораздельных баз данных. Возможно, вы приняли это решение, надеясь использовать большую вычислительную мощность нескольких процессоров, физически отделенных от первого. У каждого процессора свои системные ресурсы оперативной и дисковой памяти, которыми не пользуются совместно другие процессоры. Хотя это сопряжено с решением перечисленных выше вопросов (линии связи, сотрудники, координация работ), у этого варианта есть свои преимущества, такие как возможности равномерно распределить данные и пользовательские запросы по нескольким системам. DB2 Universal Database поддерживает такую среду.
- Текущая конфигурация - SMP (симметрическая многопроцессорная), и вы планируете добавить еще один или несколько процессоров. В этом случае вы

уже знакомы с особенностями среды этого типа. Добавляя еще один или несколько процессоров, вы просто увеличиваете вычислительную мощность среды без необходимости решать новые вопросы. DB2 Universal Database поддерживает такую среду.

- Текущая конфигурация - многораздельная база данных, и вы планируете добавить еще один или несколько разделов базы данных. В этом случае вы уже знакомы с особенностями среды этого типа. Добавляя еще один или несколько разделов базы данных, вы просто увеличиваете вычислительную мощность среды без необходимости решать новые вопросы, кроме обеспечения перехода к большему числу разделов. DB2 Universal Database поддерживает такую среду.

  Существует вариант конфигурации многораздельной базы данных, в котором разделами базы данных служат компьютеры SMP. DB2 Universal Database поддерживает такую среду.

Увеличивая систему и изменяя среду, нужно знать о возможном влиянии этих изменений на процедуры базы данных, такие как загрузка данных или резервное копирование и восстановление базы данных.

При добавлении нового раздела базы данных нельзя отбрасывать и создавать базы данных, пользующиеся новым разделом, пока процедура не будет завершена и пока новый сервер не будет успешно интегрирован в систему.

Добавление процессоров в компьютер

Если существующие процессоры полностью загружены большую часть времени, есть смысл установить на компьютер еще один или несколько процессоров. Чтобы менеджер баз данных DB2 смог пользоваться новыми процессорами, нужно проверить и, возможно, изменить несколько параметров конфигурации. (Некоторые операционные системы, например, Solaris, могут подключать и отключать процессоры динамически.) В число параметров, которые используются для определения числа процессоров и которые, возможно, понадобится изменить, входят:

- “Степень параллелизма по умолчанию (dft_degree)” на стр. 469
- “Максимальная степень параллелизма запросов (max_querydegree)” на стр. 493
- “Разрешение внутриразделького параллелизма (intra_parallel)” на стр. 494

Нужно также рассмотреть и, возможно, изменить параметры, связанные с прикладными программами. Дополнительную информацию смотрите в разделе “Параллельная обработка прикладных программ” на стр. 92.

При работе в среде, использующей протокол связи TCP/IP, нужно рассмотреть значение переменной регистра DB2TCPCONNMGRS. Дополнительную

информацию об этой переменной смотрите в разделе “Приложение A. Переменные реестра и среды DB2” на стр. 521.

Добавление разделов к системе многораздельной базы данных

Добавить разделы к системе многораздельной базы данных можно либо во время ее работы, либо во время ее остановки. Ниже описано, как это сделать. Поскольку добавление нового сервера может занять много времени, может оказаться желательным сделать это, не прерывая работы менеджера баз данных. Эта процедура описана в разделе “Добавление разделов базы данных к работающей системе” на стр. 318.

Для добавления раздела базы данных к системе служит команда ADD NODE. Команду можно ввести:

- Как опцию в `db2start`
- При помощи:
  - Команды ADD NODE процессора командной строки
  - `sqleaddn`
  - `sqlepstart`.

Метод, используемый для вызова команды, зависит от того, остановлена система (тогда нужно использовать `db2start`) или запущена (тогда нужно использовать любой из остальных вариантов).

Когда новый раздел базы данных добавляется к системе при помощи команды ADD NODE, в новом разделе создаются все существующие базы данных экземпляра. Можно также задать, какие контейнеры для временных табличных пространств будут использоваться для создаваемых баз данных. Эти контейнеры можно:

- Создать, как на узле каталога для каждой базы данных. (Это вариант по умолчанию.)
- Создать, как на другом разделе базы данных.
- Вообще не создавать. Прежде чем базу данных можно будет использовать, нужно будет добавить контейнеры временных табличных пространств к каждой базе данных при помощи оператора ALTER TABLESPACE.

Базу данных в новом разделе нельзя будет использовать для хранения данных, пока одна или несколько групп узлов не будет изменена, чтобы включить новый раздел базы данных. Дополнительную информацию о том, как изменить группу узлов, смотрите в разделе “Добавление и отбрасывание разделов баз данных” на стр. 329.

Примечание: Если в системе не определено ни одной базы данных, и вы запустили DB2 Enterprise - Extended Edition в системе на основе

UNIX, отредактируйте файл db2nodes.cfg, добавив новое определение раздела базы данных; не используйте следующих процедур, поскольку они применимы, только когда база данных существует. Дополнительную информацию об изменении файла конфигурации узла смотрите в разделе “Изменение группы узлов” книги *Administration Guide: Planning*.

Особенности Windows NT: Если вы используете DB2 Enterprise - Extended Edition в Windows NT, и в экземпляре нет ни одной базы данных, для увеличения размера системы баз данных нужно использовать команду DB2NCRT. Дополнительную информацию об этой команде смотрите в руководстве *Command Reference*. Если, с другой стороны, у вас уже есть базы данных, следует использовать команду DB2START ADDNODE, поскольку она обеспечивает создание разделов баз данных для каждой существующей базы данных. Информацию о команде DB2START и параметрах, которые нужно использовать в Windows NT, смотрите в руководстве *Command Reference*. В Windows NT не следует вручную редактировать файл конфигурации узла (db2nodes.cfg), чтобы не создать в этом файле противоречий.

Добавление разделов базы данных к работающей системе

Возможно добавление новых разделов к системе многораздельной базы данных, когда она работает и к базам данных подключены прикладные программы. Однако вновь добавленный сервер станет доступен для всех баз данных только после того, как будет закрыт и перезапущен менеджер баз данных.

Чтобы добавить раздел базы данных к системе с несколькими серверами:

1. Если раздел базы данных должен быть создан на сервере, который уже существует в системе, переходите к следующему шагу. В противном случае:
   - На платформах UNIX:
     a. Установите новый сервер. Надо обеспечить доступ к исполняемым программам (при помощи монтирования совместно используемой файловой системы или локального копирования), синхронизацию файлов операционной системы с файлами в существующих процессорах, доступность каталога sqllib в качестве совместно используемой файловой системы и задание нужных значений соответствующим параметрам операционной системы (таким как максимальное число процессов).
     b. Зарегистрируйте имя хоста с помощью сервера имен или в файле hosts в каталоге etc во всех разделах базы данных.
   - На платформах Windows NT:
     a. Установите новый сервер.
     b. Запустите на новом сервере команду ADD NODE. Эта команда выполнит локальное создание раздела базы данных для каждой базы данных, которая уже существует в системе. Параметрам базы данных

для нового раздела базы данных задаются значения по умолчанию; каждый раздел базы данных остается пустым, пока в него не поместят данные. Вы должны изменить значения параметров конфигурации базы данных, чтобы они соответствовали значениям для других разделов базы данных.

c. Перейдите к шагу три (3).

2. На каждом разделе базы данных выполните команду DB2START, задав новые значения параметров NODENUM, ADDNODE, HOSTNAME, PORT и NETNAME для этого раздела. На платформе Windows NT нужно также задать параметры COMPUTER, USER и PASSWORD. Дополнительную информацию о команде DB2START смотрите в руководстве *Command Reference*.

   При желании можно также задать источник для всех определений контейнеров временных табличных пространств, которые нужно создавать вместе с базами данных. Если не задать никакой информации о табличных пространствах, определения контейнеров временных табличных пространств будут взяты из узла каталога для каждой базы данных.

   После выполнения команды новый сервер останавливается. Информация о новом сервере изменит файл конфигурации узла только после того, как будет выполнена DB2STOP. Такой порядок гарантирует, что команда ADD NODE (вызываемая при задании параметра ADDNODE) будет запущена на корректном разделе базы данных. По завершении работы утилиты новый сервер останавливается.

3. Остановите менеджер баз данных, выполнив команду DB2STOP.

   Когда вы останавливаете все разделы базы данных в системе, файл конфигурации узла обновляется с учетом нового раздела базы данных.

4. Запустите менеджер баз данных, выполнив команду DB2START.

   Теперь вновь добавленный раздел базы данных запускается вместе с остальной системой.

   Когда запущены все разделы базы данных системы, можно выполнять общесистемные работы, такие как создание и отбрасывание базы данных.

   Примечание: Возможно, придется дважды ввести команду DB2START, чтобы все серверы раздела базы данных прочитали новый файл `db2nodes.cfg`.

5. При желании сделайте резервные копии всех баз данных на новом разделе базы данных.

6. При желании перераспределите данные, используя новый раздел базы данных. Подробности приводятся в разделе “Глава 11. Перераспределение данных между разделами баз данных” на стр. 327.

Добавление разделов базы данных к остановленной системе

Новые разделы можно добавить к системе многораздельной базы данных, когда она остановлена. Вновь добавленный раздел базы данных становится доступен для всех баз данных, когда снова запускается менеджер баз данных. У вас есть два варианта. Можно либо обновить файл конфигурации узла при помощи менеджера баз данных, либо сделать это вручную. Предварительные шаги для обеих процедур одинаковы.

Примечание: Не следует вручную обновлять файл конфигурации узла при работе в Windows NT. Нужно провести обновление этого файла при помощи менеджера баз данных (как описано ниже).

Чтобы добавить новый раздел базы данных к многосерверной системе:

1. Введите команду DB2STOP, чтобы остановить разделы базы данных.
2. Если сервер должен быть создан на процессоре, который уже существует в системе, переходите к следующему шагу. В противном случае:
   a. На платформах UNIX:
      1) Установите новый сервер. Надо обеспечить доступ к исполняемым программам (при помощи монтирования совместно используемой файловой системы или локального копирования), синхронизацию файлов операционной системы с файлами в существующих процессорах, доступность каталога `sqllib` в качестве совместно используемой файловой системы и задание нужных значений соответствующим параметрам операционной системы (таким как максимальное число процессов).
      2) Зарегистрируйте имя хоста с помощью сервера имен или в файле `hosts` в каталоге `etc` во всех разделах базы данных.
   b. На платформах Windows NT:
      1) Установите новый сервер.
      2) Запустите на новом сервере команду ADD NODE. Эта команда выполнит локальное создание раздела базы данных для каждой базы данных, которая уже существует в системе. Параметрам базы данных для нового раздела базы данных задаются значения по умолчанию; каждый раздел базы данных остается пустым, пока в него не поместят данные. Вы должны изменить значения параметров конфигурации базы данных, чтобы они соответствовали значениям для других разделов базы данных.
      3) Введите команду DB2START, чтобы запустить систему базы данных. Заметьте, что файл конфигурации узла (`db2nodes.cfg`) уже был обновлен с учетом нового сервера во время установки нового сервера.

4) При желании перераспределите данные, используя новый сервер. Дополнительные подробности о том, как это сделать, смотрите в разделе “Глава 11. Перераспределение данных между разделами баз данных” на стр. 327.

c. Если вы хотите поручить обновление файла `db2nodes.cfg` менеджеру баз данных, выполните далее инструкции из раздела “Изменение файла конфигурации узла при помощи менеджера баз данных”.

Примечание: В Windows NT не следует вручную редактировать файл `db2nodes.cfg`, чтобы не создать в этом файле противоречий. Обновление этого файла следует выполнять при помощи менеджера баз данных.

Если вы хотите обновить файл `db2nodes.cfg` сами, выполните далее инструкции из раздела “Внесение изменений в файл конфигурации узла вручную” на стр. 322.

Изменение файла конфигурации узла при помощи менеджера баз данных

После добавления одного или нескольких новых разделов к многораздельной системе баз данных, чтобы завершить обеспечение доступа к новому разделу, надо внести изменения в файл `db2nodes.cfg`. Если вы приняли решение внести изменения в файл конфигурации узла при помощи менеджера баз данных, посмотрите ниже подробности требуемых действий.

Примечание: Если вы приняли решение внести изменения в файл конфигурации узла вручную, пропустите оставшуюся часть этого раздела.

Дальнейшая процедура:

1. В новом разделе базы данных введите команду DB2START, задав параметры NODENUM, ADDNODE, HOSTNAME, PORT и NETNAME. На платформе Windows NT нужно также задать параметры COMPUTER, USER и PASSWORD. Дополнительную информацию о команде DB2START смотрите в руководстве *Command Reference*. Заданные вами значения параметров будут использованы для внесения изменений в файл конфигурации узла.

   После выполнения команды новый сервер останавливается. Информация о новом сервере изменит файл конфигурации узла только после того, как будет выполнена DB2STOP. Такой порядок гарантирует, что команда ADD NODE (вызываемая при задании параметра ADDNODE) будет запущена на корректном разделе базы данных. По завершении работы утилиты новый раздел базы данных останавливается.

2. Введите команду DB2STOP.

   После ввода команды DB2STOP в файл конфигурации узла будет внесена информация о новом разделе базы данных.

3. Введите команду DB2START, чтобы запустить систему базы данных.

Примечание: Возможно, придется дважды ввести команду DB2START, чтобы все разделы базы данных прочитали новый файл конфигурации узла.

4. При желании сделайте резервные копии всех баз данных на новом разделе базы данных.
5. При желании перераспределите данные, используя новый сервер. Подробности приводятся в разделе “Глава 11. Перераспределение данных между разделами баз данных” на стр. 327.

Внесение изменений в файл конфигурации узла вручную

После добавления одного или нескольких новых разделов к многораздельной системе баз данных, чтобы завершить обеспечение доступа к новому разделу, надо внести изменения в файл `db2nodes.cfg`. Если вы приняли решение внести изменения в файл конфигурации узла вручную, посмотрите ниже подробности требуемых действий. (Не забудьте, что при работе в Windows NT изменять файл конфигурации узла вручную нельзя.)

Примечание: Если вы приняли решение внести изменения в файл конфигурации узла при помощи менеджера баз данных, вернитесь к разделу “Изменение файла конфигурации узла при помощи менеджера баз данных” на стр. 321.

Дальнейшая процедура:

1. Отредактируйте файл `db2nodes.cfg`, добавив в него новый раздел базы данных.
2. Запустите новый узел командой:

   ```
   DB2START NODENUM номер_узла
   ```

   В качестве значения *номер_узла* укажите номер, который будет присвоен новому разделу базы данных.
3. Если новый сервер должен стать логическим разделом базы данных (это значит, что он - не узел 0), нужно при помощи команды **db2set** изменить переменную регистра DB2NODE, задав номер добавляемого раздела базы данных.
4. Запустите на новом разделе базы данных команду ADD NODE.

   Эта команда выполнит локальное создание раздела базы данных для каждой базы данных, которая уже существует в системе. Параметрам базы данных для нового раздела базы данных задаются значения по умолчанию; каждый раздел базы данных остается пустым, пока в него не поместят данные. Вы должны изменить значения параметров конфигурации базы данных, чтобы они соответствовали значениям для других разделов базы данных.
5. По завершении команды ADD NODE введите команду DB2START, чтобы запустить остальные разделы базы данных в системе.

Пока не будут успешно запущены все разделы, не следует пытаться выполнять общесистемные работы, такие как создание и отбрасывание базы данных.

6. При желании сделайте резервные копии всех новых разделов базы данных на новом сервере.
7. При желании перераспределите данные, используя новый раздел базы данных. Подробности приводятся в разделе “Глава 11. Перераспределение данных между разделами баз данных” на стр. 327.

Отбрасывание раздела базы данных из системы

Отбросить раздел базы данных можно при помощи команды DB2STOP с параметром DROP NODENUM или интерфейса API `sqlepstp`. Перед этим нужно убедиться, что отбрасываемый раздел не используется ни одной базой данных. Для соответствующей проверки введите команду DROP NODE VERIFY.

Нужно убедиться, что все транзакции, для которых этот раздел базы данных служил координатором, успешно приняты или отменены с выполнением отката. Возможно, потребуется провести аварийное восстановление на других серверах.

Например, если вы отбрасываете раздел-координатор (то есть узел координатора), и другой раздел участвовал в транзакции, завершившейся аварией перед отбрасыванием узла координатора, аварийный раздел не сможет запросить узел координатора о результате всех неоднозначных транзакций.

Чтобы отбросить раздел из системы многораздельных баз данных:

1. Перераспределите данные из всех баз данных, расположенных на этом узле. В результате отбрасываемый раздел не будет использоваться ни одной базой данных. Подробности приводятся в разделе “Глава 11. Перераспределение данных между разделами баз данных” на стр. 327.
2. Введите команду DROP NODE VERIFY или вызовите API `sqledrpn`, чтобы проверить, что сервер не используется.

В зависимости от полученного сообщения перейдите либо к шагу 3, либо к шагу 4.

3. Если вы получите сообщение SQL6034W (узел не используется ни одной базой данных), можете выполнить одно из следующих действий:
   a. Ввести команду DB2STOP с параметром DROP NODENUM, чтобы отбросить раздел базы данных. После успешного завершения команды система остановится.
   b. Если хотите, запустить менеджер баз данных командой DB2START.
4. Если вы получите сообщение SQL6035W (узел используется базой данных):
   a. При помощи команды REDISTRIBUTE NODEGROUP перераспределите данные базы, алиас которой указан в сообщении SQL6035W, из

отбрасываемого раздела по другим разделам. Пока это не будет сделано, отбросить раздел базы данных нельзя.

b. Отбросьте все мониторы событий, заданные в разделе базы данных.

c. Вернитесь к шагу 2 на стр. 323 и продолжите работу.

Проблемы при добавлении узлов в многораздельную базу данных

При добавлении узлов в многораздельную базу данных, содержащую одно или несколько системных временных табличных пространств, размер страниц которых отличается от размера страниц по умолчанию (4 Кбайта), вы можете получить сообщение об ошибке “SQL6073N Операция Add Node прервана” и SQLCODE. Это вызвано тем, что при создании узла существует только пул буферов IBMDEFAULTBP с размером страниц 4 Кбайта.

Например, для добавления узла в текущую многораздельную базу данных может использоваться команда **db2start**:

```
DB2START NODENUM 2 ADDNODE HOSTNAME newhost PORT 2
```

Если эта многораздельная база данных содержит системные временные табличные пространства, размер страниц которых равен размеру страниц по умолчанию, будет возвращено следующее сообщение:

```
SQL6075W Операция Start Database Manager успешно добавила узел.
Узел не будет активен, пока все узлы не будут остановлены и запущены еще раз.
```

Однако если системные временные табличные пространства этой многораздельной базы данных имеют размер страниц, отличный от размера страниц по умолчанию, будет возвращено сообщение:

```
SQL6073N Операция Add Node прервана. SQLCODE = "<-902>"
```

Для добавления узла можно также использовать команду ADD NODE, предварительно вручную добавив описание нового узла в файл `db2nodes.cfg`. После изменения этого файла и выполнения команды ADD NODE для многораздельной базы данных, содержащей системные временные табличные пространства, размер страниц которых равен размеру страниц по умолчанию, будет возвращено следующее сообщение:

```
DB20000I Команда ADD NODE выполнена успешно.
```

Однако если системные временные табличные пространства этой многораздельной базы данных имеют размер страниц, отличный от размера страниц по умолчанию, будет возвращено сообщение:

```
SQL6073N Операция Add Node прервана. SQLCODE = "<-902>"
```

Один из способов предотвратить описанную выше ошибку - выполнить команду:

```
DB2SET DB2_HIDDENBP=16
```

перед выполнением команды **db2start** или ADD NODE. Эта переменная реестра разрешает DB2 выделять скрытые пулы буферов по 16 страниц каждый, используя размер страницы, отличный от размера страницы по умолчанию. Это обеспечивает успешное выполнение операции ADD NODE.

Другой способ предотвращения этой ошибки - задать условие WITHOUT TABLESPACES в команде ADD NODE или **db2start**. После этого нужно будет создать пулы буферов, используя оператор CREATE BUFFERPOOL, и связать системные временные табличные пространства с пулом буферов, используя оператор ALTER TABLESPACE.

При добавлении узлов в существующую группу узлов, содержащую одно или несколько табличных пространств, размер страниц которых отличается от размера страниц по умолчанию (4 Кбайта), вы можете получить сообщение об ошибке “SQL0647N Пул буферов ″″ в настоящее время неактивен.”. Это происходит потому, что созданные на новом узле пулы буферов с размером страниц, отличным от размера страниц по умолчанию, не активируются для этих табличных пространств.

Например, для добавления узла в группу узлов можно использовать оператор ALTER NODEGROUP:

```
 DB2START
CONNECT TO mpp1
ALTER NODEGROUP ng1 ADD NODE (2)
```

Если эта группа узлов содержит табличные пространства, размер страниц которых равен размеру страниц по умолчанию, будет возвращено следующее сообщение:

```
SQL1759W Для изменения распределения данных для объектов
в группе узлов "<ng1>" требуется перераспределение группы узлов
со включением добавленных узлов или исключением отброшенных.
```

Однако если табличные пространства этой группы узлов имеют размер страниц, отличный от размера страниц по умолчанию, будет возвращено сообщение:

```
SQL0647N Пул буферов "" в настоящее время неактивен.
```

Один из способов предотвратить описанную выше ошибку - создать пулы буферов для каждого размера страниц и затем заново соединиться с базой данных перед выполнением оператора ALTER NODEGROUP:

```
 DB2START
CONNECT TO mpp1
CREATE BUFFERPOOL bp1 SIZE 1000 PAGESIZE 8192
CONNECT RESET
CONNECT TO mpp1
ALTER NODEGROUP ng1 ADD NODE (2)
```

Другой способ предотвращения этой ошибки - выполнить команду:

```
DB2SET DB2_HIDDENBP=16
```

перед выполнением команды **db2start** и операторов CONNECT и ALTER NODEGROUP.

Другая ошибка может возникнуть, когда оператор ALTER TABLESPACE используется для добавления табличного пространства к узлу. Например:

```
 DB2START
CONNECT TO mpp1
ALTER NODEGROUP ng1 ADD NODE (2) WITHOUT TABLESPACES
ALTER TABLESPACE ts1 ADD ('ts1') ON NODE (2)
```

Эта последовательность команд и операторов генерирует сообщение об ошибке SQL0647N (а не ожидаемое сообщение SQL1759W).

Чтобы правильно выполнить это изменение, необходимо заново соединиться с базой данных после выполнения оператора ALTER NODEGROUP... WITHOUT TABLESPACES.

```
DB2START
  CONNECT TO mpp1
  ALTER NODEGROUP ng1 ADD NODE (2) WITHOUT TABLESPACES
  CONNECT RESET
  CONNECT TO mpp1
  ALTER TABLESPACE ts1 ADD ('ts1') ON NODE (2)
```

Другой способ предотвращения этой ошибки - выполнить команду:

```
DB2SET DB2_HIDDENBP=16
```

перед выполнением команды **db2start** и операторов CONNECT, ALTER NODEGROUP и ALTER TABLESPACE.

Глава 11. Перераспределение данных между разделами баз данных

Перераспределение данных возможно только в среде многораздельных баз данных. Если вы работаете в среде одноразделных баз данных, приводимая ниже информация вам не потребуется.

Для перемещения данных между разделами базы данных в существующей группе узлов предназначена утилита перераспределения данных. При ее помощи можно:

- Распределять нагрузку по томам данных и процессорам между разделами баз данных.

  Это полезно, если у вас есть таблица базы данных, доступ к данным которой производится на регулярной основе.

- Ввести асимметрию при распределении данных между разделами базы данных.

  Это полезно, если у вас есть таблица базы данных, доступ только к части данных которой производится регулярно. В этой ситуации таблицу можно перераспределить таким образом, чтобы редко используемые данные хранились только на нескольких разделах базы данных в данной группе узлов, а часто используемые данные распределялись среди большего числа разделов. Это должно улучшить производительность и пропускную способность часто выполняемых программ.

Утилита перераспределения данных вызывается командой REDISTRIBUTE NODEGROUP. Дополнительную информацию о синтаксисе этой команды смотрите в руководстве *Command Reference*.

Чтобы сохранить совместное размещение таблиц, эта операция применяется ко всем таблицам в группе узлов, и перераспределение выполняется на уровне группы узлов, а не на уровне таблиц.

Чтобы добиться нужного вам распределения данных, утилита использует карту разделения для перемещения строк таблиц между разделами базы данных группы узлов. В зависимости от указанных вами опций утилита может сгенерировать карту разделения или использовать в качестве входной уже имеющуюся карту.

Примечания:

1. Вы должны на основании требований к пространству журнала, которое, как вы считаете, потребуется для перераспределения данных, задать размер файла журнала. Надо, чтобы журнал был достаточно велик, чтобы

зарегистрировать операции INSERT и DELETE во всех разделах базы данных, на которых перераспределяются данные.

2. Если вы хотите перераспределить данные в группе узлов, содержащей реплицируемые таблицы сводок, нужно сначала отбросить эти таблицы, перераспределить группу узлов, а затем создать эти таблицы снова. Группу узлов, содержащую реплицируемые таблицы сводок, нельзя перераспределить.

Как разделять данные

По умолчанию утилита перераспределения данных предполагает, что во все хеш-разделы хешируется одинаковое число строк, и равномерно разделяет хеш-разделы по всем разделам базы данных группы узлов. Если в некоторые хеш-разделы хешируется разное число строк, для задания текущего распределения можно использовать *файл распределения*. Этот файл содержит по одному значению для каждого из 4096 хеш-разделов. Каждое значение используется в качестве *веса* соответствующего хеш-раздела. Утилита перераспределения данных генерирует карту разделения назначения, в которой у всех разделов базы данных вес примерно один и тот же. Таким образом, при помощи файла распределения можно получить равномерное распределение данных по разделам, даже если сами данные распределены асимметрично.

Для создания файла распределения данных можно использовать опцию ANALYZE утилиты `AutoLoader`. Этот файл можно использовать в качестве входа утилиты перераспределения данных. Дополнительную информацию об утилите `AutoLoader` смотрите в руководстве *Data Movement Utilities Guide and Reference*.

Другой вариант - использовать для определения текущего распределения данных между хеш-разделами или разделами базы данных функции SQL PARTITION и NODENUMBER. (Функция PARTITION используется для определения распределения между хеш-разделами.) Эту информацию можно использовать для получения файла распределения и карты разделения назначения.

Например, чтобы посмотреть, в каких разделах базы данных аномально много строк при неравномерном распределении данных, введите запрос:

```
SELECT PARTITION(имя_столбца), COUNT(*) FROM имя_таблицы
   GROUP BY PARTITION(имя_столбца)
   ORDER BY PARTITION(имя_столбца) DESC
   FETCH FIRST 100 ROWS ONLY
```

В качестве `имя_таблицы` выберите наибольшую таблицу, а `имя_столбца` - подходящий столбец из этой таблицы.

Добавление и отбрасывание разделов баз данных

Для добавления или отбрасывания разделов баз данных в группе узлов используется оператор ALTER NODEGROUP. При добавлении разделов баз данных разделы должны быть заранее определены в файле конфигурации узла.

После использования оператора ALTER NODEGROUP создается новая карта разделения. Эта карта разделения может использоваться в качестве карты разделения назначения при использовании утилиты перераспределения данных. (Другой вариант - создать карту разделения назначения самостоятельно.)

Если при использовании оператора ALTER NODEGROUP вы указываете условие WITHOUT TABLESPACES, необходимо до перераспределения данных добавить в новые разделы базы данных контейнеры табличных пространств. Дополнительную информацию об операторе ALTER NODEGROUP смотрите в справочнике *SQL Reference*.

Задание карты разделения назначения

Утилита перераспределения данных использует для перераспределения данных карту разделения. Она может создать собственную карту разделения назначения или использовать заданную вами карту. Если карту разделения будете создавать вы, ее элементы определяют тип группы узлов, получающейся после перераспределения данных:

- 1 элемент для одноразделной группы узлов
- 4096 элементов для многоразделной группы узлов

Если в карте разделения назначения есть несколько разделов базы данных, для всех таблиц в группе узлов должен быть задан один и тот же ключ разделения.

Карта разделения назначения может содержать только номера разделов базы данных, определенные в таблице каталога SYSCAT.NODEGROUPDEF, за исключением тех, у которых значение IN_USE равно 'T'. ('T' означает, что раздел не находится в карте разделения назначения.) Все разделы базы данных, значение IN_USE которых равно 'D' (drop - отбросить), не находящиеся в карте разделения назначения, отбрасываются после успешного завершения перераспределения.

Как данные перераспределяются между разделами базы данных

Операция перераспределения данных выполняется на наборе таблиц в заданной группе узлов базы данных. (Перед выполнением этой операции программа должна соединиться с базой данных в разделе базы данных каталога.) Утилита использует исходную карту разделения и карту разделения назначения для определения, какие хеш-разделы надо перенести на новое место (которое задается номером нового раздела базы данных). Все строки, относящиеся к

разделу, перенесенному в новое место, перемещаются из раздела базы данных, заданного в исходной карте разделения, в раздел базы данных, заданный в карте разделения назначения.

Утилита перераспределения данных:

1. Получает для карты разделения назначения новый ID карты разделения и записывает его в производную таблицу каталога SYSCAT.PARTITIONMAPS.
2. Изменяет столбец REBALANCE_PMAP_ID в производной таблице SYSCAT.NODEGROUPS для данной группы узлов на новый ID карты разделения.
3. Добавляет все новые разделы базы данных в производную таблицу каталога SYSCAT.NODEGROUPDEF.
4. Меняет значение столбца IN_USE в производной таблице каталога SYSCAT.NODEGROUPDEF для всех отбрасываемых разделов базы данных на 'D'.
5. Выполняет для изменений каталога COMMIT.
6. Создает файлы базы данных для всех новых разделов базы данных.
7. Перераспределяет данные, таблицу за таблицей для всех таблиц в данной группе узлов. Этот процесс описан в разделе "Как перераспределяются данные в таблицах".
8. Удаляет файлы базы данных и элементы производной таблицы каталога SYSCAT.NODEGROUPDEF для разделов базы данных, ранее выделенных для отбрасывания.
9. Изменяет в записи группы узлов в производной таблице каталога SYSCAT.NODEGROUPS значение PMAP_ID на значение REBALANCE_PMAP_ID, а значение REBALANCE_PMAP_ID - на NULL.
10. Удаляет старую карту разделения из производной таблицы каталога SYSCAT.PARTITIONMAPS.
11. Выполняет для всех изменений COMMIT.

Как перераспределяются данные в таблицах

При распределении данных в таблице утилита:

1. Блокирует строку для данной таблицы в таблице каталога SYSTABLES.
2. Запрещает все пакеты, использующие эту таблицу. Соответствующий этой таблице ID карты разделения изменится, так как таблица будет перераспределена. Поскольку пакеты запрещены, компилятор должен получить для этой таблицы новую информацию о разделах и создать соответствующие пакеты.
3. Блокирует таблицу в режиме монопольного доступа.

4. Перераспределяет данные в таблице посредством операций DELETE и INSERT.
5. Если перераспределение завершено удачно, утилита:
   a. Выполняет для этой таблицы COMMIT.
   b. Переходит к следующей таблице в этой группе узлов.

   Если операция прерывается до полного перераспределения таблицы, утилита:
   a. Выполняет для произведенных над таблицей изменений операцию ROLLBACK.
   b. Прекращает перераспределение и возвращает ошибку.

При перераспределении данных важно оценить требования к пространству для журнала. Журнал должен быть достаточно велик, чтобы зарегистрировать операции INSERT и DELETE во всех разделах базы данных, в которых перераспределяются данные. Самые большие расходы на журнал будут либо в разделе базы данных, который потеряет наибольшее количество данных, либо в разделе базы данных, который приобретет наибольшее количество данных. Если производится перенос в большее количество разделов базы данных, при вычислении количества операций INSERT и DELETE должно помочь соотношение текущего числа разделов базы данных и нового числа разделов базы данных.

Например, при переходе с четырех на пять разделов базы данных примерно двадцать процентов данных будет перенесено из четырех старых разделов базы данных в новый раздел базы данных. Это значит, что над каждым из четырех старых разделов базы данных будет произведено двадцать процентов операций DELETE от общего объема данных в каждом разделе базы данных. На новом разделе базы данных будут проведены все операции INSERT (то есть эквивалент равного числа операций DELETE из всех четырех старых разделов базы данных).

Приведенный пример предполагает равномерное распределение данных. Возможен и случай с неравномерным распределением данных, например, если в ключе разделения есть много пустых значений. В этом случае все эти строки окажутся в одном разделе базы данных при старой схеме разделения и в другом разделе базы данных при новой схеме разделения. В результате объем пространства журнала, необходимого для этих двух разделов базы данных, может сильно увеличиться относительно объема, вычисленного для равномерного распределения.

При проведении фактических подсчетов следует умножить процентную долю изменений (например, двадцать процентов) на размер наибольшей таблицы. Это делается потому, что перераспределение каждой таблицы выполняется в качестве одной транзакции.

Примечание: Однако возможен случай, когда наибольшая таблица распределена равномерно, а, скажем, вторая по величине таблица содержит один или несколько перегруженных разделов базы данных. В этом случае случае, возможно, следует использовать для оценки вторую, а не первую по величине таблицу.

Когда вы вычислите максимальный объем данных, которые должны быть вставлены и удалены в разделе базы данных, удвойте его, чтобы получить максимальный размер активного журнала. Если это превысит предельную величину активного журнала (32 Гбайта), перераспределение надо выполнять в несколько шагов. Утилита под названием “makepmap” позволяет сгенерировать серию карт разделения назначения, по одной на каждый шаг.

Восстановление при ошибках перераспределения

Когда начинает выполняться операция перераспределения, в подкаталог `redist` каталога `sqllib` записывается файл. В этом файле состояния записываются все проводимые над разделами базы данных операции, имена перераспределенных таблиц и состояние завершения операции. Если таблицу нельзя перераспределить, в файл состояния записывается ее имя и соответствующий SQLCODE. Если операцию перераспределения не удается начать из-за неверных входных параметров, файл не создается и возвращается SQLCODE.

Этот файл именуется по следующему соглашению:

имябазыданных.имягруппыузлов.системноевремя (для платформ UNIX)
имябазыданных\имягруппыузлов\дата\время (для прочих платформ)

Примечание: Для платформ, отличающихся от UNIX, используются только первые восемь (8) байт параметра `имягруппыузлов`.

Если операция перераспределения данных не срабатывает, некоторые таблицы могут оказаться перераспределенными, а некоторые - нет. Это связано с тем, что перераспределение данных выполняется по одной таблице за раз. Есть два варианта восстановления:

- При опции CONTINUE операция продолжается, и оставшиеся таблицы перераспределяются.
- При опции ROLLBACK перераспределение отменяется, и перераспределенные таблицы возвращаются в исходное состояние. Операция отката может занять примерно столько же времени, сколько и первоначальная операция перераспределения.

Чтобы вы смогли воспользоваться любой из этих опций, последняя операция перераспределения данных должна неудачно завершиться так, чтобы в столбец REBALANCE_PMID в таблице каталога SYSNODEGROUPS было записано ненулевое значение.

Если файл состояния был по неосторожности удален, все еще можно попытаться выполнить операцию CONTINUE.

Перераспределение данных и другие операции

Во время работы утилиты можно выполнять указанные ниже действия над объектами группы узлов. Над перераспределяемой таблицей нельзя выполнять и их. Вы можете:

- Создавать индексы для других таблиц. Оператор CREATE INDEX использует карту разделения указанной таблицы.
- Отбрасывать другие таблицы. Оператор DROP TABLE использует карту разделения указанной таблицы.
- Отбрасывать индексы для других таблиц. Оператор DROP INDEX использует карту разделения указанной таблицы.
- Выполнять запросы по другим таблицам.
- Изменять другие таблицы.
- Создавать новые таблицы в табличном пространстве, определенном в группе узлов. Оператор CREATE TABLE использует карту разделения назначения.
- Создавать в группе узлов табличные пространства.

Во время работы утилиты нельзя выполнять:

- Другие операции перераспределения на этой группе узлов
- Оператор ALTER TABLE для любых таблиц на этой группе узлов
- Отбрасывание группы узлов
- Изменение группы узлов.

После перераспределения данных

После завершения перераспределения данных внутри группы узлов настоятельно рекомендуется запустить RUNSTATS, чтобы обновить статистику, связанную с таблицами, которые могли быть перераспределены.

Дополнительную информацию о команде RUNSTATS смотрите в руководстве *Command Reference*.

Глава 12. Измерение производительности

Тестовые измерения - нормальная часть разработки прикладной программы. Это совместная работа программистов и администраторов баз данных по измерению и повышению производительности прикладной программы. Даже если текст программы написан с максимально возможной эффективностью, возможен выигрыш в производительности за счет настройки параметров конфигурации базы данных и менеджера баз данных, а также параметров самой прикладной программы в соответствии с требованиями среды.

Есть несколько типов измеряемых показателей. Показатель *число транзакций в секунду* характеризует пропускную способность менеджера баз данных в определенных лабораторных условиях. Аналогичный показатель *прикладной программы* также характеризует пропускную способность, но в условиях, приближенных к условиям применения программы. В ходе тестовых измерений для настройки параметров конфигурации на основе "приближенных к реальным" условий производятся многократные запуски операторов SQL из прикладной программы с перебором значений параметров, пока не будет достигнута максимальная производительность.

Методы тестовых измерений, описанные в этом разделе, ориентированы на настройку параметров конфигурации. Однако те же основные приемы позволяют настраивать другие влияющие на производительность факторы, к которым относятся:

- Операторы SQL
- Индексы
- Конфигурация табличного пространства
- Текст программы
- Конфигурация оборудования.

Тестовые измерения помогают оценить влияние различных условий на работу менеджера баз данных. Вы можете создать сценарии, которые протестируют обработку тупиковых ситуаций, производительность утилит, разные методы загрузки данных, характеристики скорости транзакций при добавлении пользователей, а также влияние на прикладную программу использования нового выпуска продукта.

Далее описаны следующие темы:

Методология тестовых измерений

Техника тестовых измерений основана на научном подходе. Создается воспроизводимая среда, в которой один и тот же тест, выполняемый в одних и тех же условиях, позволяет достичь сопоставимых результатов.

Тестовые измерения могут также начаться с запуска тестовой программы в нормальной среде. После уточнения проблем производительности можно разработать специальные тестовые случаи, чтобы ограничить сферу тестирования исследуемой функции. Специальные тестовые случаи не требуют эмуляции всей программы для получения нужной информации. Начните с простых измерений, а к более сложным переходите только при необходимости.

Требования к хорошим тестовым измерениям:
- Все тесты повторяемы.
- Каждый прогон теста начинается в одном и том же состоянии системы.
- Активны только тестируемые функции или программы (если в сценарий не входит учет посторонних процессов в системе).

  Примечание: Запущенные программы используют память, даже когда они свернуты и не активны. Это повышает вероятность подкачки страниц, искажающей результаты тестовых измерений и нарушающей принцип воспроизводимости.
- Устройства и программы, используемые при тестировании, соответствуют реальным условиям вашей работы.

Как и при любых тестовых измерениях, надо сначала разработать сценарий, а потом выполнить его несколько раз. Фиксация ключевой информации после каждого прогона очень важна для определения изменений, необходимых для повышения производительности программы и базы данных.

Подготовка к тестовым измерениям

Логическое проектирование базы данных, для которой вы пишете прикладную программу, должно быть завершено до начала тестовых измерений производительности. Таблицы, производные таблицы и индексы должны быть заданы и заполнены. Таблицы должны быть нормализованы, пакеты программ должны быть связаны, и таблицы должны быть заполнены реалистичными данными.

Нужно определиться с окончательной физической структурой базы данных. Объекты менеджера баз данных должны быть помещены на свои постоянные места на диске, должны быть определены размеры файлов журналов и места для рабочих файлов и резервных копий, и должны быть протестированы процедуры резервного копирования. Кроме того, нужно проверить, что включены все возможные опции производительности в пакетах (например - блокировка строк).

Написание и отладка прикладной программы должны достичь той стадии, на которой возможно создание программ тестовых измерений (смотрите раздел “Создание программы тестовых измерений” на стр. 338). Некоторые практические ограничения прикладной программы могут никак не проявиться при тестовых измерениях; однако цель описываемых здесь испытаний - измерение производительности, а не выявление недостатков и отклонений.

Программа тестовых измерений должна выполняться в среде, максимально приближенной к окончательной производственной; в идеале - на той же модели сервера с той же памятью и той же конфигурацией дисков. Это особенно важно, когда прикладная программа в конечном счете будет работать с большим числом пользователей и большими объемами данных. Должны быть настроены операционная система и все средства связи и работы с файлами, прямо используемые в тестовых измерениях.

Важно также, чтобы тестовые измерения выполнялись на базе данных производственного размера. Отдельный оператор SQL должен возвращать столь же большой объем данных и инициировать столь же большой объем сортировки, что и в производственных условиях. Следование этому правилу позволить достичь характерных требований памяти со стороны прикладной программы.

Тестируемые операторы SQL делятся на *характерные* и *наихудшие*, как описано ниже:

Характерные SQL

К характерным SQL относятся те операторы, которые выполняются при типичных действиях тестируемой программы. Выбираемый набор этих операторов зависит от характера прикладной программы. Например, программа ввода данных может протестировать оператор INSERT, а банковская транзакция может протестировать FETCH, UPDATE и несколько операторов INSERT. Частота выполнения и объем обработки данных для операторов, выбранных как характерные, должны оцениваться как средние. Если же эти объемы больше, операторы следует отнести в категорию *наихудших*, даже если это типичные операторы SQL.

Наихудшие SQL

В эту категорию попадают следующие операторы:

- Операторы, которые часто выполняются.
- Операторы с большим объемом обработки данных.
- Операторы, критичные по времени.

  Примером служит прикладная программа, запускаемая при приеме телефонного звонка от заказчика, и те операторы в этой программе, которые должны получить и обновить информацию заказчика, пока он ждет.

- Операторы с наибольшим числом объединяемых таблиц или с наиболее сложными SQL в данной прикладной программе.

  Примером служит банковская прикладная программа, порождающая для заказчиков отчет о месячной активности по всем типам их счетов. Обычная таблица позволит создать список с адресом заказчика и номерами учетных записей; однако потребуется объединение еще с несколькими таблицами, чтобы обработать и просуммировать всю требуемую информацию о транзакциях с этими счетами. Умножьте работу, требуемую для одного счета, на несколько тысяч счетов, подлежащих обработке в тот же период времени, и потенциальная экономия времени определит требования к производительности.
- Операторы с неэффективным путем доступа, например, редко выполняемые и не поддерживаемые индексами, которые создаются для используемых таблиц.
- Операторы с большим временем обработки.
- Оператор, выполняемый при инициализации прикладной программы, но имеющий непропорциональные требования к ресурсам.

  Например, прикладная программа, которая генерирует список работ со счетами, подлежащих обработке в течение дня. При запуске этой программы первый основной оператор SQL производит 7-кратное объединение для создания огромного списка всех счетов, за которые отвечает пользователь этой прикладной программы. Этот оператор, возможно, выполняется всего несколько раз в день, но без надлежащей настройки его выполнение может занять не одну минуту.

Создание программы тестовых измерений

Существует ряд факторов, которые следует принимать во внимание при разработке и использовании программ тестовых измерений. Поскольку главная цель программы состоит в имитации пользовательской программы, общая структура программы может быть самой разной. Можно в качестве теста прогонять всю прикладную программу, добавив только средства для измерения времени анализируемых операторов SQL. Для больших или сложных прикладных программ может оказаться практичнее ограничиться блоками с важными операторами.

Другой подход состоит в том, что для тестирования определенных операторов SQL в программу тестирования включаются только сами эти операторы, а также необходимые операторы (CONNECT, PREPARE, OPEN и другие) и механизм измерения времени.

Другой важный фактор - используемый тип тестовых измерений. Один из вариантов предусматривает многократное выполнение операторов SQL в течение некоторого интервала времени. Отношение числа выполненных операторов к этому временному интервалу определит производительность

прикладной программы. Другой вариант - просто определить время, требуемое для выполнения отдельных операторов SQL.

Независимо от типа программы тестовых измерений эффективная система измерения времени необходима для вычисления времени обработки как отдельных операторов SQL, так и всей прикладной программы. Для имитации прикладных программ, в которых отдельные операторы SQL будут выполняться изолированно, может оказаться важным учитывать время для операторов CONNECT, PREPARE и COMMIT. С другой стороны, в случае прикладных программ, обрабатывающих много разных операторов, бывает достаточно одного оператора CONNECT или COMMIT, так что прежде всего следует сосредоточиться на времени выполнения отдельного оператора.

Хотя время обработки каждого запроса - важный фактор в анализе производительности, его определение может и не выявить узких мест. Например, информация об использовании процессора, блокировках и операциях ввода/вывода пула буферов может показать, что прикладную программу ограничивают операции ввода/вывода, поэтому она не использует процессор на полную мощность. Программа тестовых измерений должна позволять, если потребуется, получать такую информацию для более подробного анализа.

Не всем прикладным программам требуется посылать полный набор строк, полученных в результате запроса, на какие-либо внешние устройства. Например, некоторые программы могут использовать полный набор ответов как входные данные для другой программы (это значит, что никакие строки, выводимые первой программой, не посылаются на вывод). Форматирование данных для вывода на экран обычно имеет высокую процессорную стоимость и может не отражать потребностей пользователя. Чтобы обеспечить точную имитацию, программа тестовых измерений должна отражать способ обработки строк конкретной прикладной программой. Если строки действительно посылаются на внешнее устройство, неэффективное форматирование может потребить большую часть времени процессора и неправильно представить фактическую производительность самого оператора SQL.

Инструмент тестовых измерений db2batch: Инструмент тестовых измерений (`db2batch`) находится в подкаталоге `bin` каталога `sqllib` экземпляра. Этот инструмент учитывает многие из вышеописанных особенностей, касающихся создания программы тестовых измерений. Он читает операторы SQL из плоского файла или стандартного ввода, динамически описывает и подготавливает операторы и возвращает набор ответов. Дополнительная гибкость обеспечивается возможностью управлять размеров набора ответов, а также числом строк, которые должны быть посланы из этого набора ответов на внешнее устройство.

Можно также задать уровень предоставляемой информации, связанной с производительностью, включая время обработки, использование процессора и

пула буферов, блокировки и другую статистику, собираемую монитором баз данных. Если вы измеряете время для набора операторов SQL, db2batch также просуммирует результаты производительности и вычислит среднее арифметическое и среднее геометрическое. Дополнительную информацию о синтаксисе вызова и опциях db2batch смотрите в руководстве *Command Reference*. Можно также ввести db2batch -h в командной строке, чтобы узнать возможный синтаксис и опции.

Ниже приведен пример использования db2batch с входным файлом db2batch.sql:

```
-- db2batch.sql
-- ------------
--#SET PERF_DETAIL 3 ROWS_OUT 5

-- Этот запрос выводит список сотрудников, название их отдела
-- и число работ, которые им поручены, для
-- сотрудников, которым поручено по несколько работ на не
-- полный рабочий день.
--#COMMENT Запрос 1
select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) > 2;
--#SET PERF_DETAIL 1 ROWS_OUT 5
--#COMMENT Запрос 2
select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) <= 2;
```

Рисунок 31. Пример входного файла для тестовых измерений: db2batch.sql

Вызов инструмента тестовых измерений:

```
db2batch -d sample -f db2batch.sql
```

вернет следующий вывод:

```
--#SET PERF_DETAIL 3 ROWS_OUT 5
Query 1

Statement number: 1

select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) > 2
```

Рисунок 32. Пример вывода db2batch (Часть 1)

```
LASTNAME        FIRSTNME     DEPTNAME                       NUM_ACT
------------------------------------------------------------------------
JEFFERSON       JAMES        ADMINISTRATION SYSTEMS                3
JOHNSON         SYBIL        ADMINISTRATION SYSTEMS                4
NICHOLLS        HEATHER      INFORMATION CENTER                    4
PEREZ           MARIA        ADMINISTRATION SYSTEMS                4
SMITH           DANIEL       ADMINISTRATION SYSTEMS                7
Number of rows retrieved is:        5
Number of rows sent to output is:   5
Elapsed Time is:           0.074      seconds
Locks held currently                        = 0
Lock escalations                            = 0
Total sorts                                 = 5
Total sort time (ms)                        = 0
Sort overflows                              = 0
Buffer pool data logical reads              = 13
Buffer pool data physical reads             = 5
Buffer pool data writes                     = 0
Buffer pool index logical reads             = 3
Buffer pool index physical reads            = 0
Buffer pool index writes                    = 0
Total buffer pool read time (ms)            = 23
Total buffer pool write time (ms)           = 0
Asynchronous pool data page reads           = 0
Asynchronous pool data page writes          = 0
Asynchronous pool index page reads          = 0
Asynchronous pool index page writes         = 0
Total elapsed asynchronous read time        = 0
Total elapsed asynchronous write time       = 0
Asynchronous read requests                  = 0
LSN Gap cleaner triggers                    = 0
Dirty page steal cleaner triggers           = 0
Dirty page threshold cleaner triggers       = 0
Direct reads                                = 8
Direct writes                               = 0
Direct read requests                        = 4
Direct write requests                       = 0
Direct read elapsed time (ms)               = 0
Direct write elapsed time (ms)              = 0
Rows selected                               = 5
Log pages read                              = 0
Log pages written                           = 0
Catalog cache lookups                       = 3
Catalog cache inserts                       = 3
Buffer pool data pages copied to ext storage    = 0
Buffer pool index pages copied to ext storage   = 0
Buffer pool data pages copied from ext storage  = 0
Buffer pool index pages copied from ext storage = 0
Total Agent CPU Time (seconds)              = 0.02
Post threshold sorts                        = 0
Piped sorts requested                       = 5
Piped sorts accepted                        = 5
```

Рисунок 33. Пример вывода db2batch (Часть 1)

```
--#SET PERF_DETAIL 1 ROWS_OUT 5
Запрос 2
Номер оператора: 2
select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) <= 2
LASTNAME         FIRSTNME      DEPTNAME                        NUM_ACT
------------------------------------------------------------------------
GEYER            JOHN          SUPPORT SERVICES                      2
GOUNOT           JASON         SOFTWARE SUPPORT                      2
HAAS             CHRISTINE     SPIFFY COMPUTER SERVICE DIV.          2
JONES            WILLIAM       MANUFACTURING SYSTEMS                 2
KWAN             SALLY         INFORMATION CENTER                    2
Number of rows retrieved is:        8
Number of rows sent to output is:   5
Elapsed Time is:           0.037      seconds
Summary of Results
==================
                 Elapsed             Agent CPU         Rows      Rows
Statement #      Time (s)            Time (s)          Fetched   Printed
1                      0.074               0.020         5         5
2                      0.037       Not Collected         8         5
Arith. mean      0.055
Geom.  mean      0.052
```

Рисунок 34. Пример вывода из db2batch (Часть 2)

Приведенный пример вывода содержит отдельные элементы данных, возвращаемые монитором баз данных. Дополнительную информацию об этих и других элементах монитора смотрите в руководстве *System Monitor Guide and Reference*.

В следующем примере (на UNIX) возвращается только сводная таблица.

```
db2batch -d sample -f db2batch.sql -r /dev/null,
```

Этот вызов возвращает только сводную таблицу. При помощи опции `-r outfile1` заменен на `/dev/null`, а `outfile2` (который содержит только сводную таблицу) пуст, так что `db2batch` посылает вывод на экран:

```
Summary of Results
==================
                Elapsed             Agent CPU         Rows      Rows
Statement #     Time (s)            Time (s)          Fetched   Printed
1                    0.074                0.020         5         5
2                    0.037       Not Collected          8         5
Arith. mean     0.055
Geom.  mean     0.052
```

Рисунок 35. Пример вывода db2batch -- только сводная таблица

У этого инструмента тестовых измерений также есть опция CLI. Эта опция позволяет задать размер кэша. В следующем примере db2batch запускается в режиме CLI с кэшем на 30 операторов:

```
db2batch -d sample -f db2batch.sql -cli 30
```

db2batch можно запустить удаленно. Если вы используете параметры команды

```
-f <имя-файла>
```

или

```
-o <опции>
```

для инструмента тестовых измерений:

- Управляющие опции

  ```
  perf_detail
  ```

  и

  ```
  -p <perf_detail>
  ```

  (задающие уровень возвращаемой информации о производительности) не поддерживаются при удаленном запуске, если имеют значения больше единицы.
- Управляющие опции

  ```
  perf_detail
  ```

  и

  ```
  -p <perf_detail>
  ```

  (задающие уровень возвращаемой информации о производительности) не поддерживаются для DB2 Universal Database for Windows 3.x или DOS, если имеют значения больше единицы.

Кроме этих случаев, управляющие опции

```
perf_detail
```

и

`-p <perf_detail>`

поддерживаются для всех платформ DB2 Universal Database.

Выполнение тестовых измерений

Один из типов тестовых измерений базы данных включает выбор параметра конфигурации и выполнение теста с различными значениями этого параметра, пока не будет достигнут максимальный выигрыш. Отдельный тест должен включать несколько выполнений программы (например, 20 или 30 раз) с одним и тем же значением параметра, чтобы получить среднее время выполнения, которое точнее покажет влияние изменений параметра.

При тестовых измерениях первый (″прогревочный″) проход надо рассматриваться как особый случай, отличный от последующих проходов (так называемых нормальных запусков). Необходимость этого связана с тем, что на первом проходе выполняются начальные действия (например, инициализация пула буфера). Следовательно, такой проход займет несколько больше времени, чем обычный проход. Информация первого прохода *может* важной, но статистически не значимой. Поэтому при вычислении среднего времени или нагрузки процессора для данного набора значений параметров используйте результаты обычных проходов.

Для создания начального прохода тестовых измерений можно воспользоваться мастером по конфигурированию производительности. Вопросы, задаваемые мастером по конфигурированию производительности, дают представление о некоторых факторах, которые стоит учитывать при настройке среды для нормальных запусков в рамках тестовых измерений. Чтобы использовать мастер по конфигурированию производительности, введите `db2cc`, чтобы вызвать Центр управления, и дальнейшие действия выполняйте из него.

Если вы проводите тестовые измерения при помощи отдельных запросов, нужно убедиться, что сведено к минимуму возможное влияние предыдущих запросов. Для этого надо очистить пул буферов, для чего можно просто прочитать некоторое число страниц (не имеющих отношения к запросу) и заполнить тем самым пул буферов.

Завершив запуски с одним набором значений параметров, можно изменить один параметр. Но между запусками необходимо возвращать среду тестовых измерений в исходное состояние, для чего:

- Верните в исходное состояние данные прикладной программы и статистику менеджера баз данных. Если статистика каталогов была обновлена для тестирования, убедитесь, что во всех запусках используются одни и те же значения для статистики. Если в ходе тестов происходит обновление данных, эти данные должны быть согласованными. Для этого можно:

  - Восстановить всю базу данных с помощью утилиты RESTORE. Резервная копия базы данных окажется в предыдущем состоянии, готовая к следующему тесту.
  - Восстановить экспортированную копию данных с помощью утилиты IMPORT или LOAD. Этот метод позволяет восстановить только затронутые тестированием данные. Для таблиц и индексов, содержащих эти данные, нужно запустить утилиты REORG и RUNSTATS.
- Вернуть прикладную программу в исходное состояние, заново связав ее с базой данных.

Особенности тестовых измерений на OS/2:

- Если при выполнении сценария происходит подкачка страниц, убедитесь, что SWAPPER.DAT вернулся к начальному размеру.
- При необходимости для обеспечения воспроизводимости перезапускайте систему.

Программа тестовых измерений должна выдавать идентификатор теста, номер прохода тестовой программы, номер оператора и время выполнения. Сводка результатов после серии тестовых измерений может выглядеть примерно так:

```
Test     Iter.   Stmt    Timing         SQL Statement
Numbr    Numbr   Numbr   (hh:mm:ss.ss)
 002      05      01     00:00:01.34    CONNECT TO SAMPLE
 002      05      10     00:02:08.15    OPEN cursor_01
 002      05      15     00:00:00.24    FETCH cursor_01
 002      05      15     00:00:00.23    FETCH cursor_01
 002      05      15     00:00:00.28    FETCH cursor_01
 002      05      15     00:00:00.21    FETCH cursor_01
 002      05      15     00:00:00.20    FETCH cursor_01
 002      05      15     00:00:00.22    FETCH cursor_01
 002      05      15     00:00:00.22    FETCH cursor_01
 002      05      20     00:00:00.84    CLOSE cursor_01
 002      05      99     00:00:00.03    CONNECT RESET
```

Рисунок 36. Пример результатов тестовых измерений

Примечание: Данные в приведенном отчете служат только для иллюстрации. Они **не** являются результатами измерения.

Изучение этого отчета показывает, что соединение CONNECT (оператор 01) заняло 1,34 секунды, открытие указателя OPEN CURSOR (оператор 10) заняло 2 минуты 8,15 секунд, выборка FETCHES (оператор 15) вернула семь строк, причем наибольшая задержка составила 0,28 секунд, закрытие указателя CLOSE CURSOR (оператор 20) заняло 0,84 секунд, а сброс соединения CONNECT RESET (оператор 99) - 0,03 секунд.

Может оказаться удобнее, чтобы ваша программа выводила данные в формате ASCII с разделителями, откуда их можно будет импортировать в таблицу базы данных или в электронную таблицу для последующего статистического анализа.

Пример отчета, выводимого программой тестовых измерений:

```
      PARAMETER        VALUES FOR EACH BENCHMARK TEST
      TEST NUMBER      001      002      003     004      005
      locklist         63       63       63       63       63
>>    buffpage         1000     1175     1250     1325     1400     <<
      maxappls         8        8        8        8        8
      applheapsz       48       48       48       48       48
      dbheap           128      128      128      128      128
      sortheap         256      256      256      256      256
      maxlocks         22       22       22       22       22
      stmtheap         1024     1024     1024     1024     1024
      SQL STMT         AVERAGE TIMINGS (seconds)
        01             01.34    01.34    01.35    01.35    01.36
        10             02.15    02.00    01.55    01.24    01.00
        15             00.22    00.22    00.22    00.22    00.22
        20             00.84    00.84    00.84    00.84    00.84
        99             00.03    00.03    00.03    00.03    00.03
```

Рисунок 37. Пример отчета о тестовых измерениях времени

Примечание: Данные в приведенном отчете служат только для иллюстрации. Они **не** являются результатами какого-либо измерения.

Изучение данных в этом примере показывает, что изменение параметра *buffpage* последовательно снижает время открытия указателя OPEN CURSOR с 2,15 до 1,00 секунд. (Предполагается, что есть только один пул буферов, размер которого (NPAGES) задан -1. Это значит, что размер пула буферов управляется параметром *buffpage*.)

В целом, тестовые измерения прикладной программы сводятся к следующим шагам:

Шаг 1 Оставьте значения **по умолчанию** для всех параметров настройки базы данных и менеджера баз данных, кроме:

- Параметров, важных для рабочей нагрузки и целей теста. (Времени на полный цикл тестовых измерений и настройку всех параметров обычно не хватает; лучше в качестве отправной точки сразу назначить некоторым параметрам предполагаемые оптимальные значения.)
- Размеров журнала, которые следует определить в ходе тестирования единицы и системного тестирования прикладной программы. (Дополнительную информацию смотрите в разделе “Размер файлов журнала (logfilsiz)” на стр. 435.)
- Всех параметров, которые необходимо изменить, чтобы прикладную программу можно было выполнить (чтобы предотвратить отрицательные коды возврата SQL, например, из-за нехватки памяти для кучи операторов).

Выполните набор проходов для этого начального случая и вычислите среднее время или загрузку процессора.

Шаг 2 Выберите для тестирования один и только один параметр, и измените его значение.

Шаг 3 Выполните очередной набор проходов и вычислите среднее время или загрузку процессора.

Шаг 4 Дальнейшие действия зависят от результатов тестовых измерений:

- Если наблюдается повышение производительности, измените значение того же параметра и вернитесь к Шагу 3. Продолжайте изменять этот параметр, пока не получите максимальный выигрыш.
- Если производительность снижается или остается неизменной, восстановите предыдущее значение параметра, вернитесь к Шагу 2 и выберите новый параметр. Повторяйте эту процедуру, пока не протестируете все параметры.

Примечание: Если изобразить показатели производительности на графике, цель поиска - точка, в которой кривая выходит на плато или начинает спускаться.

Вы можете написать программу-драйвер, которая поможет выполнить тестовые измерения. Программу-драйвер можно написать, например, на языке REXX, а для платформ UNIX - в виде сценария оболочки.

Эта программа-драйвер будет вызывать программу тестовых измерений, передавать ей нужные параметры, прогонять тест нужное число раз, восстанавливать согласованное состояние среды, задавать следующий тест с новыми значениями параметров и собирать/объединять результаты тестирования. Можно написать настолько гибкую программу, чтобы она могла проводить всю серию тестовых измерений, анализировать результаты и выдавать отчет о конечных и лучших значениях параметров в данном тесте.

Глава 13. Конфигурирование DB2

Параметры конфигурации - это значения, определяющие рабочие характеристики базы данных или системы управления базами данных.

Параметры конфигурации менеджера баз данных существуют на серверах и клиентах; однако на клиенте можно задать только некоторые параметры конфигурации менеджера баз данных. Эти параметры составляют подмножество параметров конфигурации управления базами данных, которые могут быть заданы на сервере. В зависимости от типа продукта DB2 Universal Database, который вы используете, параметры конфигурации имеют свои особенности. Например, в DB2 Extended Enterprise Edition один файл конфигурации менеджера баз данных совместно используется всеми серверами разделов экземпляра. А каждый раздел базы данных имеет собственный файл конфигурации базы данных.

У DB2 есть много параметров настройки и конфигурирования. Эти параметры разбиваются на две общих категории:

- “Параметры менеджера баз данных” на стр. 351
- “Параметры базы данных” на стр. 357.

Помимо описаний отдельных параметров, обратите внимание на следующие темы, в которых существенную роль играют параметры конфигурации:

- “Настройка параметров конфигурации” на стр. 350.
- “Подробности параметров для разных целей” на стр. 364 (у каждой функциональной области свой список параметров конфигурации).
- “Приложение A. Переменные реестра и среды DB2” на стр. 521.

  На вашей конкретной платформе могут быть свои переменные среды или реестра, связанные с производительностью, и их следует учитывать наряду со связанными с производительностью параметрами конфигурации.
- “Глава 8. Производительность работы” на стр. 253.
- “Глава 12. Измерение производительности” на стр. 335.

Следует просмотреть все сводки параметров в разделах Табл. 17 на стр. 353 и Табл. 19 на стр. 360, а затем сосредоточиться на изучении и настройке тех из них, которые обеспечивают наиболее существенный выигрыш в вашей рабочей среде.

Настройка параметров конфигурации

Вас может удовлетворить дисковое пространство и память, которые менеджер баз данных отводит, исходя из значений параметров по умолчанию. Однако возможны ситуации, когда значения по умолчанию не позволяют достичь максимальной производительности.

Поскольку значения по умолчанию ориентированы на компьютеры с относительно небольшой памятью, причем специально используемые как серверы баз данных, изменение параметров может потребоваться в среде, в которой есть:

- Большие базы данных
- Многочисленные соединения
- Требования высокой производительности для определенной прикладной программы
- Необычная загрузка или тип запроса или транзакции
- Другая конфигурация или применение компьютера.

Каждая среда обработки транзакций уникальна в тех или иных отношениях. Эти особенности могут серьезно сказаться на производительности менеджера баз данных, если использовать конфигурацию по умолчанию. По этой причине настоятельно рекомендуется настраивать конфигурацию для своей среды.

У разных типов прикладных программ и пользователей различные требования и ожидания для времени ответа. Диапазон прикладных программ может простираться от простых экранов ввода данных до огромных программ с десятками сложных операторов SQL, запрашивающих десятки таблиц в каждой единице работы. Например, требования к времени ответа могут существенно различаться для службы работы с клиентами по телефону и пакетной прикладной программы создания отчета.

Другие темы, которые могут быть полезны при тестовых измерениях прикладной программы для настройки параметров конфигурации:

- “Параметры менеджера баз данных” на стр. 351
- “Параметры базы данных” на стр. 357
- “Подробности параметров для разных целей” на стр. 364 (у каждой функциональной области свой список параметров конфигурации)
- “Глава 8. Производительность работы” на стр. 253
- “Глава 12. Измерение производительности” на стр. 335
- Описание элементов Монитора баз данных в руководстве *System Monitor Guide and Reference*.

Параметры менеджера баз данных

Параметры менеджера баз данных хранятся в файле с именем db2systm. Этот файл создается при создании экземпляра менеджера баз данных. В средах на основе UNIX этот файл находится в подкаталоге sqllib каталога экземпляра менеджера баз данных. В остальных средах этот файл по умолчанию создается в подкаталоге экземпляра каталога sqllib. Если задана переменная DB2INSTPROF, этот файл находится в подкаталоге instance каталога, заданного переменной DB2INSTPROF.

В среде многораздельных баз данных этот файл находится в совместно используемой файловой системе, так что все серверы разделов базы данных имеют доступ к одному и тому же файлу. Конфигурация менеджера баз данных на всех серверах разделов базы данных одна и та же.

Большинство параметров либо влияет на объем системных ресурсов, который будет отводиться одному экземпляру менеджера баз данных, либо задают настройку менеджера баз данных и разных подсистем связи, учитывающую особенности среды. Кроме того, существуют параметры, которые служат чисто информационным целям и не допускают изменения. Все эти параметры имеют глобальную применимость, независимо от всех отдельных баз данных, хранящихся в этом экземпляре менеджера баз данных.

Файл db2systm не допускает прямого редактирования. Его изменение и просмотр возможны лишь через предоставляемый интерфейс API или инструменты, вызывающие этот интерфейс.

Внимание: Отредактировав этот файл каким-либо иным методом, не предусмотренным в системе, вы можете привести систему в нерабочее состояние. **Мы настоятельно рекомендуем** не использовать для изменения этого файла никаких методов, кроме описанных в документации и поддерживаемых DB2.

Для сброса, изменения и просмотра параметров конфигурации менеджера баз данных вы можете использовать один из следующих методов:

- Использование Центра управления. В Центре управления есть записная книжка Конфигурировать экземпляр, которая позволяет задавать параметры конфигурации менеджера баз данных на клиенте или сервере. Кроме того, в Центре управления есть мастер по конфигурированию производительности, позволяющий изменять параметры конфигурации на сервере. Этот мастер генерирует значения на основе ваших ответов на набор вопросов, например, о рабочей нагрузке и типе транзакций, выполняемых на базе данных. Информацию об использовании этих интерфейсов смотрите в электронной справке Центра управления.

- Использование процессора командной строки. Быстрый и удобный способ изменения настройки - ввод команд. В руководстве *Command Reference* вы найдете дополнительную информацию о следующих командах:
  - GET DATABASE MANAGER CONFIGURATION (или GET DBM CFG)
  - UPDATE DATABASE MANAGER CONFIGURATION (или UPDATE DBM CFG)
  - RESET DATABASE MANAGER CONFIGURATION (или RESET DBM CFG).
- Использование интерфейсов прикладного программирования (API). Интерфейсы API удобно вызывать из прикладной программы. Дополнительную информацию смотрите в справочнике *Administrative API Reference*.
- Использование Ассистента конфигурирования клиента. При помощи Ассистента конфигурирования клиента можно задавать параметры конфигурации менеджера баз данных только на клиенте.

Чтобы новые значения параметров вступили в силу, нужно после изменения параметров остановить менеджер баз данных (`db2stop`) и затем перезапустить его (`db2start`). С точки зрения клиентов, изменения параметров конфигурации менеджера баз данных вступают в силу при следующем подключении клиента к серверу. Но хотя новые значения параметров еще не действуют, при просмотре параметров будут показаны последние изменения.

Примечание: Нет необходимости перезапускать менеджер баз данных, когда вы изменяете значение параметра *dft_monswitches*; этот параметр автоматически обновляется при изменении.

Сводка параметров конфигурации Менеджера баз данных

В следующей таблице перечисляются параметры из файла конфигурации менеджера баз данных для серверов баз данных. Изменяя параметры конфигурации менеджера баз данных, изучите детальную информацию о каждом параметре. В описание каждого параметра входят особенности операционной среды, включая значения параметра по умолчанию.

В следующей таблице в столбце “Влияние на производительность” приведены сведения о степени влияния каждого параметра на производительность системы. В этом столбце невозможно точно описать особенности всевозможных сред; эту информацию нужно рассматривать как обобщение.

- **Сильно** – указывает, что параметр существенно влияет на производительность. Значения таких параметров следует выбирать осознанно; в некоторых случаях это значит, что следует принять значение по умолчанию.
- **Средне** – указывает, что параметр до некоторой степени влияет на производительность. Внимание, которое при настройке стоит уделить этим параметрам, зависит от вашей конкретной среды и потребностей.

- **Слабо** – указывает, что параметр относительно редко или относительно слабо влияет на производительность.
- **Не влияет** – указывает, что параметр не оказывает прямого влияния на производительность. Эти параметры нет необходимости настраивать ради производительности, но они могут играть важную роль в других аспектах конфигурации системы, например, для поддержки возможности связи.

Таблица 17. Настраиваемые параметры конфигурации менеджера баз данных

Параметр	Влияние на производительность	Дополнительная информация
agentpri	Сильное	“Приоритет агентов (agentpri)” на стр. 423
agent_stack_sz	Слабое	“Размер стека агента (agent_stack_sz)” на стр. 390
aslheapsz	Сильное	“Размер кучи слоя поддержки программ (aslheapsz)” на стр. 395
audit_buf_sz	Сильное	“Размер буфера аудита (audit_buf_sz)” на стр. 404
authentication	Слабое	“Тип аутентификации (authentication)” на стр. 511
backbufsz	Среднее	“Размер буфера резервного копирования по умолчанию (backbufsz)” на стр. 373
catalog_noauth	Нет	“Разрешение каталогизации без полномочий (catalog_noauth)” на стр. 513
comm_bandwidth	Среднее	“Пропускная способность связи (comm_bandwidth)” на стр. 500
conn_elapse	Среднее	“Затраченное время соединения (conn_elapse)” на стр. 487
cpuspeed	Слабое (смотрите примечание)	“Скорость процессора (cpuspeed)” на стр. 501
datalinks	Слабое	“Поддержка связей данных (datalinks)” на стр. 464
dft_account_str	Нет	“Счет оплаты по умолчанию (dft_account_str)” на стр. 506
dft_client_adpt	Нет	“Номер адаптера клиента по умолчанию (dft_client_adpt)” на стр. 483
dft_client_comm	Нет	“Протокол связи клиента по умолчанию (dft_client_comm)” на стр. 482
dft_monswitches • *dft_mon_bufpool* • *dft_mon_lock* • *dft_mon_sort* • *dft_mon_stmt* • *dft_mon_table* • *dft_mon_uow*	Среднее	“Переключатели системного монитора баз данных по умолчанию (dft_monswitches)” на стр. 498

Таблица 17. Настраиваемые параметры конфигурации менеджера баз данных (продолжение)

Параметр	Влияние на производительность	Дополнительная информация
dftdbpath	Нет	“Путь баз данных по умолчанию (dftdbpath)” на стр. 513
diaglevel	Слабое	“Уровень захвата диагностических сообщений (diaglevel)” на стр. 495
diagpath	Нет	“Путь каталога данных диагностики (diagpath)” на стр. 496
dir_cache	Среднее	“Поддержка кэша каталогов (dir_cache)” на стр. 403
dir_obj_name	Нет	“Имя объекта в пространстве имен DCE (dir_obj_name)” на стр. 480
dir_path_name	Нет	“Каталог в пространстве имен DCE (dir_path_name)” на стр. 480
dir_type	Нет	“Тип служб каталога (dir_type)” на стр. 479
discover	Среднее	“Режим поиска (discover)” на стр. 484
discover_comm	Слабое	“Протоколы связи поиска (discover_comm)” на стр. 485
discover_inst	Слабое	“Экземпляр сервера Discover (discover_inst)” на стр. 486
dos_rqrioblk	Сильное	“Размер блока ввода-вывода реквестера DOS (dos_rqrioblk)” на стр. 399
drda_heap_sz	Слабое	“Параметр drda_heap_sz (размер кучи DRDA)” на стр. 388
fcm_num_anchors	Сильное	“Число привязок сообщений FCM (fcm_num_anchors)” на стр. 488
fcm_num_buffers	Сильное	“Число буферов FCM (fcm_num_buffers)” на стр. 488
fcm_num_connect	Сильное	“Число записей соединений FCM (fcm_num_connect)” на стр. 490
fcm_num_rqb	Сильное	“Число блоков требований FCM (fcm_num_rqb)” на стр. 490
federated	Среднее	“Поддержка систем баз данных объединения (federated)” на стр. 507
fileserver	Нет	“Имя файл-сервера IPX/SPX (fileserver)” на стр. 476
indexrec	Среднее	“Время пересоздания индекса (indexrec)” на стр. 448
initdari_jvm	Среднее	“Инициализировать процессы DARI в JVM (initdari_jvm)” на стр. 433
intra_parallel	Сильное	“Разрешение внутриразделького параллелизма (intra_parallel)” на стр. 494

Таблица 17. Настраиваемые параметры конфигурации менеджера баз данных (продолжение)

Параметр	Влияние на производительность	Дополнительная информация
ipx_socket	Нет	“Номер гнезда IPX/SPX (ipx_socket)” на стр. 478
java_heap_sz	Сильное	“Максимальный размер кучи интерпретатора Java (java_heap_sz)” на стр. 405
jdk11_path	Нет	“Путь установки Java Development Kit 1.1 (jdk11_path)” на стр. 507
keepdari	Среднее	“Индикатор сохранения процесса DARI (keepdari)” на стр. 431
maxagents	Среднее	“Максимальное число агентов (maxagents)” на стр. 425
maxcagents	Среднее	“Максимальное число одновременных агентов (maxcagents)” на стр. 426
max_connretries	Среднее	“Число попыток соединений с узлом (max_connretries)” на стр. 491
max_coordagents	Среднее	“Максимальное число координирующих агентов (max_coordagents)” на стр. 427
maxdari	Среднее	“Максимальное число процессов DARI (maxdari)” на стр. 432
max_logicagents	Среднее	“Максимальное число логических агентов (max_logicagents)” на стр. 428
max_querydegree	Сильное	“Максимальная степень параллелизма запросов (max_querydegree)” на стр. 493
max_time_diff	Среднее	“Максимальная разница времени между узлами (max_time_diff)” на стр. 492
maxtotfilop	Среднее	“Максимальное число открытых файлов (maxtotfilop)” на стр. 422
min_priv_mem	Среднее	“Минимальная принятая частная память (min_priv_mem)” на стр. 392
mon_heap_sz	Слабое	“Размер кучи монитора баз данных (mon_heap_sz)” на стр. 401
nname	Нет	“Имя рабочей станции NetBIOS (nname)” на стр. 474
notifylevel	Слабое	“Уровень оповещения (notifylevel)” на стр. 497
numdb	Слабое	“Максимальное число одновременно активных баз данных (numdb)” на стр. 502
num_initagents	Среднее	“Начальное число агентов в пуле (num_initagents)” на стр. 430

Таблица 17. Настраиваемые параметры конфигурации менеджера баз данных (продолжение)

Параметр	Влияние на производительность	Дополнительная информация
num_initdaris	Среднее	“Начальное число изолированных процессов DARI в пуле (num_initdaris)” на стр. 434
num_poolagents	Сильное	“Размер пула агентов (num_poolagents)” на стр. 428
objectname	Нет	“Имя объекта сервера DB2 IPX/SPX (objectname)” на стр. 477
priv_mem_thresh	Среднее	“Порог собственной памяти (priv_mem_thresh)” на стр. 392
query_heap_sz	Среднее	“Размер кучи запросов (query_heap_sz)” на стр. 387
restbufsz	Среднее	“Размер буфера восстановления по умолчанию (restbufsz)” на стр. 374
resync_interval	Нет	“Интервал ресинхронизации транзакций (resync_interval)” на стр. 454
route_obj_name	Нет	“Имя объекта информации маршрутизации (route_obj_name)” на стр. 481
rqrioblk	Сильное	“Размер блока ввода-вывода клиента (rqrioblk)” на стр. 398
sheapthres	Сильное	“Порог кучи сортировки (sheapthres)” на стр. 382
spm_log_file_sz	Слабое	“Размер файлов журнала менеджера точек синхронизации (spm_log_file_sz)” на стр. 456
spm_log_path	Среднее	“Путь файлов журнала менеджера точек синхронизации (spm_log_path)” на стр. 455
spm_max_resync	Слабое	“Предельное число агентов ресинхронизации менеджера точек синхронизации (spm_max_resync)” на стр. 457
spm_name	Нет	“Имя менеджера точек синхронизации (spm_name)” на стр. 456
ss_logon	Нет	“LOGON, требуемый для DB2START/DB2STOP (ss_logon)” на стр. 515
start_stop_time	Слабое	“Срок ожидания запуска и остановки (start_stop_time)” на стр. 492
svcename	Нет	“Имя службы TCP/IP (svcename)” на стр. 475
sysadm_group	Нет	“Имя группы с правами администратора системы (sysadm_group)” на стр. 508
sysctrl_group	Нет	“Имя группы с правами управления системой (sysctrl_group)” на стр. 509
sysmaint_group	Нет	“Имя группы с правами обслуживания системы (sysmaint_group)” на стр. 510

Таблица 17. Настраиваемые параметры конфигурации менеджера баз данных (продолжение)

Параметр	Влияние на производительность	Дополнительная информация
tm_database	Нет	“Имя базы данных менеджера транзакций (tm_database)” на стр. 454
tp_mon_name	Нет	“Имя монитора процессора транзакций (tp_mon_name)” на стр. 503
tpname	Нет	“Имя программы транзакций APPC (tpname)” на стр. 476
trust_allclnts	Нет	“Доверять всем клиентам (trust_allclnts)” на стр. 515
trust_clntauth	Нет	“Аутентификация доверенных клиентов (trust_clntauth)” на стр. 516
udf_mem_sz	Слабое	“Размер совместной памяти пользовательских функций (udf_mem_sz)” на стр. 389
Примечание: Параметр *cpuspeed* может существенно влиять на производительность, но вам не следует отказываться от значения по умолчанию, за исключением весьма специфических обстоятельств, приведенных в описании параметра.		

Таблица 18. Информационные параметры конфигурации менеджера баз данных

Параметр	Дополнительная информация
nodetype	“Тип узла компьютера (nodetype)” на стр. 505
release	“Уровень выпуска файла конфигурации (release)” на стр. 459

Параметры базы данных

Параметры отдельной базы данных хранятся в файле конфигурации с именем SQLDBCON. Этот файл хранится вместе с другими файлами управления базы данных в каталоге SQLnnnnn, где nnnnn - это номер, присваиваемый при создании базы данных. (Дополнительную информацию о расположении этого каталога смотрите в разделе “Физические каталоги баз данных” руководства *Administration Guide: Planning*.) У каждой базы данных есть собственный файл конфигурации, и большинство параметров из этого файла задают объем ресурсов, выделяемых базе данных. Кроме того, файл содержит описательную информацию, а также флаги состояния базы данных.

Файл SQLDBCON не подлежит прямому редактированию, и изменение и просмотр этого файла возможны лишь через предоставляемый интерфейс API или инструменты, вызывающие этот интерфейс.

Внимание: Отредактировав этот файл каким-либо иным методом, не предусмотренным в системе, вы можете привести систему в нерабочее состояние. **Мы настоятельно рекомендуем** не использовать для изменения этого файла никаких методов, кроме описанных в документации и поддерживаемых DB2.

Для сброса, изменения и просмотра параметров конфигурации базы данных можно использовать один из следующих трех методов:

- Использование Центра управления. В Центре управления есть записная книжка Конфигурировать базу данных, а также мастер по конфигурированию производительности, которые позволят изменять параметры конфигурации. Этот мастер генерирует значения на основе ваших ответов на набор вопросов, например, о рабочей нагрузке и типе транзакций, выполняемых на базе данных. Информацию об использовании этих интерфейсов смотрите в электронной справке Центра управления.

  В среде многораздельных баз данных файл SQLDBCON есть для каждого раздела базы данных. Записная книжка Конфигурировать базу данных Центра управления изменит значение во всех разделах, если запустить ее из объекта базы данных в дереве Центра управления. Если запустить записную книжку из объекта раздела базы данных, она будет изменять значения только для этого раздела. (Однако мы рекомендуем, чтобы параметры конфигурации имели одни и те же значения на всех разделах.)

  Примечание: Мастер по конфигурации производительности в среде многораздельных баз данных недоступен.

- Использование процессора командной строки. Быстрый и удобный способ изменения настройки - ввод команд. В руководстве *Command Reference* вы найдете дополнительную информацию о следующих командах:
  - GET DATABASE CONFIGURATION (или GET DB CFG)
  - UPDATE DATABASE CONFIGURATION (или UPDATE DB CFG)
  - RESET DATABASE CONFIGURATION (или RESET DB CFG)
- Использование интерфейсов прикладного программирования (API). Интерфейсы API удобно вызывать из программы на языке хоста. Дополнительную информацию смотрите в справочнике *Administrative API Reference*.

Для большинства изменяемых параметров новые значения не вступают в силу, пока остаются подключенные к базе данных программы. Все прикладные программы должны сначала отключиться от базы данных. (Если база данных была активирована, ее нужно деактивировать и активировать снова.) Тогда при первом новом подключении к базе данных изменения вступят в силу. Следует отметить, что вступление в силу изменений некоторых параметров, таких как *newlogpath*, *logfilsiz* и *logprimary*, может занять достаточно длительное время, поскольку требует служебных операций, связанных с отведением пространства.

Возможно, вы захотите создать тестовое подключение к базе данных и произвести изменение во время тестового подключения, чтобы служебные операции не помешали работе остальных пользователей. Если такие помехи для вас нежелательны, можно также использовать команду ACTIVATE DATABASE, как описано в руководстве *Command Reference*.

Примечание: Нет необходимости отключаться от базы данных, когда вы изменяете значение параметра *mincommit*; этот параметр автоматически обновляется при изменении.

Изменение некоторых параметров конфигурации базы данных может повлиять на план доступа, выбираемый программой оптимизации SQL. Эти параметры баз данных описаны в разделе “Параметры конфигурации, влияющие на оптимизацию запросов” на стр. 95. После изменения любого из описанных там параметров разумно будет повторно связать прикладные программы, чтобы обеспечить использование наилучшего плана доступа для операторов SQL. Дополнительную информацию о команде BIND смотрите в руководстве *Command Reference*.

Хотя новые значения параметров еще не действуют, при просмотре параметров будут показаны последние изменения.

Примечание: Ряд параметров конфигурации баз данных (например, *userexit*) допускает только два значения, которые в справках и других книгах DB2 названы “Yes” и “No” (Да и Нет) или “On” и “Off” (Вкл и Выкл). На всякий случай поясним, что “Yes” эквивалентно “On”, а “No” эквивалентно “Off”.

Сводка параметров конфигурации баз данных

В следующей таблице перечисляются параметры из файла конфигурации баз данных. Изменяя параметры конфигурации баз данных, изучите детальную информацию о параметре.

В следующей таблице в столбце “Влияние на производительность” приведены сведения о степени влияния каждого параметра на производительность системы. В этом столбце невозможно точно описать особенности всевозможных сред; эту информацию нужно рассматривать как обобщение.

- **Сильно** – указывает, что параметр существенно влияет на производительность. Значения таких параметров следует выбирать осознанно; в некоторых случаях это значит, что следует принять значение по умолчанию.
- **Средне** – указывает, что параметр до некоторой степени влияет на производительность. Внимание, которое при настройке стоит уделить этим параметрам, зависит от вашей конкретной среды и потребностей.

- **Слабо** – указывает, что параметр относительно редко или относительно слабо влияет на производительность.
- **Не влияет** – указывает, что параметр не оказывает прямого влияния на производительность. Эти параметры нет необходимости настраивать ради производительности, но они могут играть важную роль в других аспектах конфигурации системы, например, для поддержки возможности связи.

Таблица 19. Конфигурируемые параметры конфигурации баз данных

Параметр	Влияние на производительность	Дополнительная информация
app_ctl_heap_sz	Среднее	“Размер кучи управления прикладной программы (app_ctl_heap_sz)” на стр. 379
applheapsz	Среднее	“Размер кучи прикладных программ (applheapsz)” на стр. 385
autorestart	Слабое	“Разрешение автоматического перезапуска (autorestart)” на стр. 447
avg_appls	Сильное	“Среднее число активных программ (avg_appls)” на стр. 420
buffpage	Сильное (если активен)	“Размер пула буферов (buffpage)” на стр. 366
catalogcache_sz	Среднее	“Размер кэша каталога (catalogcache_sz)” на стр. 370
chngpgs_thresh	Сильное	“Порог числа измененных страниц (chngpgs_thresh)” на стр. 411
copyprotect	Нет	“Включение защиты от копирования (copyprotect)” на стр. 462
dbheap	Среднее	“Куча базы данных (dbheap)” на стр. 369
dft_degree	Сильное	“Степень параллелизма по умолчанию (dft_degree)” на стр. 469
dft_extent_sz	Среднее	“Размер экстента по умолчанию для табличных пространств (dft_extent_sz)” на стр. 417
dft_loadrec_ses	Среднее	“Число сеансов по умолчанию при восстановлении загрузки (dft_loadrec_ses)” на стр. 449
dft_prefetch_sz	Среднее	“Размер предварительной выборки по умолчанию (dft_prefetch_sz)” на стр. 415
dft_queryopt	Среднее	“Класс оптимизации запросов по умолчанию (dft_queryopt)” на стр. 470
dft_refresh_age	Среднее	“Срок обновления по умолчанию (dft_refresh_age)” на стр. 471
dft_sqlmathwarn	Нет	“Продолжение при арифметических исключительных ситуациях (dft_sqlmathwarn)” на стр. 468

Таблица 19. Конфигурируемые параметры конфигурации баз данных (продолжение)

Параметр	Влияние на производительность	Дополнительная информация
dir_obj_name	Нет	“Имя объекта в пространстве имен DCE (dir_obj_name)” на стр. 480
discover_db	Среднее	“Поиск баз данных (discover_db)” на стр. 484
dlchktime	Среднее	“Интервал проверки тупиковых ситуаций (dlchktime)” на стр. 406
dl_expint	Нет	“Срок действия маркера связей данных (dl_expint)” на стр. 462
dl_num_copies	Нет	“Число копий связей данных (dl_num_copies)” на стр. 463
dl_time_drop	Нет	“Время после отбрасывания связей данных (dl_time_drop)” на стр. 463
dl_token	Слабое	“Алгоритм маркера связей данных (dl_token)” на стр. 464
dl_upper	Нет	“Верхний регистр в маркере связей данных (dl_upper)” на стр. 464
dyn_query_mgmt	Слабое	“Управление запросами динамического SQL (dyn_query_mgmt)” на стр. 458
estore_seg_sz	Среднее	“Размер сегмента расширения памяти (estore_seg_sz)” на стр. 417
indexrec	Среднее	“Время пересоздания индекса (indexrec)” на стр. 448
indexsort	Слабое (смотрите примечание на странице 363)	“Флаг сортировки индекса (indexsort)” на стр. 414
locklist	Сильное, если влияет на расширение блокировок	“Максимальная память для списка блокировок (locklist)” на стр. 375
locktimeout	Среднее	“Срок ожидания блокировки (locktimeout)” на стр. 409
logbufsz	Сильное	“Размер буфера журнала (logbufsz)” на стр. 371
logfilsiz	Среднее	“Размер файлов журнала (logfilsiz)” на стр. 435
logprimary	Среднее	“Число первичных файлов журнала (logprimary)” на стр. 436
logretain	Слабое	“Разрешение хранения журнала (logretain)” на стр. 445
logsecond	Среднее	“Число вторичных файлов журнала (logsecond)” на стр. 438
maxappls	Среднее	“Максимальное число активных прикладных программ (maxappls)” на стр. 419

Таблица 19. Конфигурируемые параметры конфигурации баз данных (продолжение)

Параметр	Влияние на производительность	Дополнительная информация
maxfilop	Среднее	“Максимальное число файлов баз данных, открытых для программы (maxfilop)” на стр. 421
maxlocks	Сильное, если влияет на расширение блокировок	“Максимальный процент списка блокировок перед расширением (maxlocks)” на стр. 407
mincommit	Сильное	“Число принятий для группировки (mincommit)” на стр. 442
min_dec_div_3	Сильное	“Параметр min_dec_div_3” на стр. 396
newlogpath	Слабое	“Изменить путь журнала базы данных (newlogpath)” на стр. 439
num_db_backups	Нет	“Число резервных копий базы данных (num_db_backups)” на стр. 450
num_estore_segs	Среднее	“Число сегментов расширения памяти (num_estore_segs)” на стр. 418
num_freqvalues	Слабое	“Число сохраняемых частых значений (num_freqvalues)” на стр. 471
num_iocleaners	Сильное	“Число асинхронных чистильщиков страниц (num_iocleaners)” на стр. 412
num_ioservers	Сильное	“Число серверов ввода/вывода (num_ioservers)” на стр. 413
num_quantiles	Слабое	“Число квантилей для столбцов (num_quantiles)” на стр. 472
pckcachesz	Сильное	“Размер кэша пакета (pckcachesz)” на стр. 378
rec_his_retentn	Нет	“Период хранения хронологии восстановления (rec_his_retentn)” на стр. 450
seqdetect	Сильное	“Флаг обнаружения последовательного чтения (seqdetect)” на стр. 415
softmax	Среднее	“Диапазон восстановления и интервал мягких контрольных точек (softmax)” на стр. 443
sortheap	Сильное	“Размер кучи сортировки (sortheap)” на стр. 381
stat_heap_sz	Слабое	“Размер кучи статистики (stat_heap_sz)” на стр. 386
stmtheap	Среднее	“Размер кучи операторов (stmtheap)” на стр. 385
trackmod	Слабое	“Разрешить отслеживание измененных страниц (trackmod)” на стр. 451
tsm_mgmtclass	Нет	“Класс управления Tivoli Storage Manager (tsm_mgmtclass)” на стр. 452
tsm_nodename	Нет	“Имя узла Tivoli Storage Manager (tsm_nodename)” на стр. 452

Таблица 19. Конфигурируемые параметры конфигурации баз данных (продолжение)

Параметр	Влияние на производительность	Дополнительная информация
tsm_owner	Нет	“Имя владельца Tivoli Storage Manager (tsm_owner)” на стр. 453
tsm_password	Нет	“Пароль Tivoli Storage Manager (tsm_password)” на стр. 452
userexit	Слабое	“Разрешение обработчика пользователя (userexit)” на стр. 446
util_heap_sz	Слабое	“Размер кучи утилит (util_heap_sz)” на стр. 372
Примечание: Изменение значения по умолчанию параметра *indexsort* может оказать отрицательное влияние на производительность при создании индексов. Для этого параметра лучше по возможности оставить значение по умолчанию.		

Таблица 20. Информационные параметры конфигурации баз данных

Параметр	Дополнительная информация
backup_pending	“Индикатор отложенного резервного копирования (backup_pending)” на стр. 465
codepage	“Кодовая страница для базы данных (codepage)” на стр. 461
codeset	“Кодовый набор для базы данных (codeset)” на стр. 460
collate_info	“Информация упорядочения (collate_info)” на стр. 461
country	“Код страны для базы данных” на стр. 460
database_consistent	“Согласованность базы данных (database_consistent)” на стр. 465
database_level	“Уровень выпуска базы данных (database_level)” на стр. 459
log_retain_status	“Индикатор состояния сохранения журналов (log_retain_status)” на стр. 466
loghead	“Первый активный файл журнала (loghead)” на стр. 441
путь_журнала	“Положение файлов журналов (logpath)” на стр. 441
multipage_alloc	“Многостраничное размещение файлов разрешено (multipage_alloc)” на стр. 467
numsegs	“Число контейнеров SMS по умолчанию (numsegs)” на стр. 416
release	“Уровень выпуска файла конфигурации (release)” на стр. 459

Таблица 20. Информационные параметры конфигурации баз данных (продолжение)

Параметр	Дополнительная информация
restore_pending	“Отложенное восстановление (restore_pending)” на стр. 467
rollfwd_pending	“Индикатор отложенного повтора транзакций (rollfwd_pending)” на стр. 466
territory	“Территория для базы данных (territory)” на стр. 460
user_exit_status	“Индикатор состояния обработчика пользователя (user_exit_status)” на стр. 466

Подробности параметров для разных целей

В следующих разделах содержатся дополнительные детали, объясняющие работу и настройку различных параметров конфигурации. В этом описании отдельные параметры сгруппированы в зависимости от их назначения:

- “Управление мощностью” на стр. 365
- “Ведение журнала и восстановление” на стр. 434
- “Управление базами данных” на стр. 458
- “Связь” на стр. 474
- “Многораздельная база данных” на стр. 487
- “Управление экземпляром” на стр. 495.

Для каждого параметра предлагается следующая информация:

Тип конфигурации
Указывает, в каком файле конфигурации содержится значение параметра:

- Менеджер баз данных (параметр влияет на экземпляр менеджера баз данных и все определенные в нем базы данных)
- База данных (параметр влияет на конкретную базу данных)

Тип параметра
Указывает, можно ли изменять значение параметра:

- *Конфигурируемый*

  Существует диапазон допустимых значений, и, возможно, нужна настройка параметра с учетом сведений о прикладных программах и/или тестовых измерениях, которыми располагает администратор баз данных.
- *Информационный*

Эти параметры изменяются только самим менеджером баз данных и содержат такую информацию, как выпуск DB2, в котором база данных создана, или сообщение о том, что затребованное резервное копирование отложено.

Управление мощностью

Есть ряд параметров конфигурации, как на уровне базы данных, так и на уровне менеджера баз данных, которые влияют на производительность системы. Эти параметры делятся на следующие группы:

- "Совместная память баз данных"
- "Совместная память прикладных программ" на стр. 379
- "Собственная память агента" на стр. 381
- "Память связи агента с прикладной программой" на стр. 394
- "Память экземпляра менеджера баз данных" на стр. 401
- "Блокировки" на стр. 406
- "Ввод/вывод и хранение" на стр. 410
- "Агенты" на стр. 418
- "Хранимые процедуры (DARI)" на стр. 430.

Начальные сведения об управлении памятью DB2 смотрите в разделе "Как DB2 использует память" на стр. 253.

Совместная память баз данных

Следующие параметры влияют на глобальную память баз данных, отводимую на вашей системе:

- "Размер пула буферов (buffpage)" на стр. 366.
- "Куча базы данных (dbheap)" на стр. 369.
- "Размер кэша каталога (catalogcache_sz)" на стр. 370.
- "Размер буфера журнала (logbufsz)" на стр. 371.
- "Размер кучи утилит (util_heap_sz)" на стр. 372.
- "Размер буфера резервного копирования по умолчанию (backbufsz)" на стр. 373.
- "Размер буфера восстановления по умолчанию (restbufsz)" на стр. 374.
- "Максимальная память для списка блокировок (locklist)" на стр. 375.
- "Размер кэша пакета (pckcachesz)" на стр. 378.

Информацию о том, как глобальная память баз данных связана с остальной памятью, отводимой менеджером баз данных, смотрите в разделе "Как DB2 использует память" на стр. 253.

Размер пула буферов (buffpage)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

32-битные платформы UNIX
1000 [2 – 524288]

64-битные платформы UNIX
1000 [2 – 2147483647]

OS/2 и Windows NT
250 [2 – 524288]

Единица измерения Страницы

Когда размещается Когда первая программа соединяется с базой данных

Когда освобождается Когда последняя программа отсоединяется от базы данных

Параметры, связанные с данным

- “Порог числа измененных страниц (chngpgs_thresh)” на стр. 411
- “Куча базы данных (dbheap)” на стр. 369
- “Число асинхронных чистильщиков страниц (num_iocleaners)” на стр. 412

У каждой базы данных есть как минимум один пул буферов (IBMDEFAULTBP, который создается при создании базы данных), а возможно, и больше. Все пулы буферов размещаются в глобальной памяти, доступной всем использующим базу данных программам. Эта память выделяется на компьютере, где расположена база данных. Если пулы буферов достаточно велики, чтобы хранить требуемые из памяти данные, дисковый ввод/вывод сокращается. Наоборот, если пулы буферов недостаточно велики, общая производительность базы данных может сильно ухудшиться, и ввод/вывод для менеджера баз данных может оказаться узким местом из-за большого количества дисковых операций ввода/вывода для обработки данных, требуемых вашей программой.

Если оператор CREATE BUFFERPOOL или ALTER BUFFERPOOL запускается со значением NPAGES -1, размером пула буферов управляет параметр *buffpage*; иначе параметр *buffpage* игнорируется, и пул буферов создается с числом страниц, задаваемых параметром NPAGES.

Чтобы определить, активен параметр *buffpage* для пула буферов или нет, выполните операцию:

```
SELECT * from SYSCAT.BUFFERPOOLS.
```

Параметр *buffpage* используется для каждого пула буферов, заданного с опцией NPAGES -1.

Надо найти компромисс между размером пула буферов и размерами памяти, выделяемой для других пользователей системы. Требования к памяти серверов базы данных важны на серверах с большим числом пользователей и высокой интенсивностью транзакций, так как серверы баз данных и файл-серверы или серверы связи часто разделены и находятся на разных компьютерах.

Если в запросах используются псевдонимы, попробуйте увеличить размер пула буферов, когда:
- Оптимизатор решил, что все операции надо выполнять локально. При обработке запроса оптимизатор, как правило, переносит операции на источники данных, где это возможно. Например, на источнике данных обычно выполняется оператор GROUP BY. Однако возможно, что материализация таблицы в DB2 и локальное выполнение операции окажутся выгоднее. Такая ситуация возможна, если рабочая станция сервера DB2 мощнее рабочей станции источника данных.
- Операции сортировки должны выполняться локально. Запросы, содержащие псевдонимы, сортируются в соответствии с последовательностью упорядочения DB2. Если в источнике данных используется другая последовательность упорядочения, все операции сортировки выполняются локально.

Все пулы буферов выделяются при первом соединении программы с базой данных, или когда база данных явно активирована. По мере того, как программа требует данные из базы данных, страницы, содержащие эти данные, передаются с диска в один из пулов буферов. (Обратите внимание на то, что данные базы данных хранятся в страницах таблиц на диске.) Страница записывается обратно на диск, если она изменена и происходит одно из следующих событий:
- Все программы отсоединяются от базы данных
- База данных явно деактивируется
- База данных стабилизируется (то есть происходит принятие для всех соединенных программ)
- Пространство данной таблицы требуется для другой страницы, которую необходимо считать в пул буферов
- Программа очистки страниц (*num_iocleaners*) запущена и активирована менеджером баз данных.

Рекомендации:
- Для создания и изменения пулов буферов и их размеров вместо параметра конфигурации *buffpage* можно использовать операторы CREATE BUFFERPOOL и ALTER BUFFERPOOL SQL.

- Размер пула буферов используется оптимизатором при определении планов доступа. После изменения этого параметра, возможно, следует выполнить повторное связывание прикладных программ (используя команду REBIND PACKAGE).
- Так размеры всех пулов буферов могут сильно влиять на производительность, для избежания чрезмерной подкачки страниц следует учитывать следующие факторы:
  - Объем установленной на компьютере памяти.
  - Память, требуемая другими программами, работающими одновременно с менеджером баз данных на одном компьютере.

  Подкачка страниц происходит, когда для хранения страницы, к которой обратились, не хватает памяти. Как результат, данная страница записывается ("подкачивается") во временную дисковую память, чтобы освободилось место для другой страницы. Когда эта страница во временной дисковой памяти снова потребуется, она "подкачивается обратно" в память.
- При желании для пулов буферов базы данных можно выделить до 75% памяти компьютера, если:
  - Пользователей много
  - Компьютер используется только как сервер баз данных
  - Большинство обращений производятся к одним и тем же станицам данных и страницам индекса
  - На компьютере всего одна база данных.
- Для каждой выделенной страницы пула буферов определенное пространство используется в куче базы данных для внутренних управляющих структур.

  При увеличении общего размера пула буферов (или пулов буферов) может также потребоваться увеличить и размер кучи базы данных *dbheap*.
- В среде объединения если последовательность упорядочения источника данных совпадает с последовательностью упорядочения DB2, убедитесь, что это отражено в опции сервера collating_sequence. Опция collating_sequence должна иметь значение "Y". Вы можете создавать базы данных объединения с определенной последовательностью упорядочения, которая совпадает с последовательностью упорядочения источника данных. Такой подход может повысить производительность, если на всех источниках данных используется одна и та же последовательность упорядочения, или если большая часть функций столбцов применяется к источникам данных, использующих одинаковую последовательность упорядочения. Если на источнике данных последовательность упорядочения отличается от последовательности упорядочения DB2, большинство операций, зависящих от последовательности упорядочения DB2, нельзя выполнять удаленно на источнике данных. Дополнительную информацию об этой опции сервера смотрите в разделе *Administration Guide: Implementation*.

Для вычисления коэффициента попадания в пул буферов можно воспользоваться системным монитором баз данных, который помогает настроить пулы буферов. Смотрите книгу *System Monitor Guide and Reference*.

Куча базы данных (dbheap)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

UNIX 1200 [32 – 524288]

OS/2 и Windows NT, Сервер баз данных с локальными и удаленными клиентами
600 [32 – 524288]

OS/2 и Windows NT, Сервер баз данных с локальными клиентами
300 [32 – 524288]

Единица измерения Страницы (4 Кбайта)

Когда размещается При первом соединении с базой данных

Когда освобождается Когда последняя программа отсоединяется от базы данных

Параметры, связанные с данным

- “Размер кэша каталога (catalogcache_sz)” на стр. 370
- “Размер буфера журнала (logbufsz)” на стр. 371

У каждой базы данных есть одна куча базы данных и менеджер баз данных использует ее для всех прикладных программ, соединенных с этой базой данных. Куча базы данных содержит информацию управляющего блока для таблиц, индексов, табличных пространств и пулов буферов. Она содержит также пространство для буфера журнала (*logbufsz*) и кэша каталога (*catalogcache_sz*). Поэтому размер кучи будет зависеть от числа управляющих блоков, находящихся в куче в данный момент. Информация управляющего блока хранится в куче до тех пор, пока все прикладные программы не отсоединятся от базы данных.

При первом соединении с базой данных выделяется минимальный объем кучи, необходимый для начала работы менеджера баз данных. При необходимости размер этой области данных расширяется вплоть до максимального значения, заданного параметром *dbheap*.

Рекомендация: Нужно увеличить это значение, если прикладная программа получает код ошибки, указывающий, что для обработки оператора не хватает памяти в куче базы данных.

Узнать максимальный объем памяти, который был использован для кучи базы данных, можно с помощью системного монитора баз данных. Дополнительную информацию смотрите в описании элемента монитора *db_heap_top (максимальная выделенная куча базы данных)* в руководстве *System Monitor Guide and Reference*.

При задании этого параметра нужно учитывать:

- Значение параметра *logbufsz*, так как буфер журнала выделяется из кучи базы данных.
- Значение параметра *catalogcache_sz*, так как кэш каталога выделяется из кучи базы данных.

Размер кэша каталога (catalogcache_sz)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

UNIX 64 [1 – 60000]

OS/2 и Windows NT, Сервер баз данных с локальными и удаленными клиентами 32 [1 – 60000]

OS/2 и Windows NT, Сервер баз данных с локальными клиентами 16 [1 – 60000]

Единица измерения Страницы (4 Кбайта)

Параметры, связанные с данным

- “Куча базы данных (dbheap)” на стр. 369
- “Размер буфера журнала (logbufsz)” на стр. 371
- “Размер кучи управления прикладной программы (app_ctl_heap_sz)” на стр. 379

Этот параметр задает максимальное пространство, которое кэш каталога может использовать из кучи базы данных (*dbheap*). Кэш каталога используется для хранения информации дескриптора таблиц, которая требуется при обращении к таблице, производной таблице или алиасу во время компиляции оператора SQL.

С помощью этого кэша можно улучшить производительность при связывании операторов SQL (включая динамический SQL), если в предыдущих операторах были указаны те же самые таблицы, производные таблицы или алиасы. Информация дескриптора для объявленных временных таблиц хранится не в кэше каталога, а в куче управления программы.

Запуск любых операторов DDL для таблицы приведет к удалению записи об этой таблице из кэша каталога. В прочих случаях запись таблицы сохраняется в кэше до тех пор, пока не понадобится место для другой таблицы; она не будет удалена из кэша, пока не завершено выполнение любых единиц работы, обращающихся к этой таблице.

Рекомендация: Начните со значения по умолчанию и настройте его с помощью системного монитора базы данных.

Посмотрите в книге *System Monitor Guide and Reference* сведения о следующих элементах монитора:
- *cat_cache_lookups* (просмотры кэша каталога)
- *cat_cache_inserts* (вставки кэша каталога)
- *cat_cache_overflows* (переполнения кэша каталога)
- *cat_cache_heap_full* (полная куча кэша каталога)

Эти элементы системного монитора баз данных могут помочь определить, следует ли вам настраивать данный параметр конфигурации. При настройке значение этого параметра следует увеличивать постепенно, например, на две страницы за один раз.

Как правило, в кэше требуется больше места, если единица работы содержит несколько динамических операторов SQL, или при когда связываются пакеты, содержащие много статических операторов SQL.

Задавая размер кэша каталога, следует учесть также размер файлов журнала (*logbufsz*), так как и *catalogcache_sz*, и *logbufsz* выделяются из кучи базы данных (*dbheap*).

Размер буфера журнала (logbufsz)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

32-битные платформы UNIX
8 [4 – 4096]

64-битные платформы UNIX
8 [4 – 65535]

OS/2 и Windows NT
8 [4 – 4096]

Единица измерения Страницы (4 Кбайта)

Параметры, связанные с данным

- "Размер кэша каталога (catalogcache_sz)" на стр. 370
- "Куча базы данных (dbheap)" на стр. 369
- "Число принятий для группировки (mincommit)" на стр. 442

Этот параметр задает часть кучи базы данных (определяемую параметром *dbheap*), используемую как буфер для записей журнала перед их помещением на диск. Записи журнала записываются на диск в следующих ситуациях:
- Принятие транзакции или группы транзакций, определяемой параметром конфигурации *mincommit*
- Заполнение буфера журнала
- Некоторое другое внутреннее событие менеджера баз данных.

Значение этого параметра не должно превышать значения параметра *dbheap*. Буферизация записей журнала приводит к увеличению эффективности ввода/вывода файлов журнала, поскольку записи журнала записываются на диск реже и за один раз на диск записывается больше записей.

Рекомендация: Увеличьте размер буфера, если с диска, отведенного для записи журналов, идет интенсивное чтение или этот диск активно используется. Увеличивая значение этого параметра, подумайте, не стоит ли увеличить также значение параметра *dbheap*, поскольку буфер журнала выделяется из области памяти, размер которой задается параметром *dbheap*.

При помощи монитора баз данных можно определить, сколько пространства буферов журнала используется для определенной транзакции (или единицы работы).

Дополнительную информации смотрите в описании элемента монитора *log_space_used* (пространство журнала, используемой единицей работы) в руководстве *System Monitor Guide and Reference*.

При задании размера буфера журнала рассмотрите также размер кэша каталога (*catalogcache_sz*), поскольку и *logbufsz_sz*, и *catalogcache_sz* выделяются из кучи базы данных (*dbheap*).

Размер кучи утилит (util_heap_sz)

Тип конфигурации	База данных
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	5000 [16 – 524288]
Единица измерения	Страницы (4 Кбайта)
Когда размещается	Когда требуется утилитам менеджера баз данных

Когда освобождается Когда утилите больше не нужна память

Параметры, связанные с данным

- "Размер буфера резервного копирования по умолчанию (backbufsz)"
- "Размер буфера восстановления по умолчанию (restbufsz)" на стр. 374

Этот параметр задает максимальный объем памяти, который может быть использован одновременно утилитами BACKUP, RESTORE и LOAD (включая загрузку копии).

Рекомендация: Используйте значение по умолчанию; если же вашим утилитам не хватает памяти, увеличьте его. Если в вашей системе мало памяти, вы можете захотите уменьшить значение этого параметра, чтобы отвести утилитам меньше памяти. Если задать слишком малое значение, возможно, утилиты не смогут работать одновременно. Надо задать достаточное значение для размещения всех буферов, которые вы хотите задать для одновременно работающих утилит.

Размер буфера резервного копирования по умолчанию (backbufsz)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 1024 [8 – 16384]

Единица измерения Страницы (4 Кбайта)

Когда размещается Когда вызывается утилита резервного копирования

Когда освобождается Когда утилита резервного копирования завершает работу

Параметры, связанные с данным

- "Размер буфера восстановления по умолчанию (restbufsz)" на стр. 374
- "Размер кучи утилит (util_heap_sz)" на стр. 372

Этот параметр задает размер буфера, используемый для резервного копирования базы данных, если при вызове программы резервного копирования размер буфера не задан явно. Дополнительную информацию об утилите резервного копирования смотрите в справочнике *Command Reference*.

При резервном копировании базы данные сначала записываются во внутренний буфер. Когда буфер заполнен, данные из него переписываются на носитель.

Настройка размера буфера может улучшить производительность утилиты резервного копирования, а также минимизировать ее влияние на производительность других выполняемых одновременно операций базы данных.

Размер буфера восстановления по умолчанию (restbufsz)

Тип конфигурации	Менеджер баз данных
Применяется к	• Сервер баз данных с локальными и удаленными клиентами • Сервер баз данных с локальными клиентами • Сервер многораздельных баз данных с локальными и удаленными клиентами • Сервер сателлитных баз данных с удаленными клиентами
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	1024 [8 – 16384]
Единица измерения	Страницы (4 Кбайта)
Когда размещается	При вызове утилиты восстановления
Когда освобождается	Когда утилита восстановления завершает работу
Параметры, связанные с данным	• “Размер буфера резервного копирования по умолчанию (backbufsz)” на стр. 373 • “Размер кучи утилит (util_heap_sz)” на стр. 372

Этот параметр задает размер буфера, используемый для восстановления базы данных, если размер буфера не задан явно при вызове утилиты восстановления базы данных. Дополнительную информацию об утилите восстановления смотрите в справочнике *Command Reference*.

При восстановлении базы данные из носителя копии сначала записываются во внутренний буфер. Когда буфер заполнен, данные из него переписываются в место назначения.

Настройка размера буфера может улучшить производительность утилиты восстановления, а также минимизировать ее влияние на производительность других выполняемых одновременно операций базы данных.

Максимальная память для списка блокировок (locklist)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

UNIX 100 [4 – 524288]

OS/2 и NT Сервер баз данных с локальными и удаленными клиентами 50 [4 – 524288]

OS/2 и NT Сервер баз данных с локальными клиентами 25 [4 – 60000]

Единица измерения Страницы (4 Кбайта)

Когда размещается Когда первая программа соединяется с базой данных

Когда освобождается Когда последняя программа отсоединяется от базы данных

Параметры, связанные с данным

- “Максимальный процент списка блокировок перед расширением (maxlocks)” на стр. 407
- “Максимальное число активных прикладных программ (maxappls)” на стр. 419

Этот параметр задает количество памяти, которое отводится для списка блокировок. Для каждой базы данных есть свой список блокировок, содержащий блокировки для всех программ, одновременно соединенных с этой базой данных. Блокировка - это механизм, который менеджер баз данных использует для управления одновременным обращением нескольких программ к данным в базе данных. Могут блокироваться как строки, так и таблицы. Менеджер базы данных также может получать блокировки для внутреннего использования.

Дополнительную информацию о блокировках смотрите в разделе “Блокировка” на стр. 52.

Каждой блокировке требуется 36 или 72 байта списка блокировок в зависимости от того, есть ли для данного объекта другие блокировки:

- Если других блокировок нет, для блокировки данного объекта требуется 72 байта

- Если для данного объекта уже есть блокировки, для записи новой блокировки требуется 36 байт.

Когда процент списка блокировок, используемых программой, достигает *maxlocks*, менеджер баз данных выполняет для этих блокировок расширение - блокировки строк преобразуются в блокировку таблицы (как описано ниже). Хотя сам процесс расширения не занимает много времени, блокировки целых таблиц (по сравнению с блокировками отдельных строк) уменьшают степень параллелизма, и может упасть общая производительность базы данных для последовательных обращений к этим таблицам. Для управления списком блокировок есть следующие рекомендации:

- Для освобождения блокировок выполняйте частые принятия (COMMIT).
- Перед тем, как выполнить большое число изменений, блокируйте всю таблицу (с использованием оператора SQL LOCK TABLE). При этом используется только одна блокировка, и другие программы не будут мешать изменениям, хотя параллелизм данных и уменьшится.

  Кроме того, для управления способом блокировки заданной таблицы можно воспользоваться параметром LOCKSIZE оператора ALTER TABLE. Подробную информацию смотрите в справочнике *SQL Reference*.

  К автоматической блокировке таблиц может привести использование уровня изоляции Многократное чтение. Дополнительную информацию об уровнях изоляции смотрите в разделе “Глава 3. Особенности прикладного программирования” на стр. 43.
- Для уменьшения числа совместно используемых удерживаемых блокировок используйте по возможности уровень изоляции Стабильность на уровне указателя. Еще сильнее можно уменьшить число блокировок, если вместо уровня Стабильность на уровне указателя использовать уровень изоляции Чтение непринятого, если при этом обеспечиваются требования программ в отношении целостности .

При заполнении списка блокировок может упасть производительность, так как при расширении блокировок будет генерироваться больше блокировок таблиц и меньше блокировок строк, что что приведет к уменьшению параллелизма на совместно используемых объектах в базе данных. Кроме того, может увеличиться число тупиковых ситуаций между программами, приводящими к откатам транзакций (поскольку все программы будут ждать нескольких блокировок таблиц). Когда для базы данных будет достигнуто максимальное число требований блокировок, в программу возвращается SQLCODE -912.

Рекомендация: Если вы опасаетесь, что расширения блокировок приведут к снижению производительности, может потребоваться увеличить значение данного параметра или параметра *maxlocks*. Наличие расширений блокировок можно определить, воспользовавшись системным монитором баз данных.

Кроме того, смотрите описание элемента монитора *lock_escals (расширения блокировок)* в книге *System Monitor Guide and Reference*.

Чтобы определить число страниц, требующихся для списка блокировок, выполните следующие действия:

1. Вычислите нижний предел для размера списка блокировок:

   ```
   (512 * 36 * maxappls) / 4096
   ```

   где 512 - оценка среднего числа блокировок на программу, maxappls - максимальное число программ, а 36 - число байтов для каждой блокировки объекта, где уже есть блокировки.
2. Вычислите верхний предел для размера списка блокировок:

   ```
   (512 * 72 * maxappls) / 4096
   ```

   где 72 - число байтов для первой блокировки объекта.
3. Оцените степень параллелизма для данных и, исходя из того, что вы хотите получить, выберите для параметра *locklist* начальное значение между нижней и верхней границами, которые вы вычислили.
4. Настройте значение данного параметра с помощью системного монитора баз данных, как описано ниже.

Используя системный монитор баз данных, можно определить максимальное число блокировок, удерживаемых данной транзакцией.

Дополнительную информацию смотрите в описании элемента монитора *locks_held_top (максимальное число удерживаемых блокировок)* в книге *System Monitor Guide and Reference*.

Эти сведения помогут проверить и скорректировать оцененное число блокировок на программу. В ходе выполнения этой проверки придется опробовать несколько программ (не забудьте, что информация монитора предоставляется на уровне транзакции, а не на уровне программы).

Увеличение значения *locklist* может также потребоваться, если увеличивается значение *maxappls*, или если запускаемые программы выполняют принятия редко.

После изменения этого параметра, возможно, следует выполнить повторное связывание прикладных программ (используя команду REBIND PACKAGE).

Дополнительную информацию о производительности программ и о влиянии оптимизации запросов смотрите в разделе “Часть 2. Настройка производительности прикладных программ” на стр. 41.

Размер кэша пакета (pckcachesz)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

32-битные платформы UNIX
-1 [-1, 32 – 128000]

64-битные платформы UNIX
-1 [-1, 32 – 524288]

OS/2 и Windows NT
-1 [-1, 32 – 128000]

Единица измерения Страницы (4 Кбайта)

Когда размещается При инициализации базы данных

Когда освобождается При закрытии базы данных

Этот параметр задает размеры кэша, выделяемого из глобальной памяти базы данных и используемого для статических и динамических операторов SQL базы данных. В системе многораздельных баз данных для каждого раздела базы данных используется свой кэш пакета.

Кэширование пакетов позволяет менеджеру баз данных сократить внутренние затраты, устраняя необходимость обращаться при перезагрузке пакета к системным каталогам, или, в случае динамического SQL, устраняя необходимость компиляции. Части пакета хранятся в кэше пакета, пока не произойдет одно из следующих событий:
- База данных закрывается
- Пакет или динамический оператор SQL становятся недействительными
- Кэш переполняется.

Такое кэширование для статического или динамического оператора SQL может улучшить производительность, особенно если один и тот же оператор многократно используется прикладными программами, соединенными с базой данных. Это особенно важно для прикладной программы обработки транзакций.

Если принять значение по умолчанию (-1), для расчета выделения страниц используется восьмикратное значение параметра конфигурации *maxappls*. Исключение имеет место, когда восьмикратное значение *maxappls* меньше 32. В этой ситуации значение по умолчанию -1 задает для параметра *pckcachesz* значение 32.

Рекомендация: При настройке этого параметра необходимо оценить, может ли дополнительная память, зарезервированная под кэш пакета, использоваться

эффективнее, если ее отвести для другой цели, например, для пула буферов. Поэтому при настройке данного параметра следует использовать методы тестирования производительности.

Настройка этого параметра особенно важна, когда изначально используется несколько частей пакета, а затем повторно запускаются лишь немногие из них. Если кэш слишком велик, на хранение копий исходных частей пакета затрачивается лишняя память.

Посмотрите в книге *System Monitor Guide and Reference* сведения о следующих элементах монитора:
- *pkg_cache_lookups* (просмотры кэша пакета)
- *pkg_cache_inserts* (вставки кэша пакета)
- *pkg_cache_size_top* (наибольший размер кэша пакета)
- *pkg_cache_num_overflows* (число переполнений кэша пакета)

Эти элементы системного монитора баз данных могут помочь определить, следует ли вам настраивать данный параметр конфигурации.

Примечание: Кэш пакета - это рабочий кэш, поэтому задавать нулевое значение этого параметра нельзя. В этом кэше должно быть оставлено достаточно памяти для хранения всех операторов SQL, выполняемых в данный момент. Если выделено больше места, чем сейчас требуется, производится кэширование. Если эти разделы понадобятся в дальнейшем, их можно выполнить уже без повторной загрузки или компиляции.

Предел, заданный параметром *pckcachesz*, не задает жесткого ограничения. Он может быть при необходимости превышен, если в совместно используемой памяти базы данных еще есть место. С помощью элемента монитора *pkg_cache_size_top* можно определить максимальный достигнутый размер кэша пакета, а с помощью элемента монитора *pkg_cache_num_overflows* - узнать, сколько раз был превышен предел, заданный параметром *pckcachesz*.

Совместная память прикладных программ

Следующий параметр задает рабочую область, используемую всеми агентами (координирующими агентами и подагентами), работающими для прикладной программы:

- “Размер кучи управления прикладной программы (app_ctl_heap_sz)”

Размер кучи управления прикладной программы (app_ctl_heap_sz)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

Сервер баз данных с локальными и удаленными клиентами 128 [1–64000]

Сервер баз данных с локальными клиентами
64 [1–64000] (для платформ, кроме UNIX)

128 [1–64000] (для платформ на основе UNIX)

Сервер многораздельных баз данных с локальными и удаленными клиентами
256 [1–64000]

Единица измерения Страницы (4 Кбайта)

Когда размещается При запуске программы

Когда освобождается При завершении работы программы

Параметры, связанные с данным

- “Размер кэша каталога (catalogcache_sz)” на стр. 370
- “Размер кучи прикладных программ (applheapsz)” на стр. 385
- “Разрешение внутрираздельного параллелизма (intra_parallel)” на стр. 494

Для многораздельных и одноразделных баз данных с включенным внутренним параллелизмом (intra_parallel=ON) это размер совместно используемой области памяти, выделенной для кучи управления программами. Для одноразделных баз данных, где внутренний параллелизм отключен (intra_parallel=OFF), это максимальная собственная память, которая будет выделена для кучи. На каждое соединение раздела существует одна куча управления программами.

Куча управления программами требуется, прежде всего, для совместного использования информации между агентами, работающими для одного запроса, а в среде многораздельных баз данных - для хранения выполняемых разделов, представляющих операторы SQL. Использование этой кучи минимально в одноразделных базах данных, где выполняются запросы со степенью параллелизма меньше либо равной 1.

Эта куча служит также для хранения информации дескриптора для объявленных временных таблиц. Информация дескриптора для всех объявленных временных таблиц, которые не были явным образом отброшены, сохраняется в памяти этой кучи и не может быть отброшена, пока не отброшена объявленная временная таблица.

Рекомендация: Начните со значения по умолчанию. Если вы работаете со сложными прикладными программами, ваша система содержит большое число разделов баз данных или же вы используете объявленные временные таблицы, может потребоваться увеличить значение этого параметра. Количество требуемой памяти возрастает с увеличением числа одновременно активных объявленных временных таблиц. У объявленной временной таблица со многими столбцами размер дескриптора таблицы больше, чем у таблицы с малым числом столбцов, поэтому большое число столбцов в объявленных временных таблицах также увеличивает требования к куче управления программы.

Собственная память агента

Следующие параметры влияют на объем памяти, используемый для каждого агента базы данных:

- “Размер кучи сортировки (sortheap)”.
- “Порог кучи сортировки (sheapthres)” на стр. 382.
- “Размер кучи операторов (stmtheap)” на стр. 385.
- “Размер кучи прикладных программ (applheapsz)” на стр. 385.
- “Размер кучи статистики (stat_heap_sz)” на стр. 386.
- “Размер кучи запросов (query_heap_sz)” на стр. 387.
- “Параметр drda_heap_sz (размер кучи DRDA)” на стр. 388.
- “Размер совместной памяти пользовательских функций (udf_mem_sz)” на стр. 389.
- “Размер стека агента (agent_stack_sz)” на стр. 390.
- “Минимальная принятая частная память (min_priv_mem)” на стр. 392.
- “Порог собственной памяти (priv_mem_thresh)” на стр. 392.
- “Максимальный размер кучи интерпретатора Java (java_heap_sz)” на стр. 405. На платформах на основе UNIX *java_heap_sz* отводится на каждый агент.

Информацию о том, как собственная память агента связана с остальной памятью, отводимой менеджером баз данных, смотрите в разделе “Как DB2 использует память” на стр. 253.

Размер кучи сортировки (sortheap)

Тип конфигурации	База данных
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	256 [16 – 524288]
Единица измерения	Страницы (4 Кбайта)
Когда размещается	Когда требуется для выполнения сортировок
Когда освобождается	Когда сортировка выполнена

Параметры, связанные с данным

"Порог кучи сортировки (sheapthres)"

Этот параметр задает максимальное число собственных страниц памяти, используемых для собственных сортировок, или максимальное число совместно используемых страниц памяти, используемых для совместных сортировок. Для собственных сортировок этот параметр влияет на собственную память агента. Для совместных сортировок этот параметр влияет на совместно используемую память базы данных. Для каждой сортировки менеджер баз данных выделяет при необходимости отдельную кучу сортировки. Куча сортировки - это область, где сортируются данные. По решению оптимизатора может выделяться куча сортировки меньшего размера, чем задано этим параметром.

Рекомендации:

При работе с кучей сортировки нужно учитывать следующее:

- Выбор подходящих индексов может минимизировать использование кучи сортировки.
- Буферы хеш-объединений и динамические битовые образы (используемые для операций AND для индексов и для объединений типа ″звезда″) используют память кучи сортировки. Если используются эти техники, увеличьте размер этого параметра.
- Увеличьте размер этого параметра, если часто требуются сортировки большого объема данных.
- Увеличивая значение этого параметра, проверьте, не нужно ли также изменить значение параметра *sheapthres* в файле конфигурации менеджера баз данных.
- Размер кучи сортировки используется оптимизатором при определении плана доступа. После изменения этого параметра, возможно, следует выполнить повторное связывание прикладных программ (используя команду REBIND PACKAGE).

Порог кучи сортировки (sheapthres)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

32-битные платформы UNIX
20000 [250 – 2097152]

64-битные платформы UNIX
20000 [250 – 2147483647]

OS/2 и Windows NT
10000 [250 – 2097152]

Единица измерения Страницы (4 Кбайта)

Параметры, связанные с данным
“Размер кучи сортировки (sortheap)” на стр. 381

Собственные и совместно используемые сортировки используют память из двух различных источников. Размер области памяти совместно используемой сортировки статически определяется во время первого соединения с базой данных на основе значения параметра *sheapthres*. Размер области памяти для собственных сортировок не ограничивается.

Для собственных и совместно используемых сортировок параметр *sheapthres* используется по-разному:

- Для собственных сортировок этот параметр задает *мягкий* пределом на уровне экземпляра для объема памяти, который может быть занят собственными сортировками в любой момент времени. Когда общие затраты памяти на собственные сортировки для экземпляра достигнут этого предела, память, выделяемая на дополнительные поступающие требования собственных сортировок, будет существенно сокращена.
- Для совместно используемых сортировок этот параметр задает жесткий предел на уровне базы данных для объема памяти, который используется совместно используемыми сортировками в любой момент времени. При достижении этого предела никакие дальнейшие требования памяти на совместно используемые сортировки не допускаются (пока общие затраты памяти на совместно используемые сортировки не упадут ниже предела, заданного параметром *sheapthres*).

Примеры операций, использующих кучу сортировки: сортировки, хеш-объединения, динамические битовые образы (используемые для операций AND на индексами и объединений типа ″звезда″) и операции, в которых таблицы находятся в памяти.

Явное задание этого порога предохраняет от использования менеджером баз данных чрезмерных объемов памяти для большого числа сортировок.

Не нужно увеличивать значение этого параметра при переходе из одноузловой среды в многоузловую. Если параметры конфигурации базы данных и

менеджера баз данных настроены в одноузловой среде (в DB2 EE), в большинстве случаев эти же параметры дадут хорошие результаты в многоузловой среде (в DB2 EEE).

Поскольку параметр порога кучи сортировки является параметром конфигурации менеджера баз данных, он применяется для всего экземпляра DB2. Единственный способ задать для этого параметра различные значения на разных узлах или разделах - создать несколько экземпляров DB2. При этом на разных группах узлов должны быть разные базы данных DB2. Такое решение делает бессмысленными многие преимущества среды многораздельных баз данных.

Рекомендация: В идеальном случае для этого параметра следует установить значение, кратное (в разумных пределах) наибольшему значению параметра *sortheap* для экземпляра менеджера баз данных. Значение этого параметра должно **как минимум** вдвое превышать значение параметра *sortheap* для любой базы данных в экземпляре.

Если выполняются собственные сортировки, и в системе достаточно памяти, идеальное значение для этого параметра можно вычислить так:

1. Для каждой базы данных вычислите типично используемую кучу сортировки:

   ```
   (типичное число одновременно запущенных в базе данных агентов)
   * (заданная для этой базы данных куча сортировки)
   ```

2. Вычислите сумму этих результатов - размер общей кучи сортировки, которую можно использовать в типичных случаях для всех баз данных в данном экземпляре.

Информацию о сортировках в среде SMP смотрите в разделе “Стратегии параллельной сортировки” на стр. 205.

Чтобы настроить этот параметр и найти правильный компромисс между производительностью сортировки и использованием памяти, следует воспользоваться методами измерения производительности. Дополнительную информацию смотрите в разделе “Глава 12. Измерение производительности” на стр. 335.

Дополнительную информацию о сортировке можно найти также в разделе “Сортировка” на стр. 278.

Для трассировки сортировок можно воспользоваться системным монитором баз данных.

Дополнительную информации о постпороговой сортировке смотрите в описании элемента монитора *post_threshold_sorts* в руководстве *System Monitor Guide and Reference*.

Размер кучи операторов (stmtheap)

Тип конфигурации	База данных
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	2048 [128 – 60000]
Единица измерения	Страницы (4 Кбайта)
Когда размещается	Для каждого оператора во время прекомпиляции или связывания
Когда освобождается	После завершения прекомпиляции или связывания каждого оператора

Компилятор SQL использует кучу операторов в качестве рабочего пространства при компиляции оператора SQL. Данный параметр задает размер этого рабочего пространства.

Эта область не остается выделенной постоянно, она выделяется и освобождается для каждого обрабатываемого оператора SQL. Учтите, что для динамических операторов SQL эта рабочая область будет использоваться во время выполнения программы, в то время как для статических операторов SQL она используется во время процесса связывания, но не во время выполнения программы.

Рекомендация: В большинстве случаев для этого параметра можно использовать значение по умолчанию. Если при попытке оптимизации очень сложного оператора SQL менеджер баз данных выдает сообщение об ошибке (слишком сложный оператор), постепенно увеличивайте значение этого параметра на одну и ту же величину (например, 256 или 1024), пока ошибка не перестанет повторяться.

Размер кучи прикладных программ (applheapsz)

Тип конфигурации	База данных
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	128 [16 – 60000]
	64 [16 – 60000] (в среде многораздельной базы данных)
Единица измерения	Страницы (4 Кбайта)
Когда размещается	Когда агент инициализируется для выполнения работы для прикладной программы
Когда освобождается	Когда агент завершает работу для прикладной программы

Параметры, связанные с данным

“Размер кучи управления прикладной программы (app_ctl_heap_sz)” на стр. 379

Этот параметр задает число собственных страниц памяти, которые менеджер баз данных может использовать для конкретного агента или подагента.

Эта куча выделяется при инициализации агента или подагента для прикладной программы. Выделяется минимальный объем памяти, необходимый для обработки запроса, переданного этому агенту или подагенту. Если агенту или подагенту требуется больший объем кучи для обработки более сложных операторов SQL, менеджер баз данных выделяет нужный объем памяти, но не более максимального объема, заданного этим параметром.

Примечание: В среде многораздельных баз данных куча управления прикладными программами (*app_ctl_heap_sz*) используется для хранения копий выполняемых частей операторов SQL для агентов и подагентов. Однако подагенты SMP используют *applheapsz* так же, как агенты во всех других средах.

Рекомендация: Увеличьте значение этого параметра, если прикладная программа получает код ошибки, указывающий, что в куче прикладных программ недостаточно памяти.

Куча прикладных программ (*applheapsz*) выделяется в собственной памяти агента.

Размер кучи статистики (stat_heap_sz)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 4384 [1096 – 524288]

Единица измерения Страницы (4 Кбайта)

Когда размещается При запуске утилиты RUNSTATS

Когда освобождается При завершении работы утилиты RUNSTATS

Параметры, связанные с данным

- “Число сохраняемых частых значений (num_freqvalues)” на стр. 471
- “Число квантилей для столбцов (num_quantiles)” на стр. 472

Этот параметр задает **максимальный** размер кучи, используемый для сбора статистики по команде RUNSTATS.

Рекомендация: Значение по умолчанию приемлемо, если не собирается статистика распределения или же эта статистика собирается для таблиц с относительно небольшим числом столбцов. Минимальное значение **не** рекомендуется при сборе статистики распределения, так как его достаточно лишь для таблиц с одним - двумя столбцами.

Надо настроить этот параметр на основе числа столбцов, для которых собирается статистика. Таблицы с относительно небольшим числом столбцов требуют меньше памяти для сбора статистики распределения. Широкие таблицы с большим числом столбцов требуют значительно большей памяти. Если вы собираете статистику распределения для таблиц с большим числом столбцов, что требует большого объема кучи статистики, лучше делать это в период малой активности системы, чтобы требования к памяти не мешали работе других пользователей.

Размер кучи запросов (query_heap_sz)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 1000 [2 – 524288]

Единица измерения Страницы (4 Кбайта)

Когда размещается При соединении программы (локальной или удаленной) с базой данных

Когда освобождается При отсоединении программы от базы данных или отключении ее от экземпляра

Параметры, связанные с данным

“Размер кучи слоя поддержки программ (aslheapsz)” на стр. 395

Этот параметр задает **максимальное** количество памяти, которое может быть отведено для кучи запросов. Куча запросов используется для хранения каждого запроса в собственной памяти агента. Информация для каждого запроса состоит из входной и выходной SQLDA, текста оператора, SQLCA, имени пакета, создателя, номера части и маркера согласованности. Этот параметр нужен,

чтобы предотвратить использование программой неоправданно большого количества виртуальной памяти в агенте.

Куча запросов используется также для выделения памяти для размещения указателей блокировки. Эта память состоит из блока управления указателем и полностью разрешенной выходной SQLDA.

Исходно выделяемая куча запросов будет иметь тот же размер, что и куча слоя поддержки программ, заданная параметром *aslheapsz*. Размер кучи запросов должен быть не менее 2 и не меньше значения параметра *aslheapsz*. Если размер кучи запросов недостаточен для обработки данного запроса, куча будет размещена заново с требуемым размером (но не больше *query_heap_sz*). Если размер кучи запросов более, чем в полтора раза превосходит значение *aslheapsz*, по окончании запроса куча будет переразмещена с размером *aslheapsz*.

Рекомендация: в большинстве случаев можно обойтись значением по умолчанию. Заданное значение *query_heap_sz* должно не менее чем в пять раз превышать значение *aslheapsz*. Это позволит запросам использовать больше, чем *aslheapsz*, памяти и в то же время позволит открытия в любой момент три или четыре указателя блокирования.

При работе с очень большими объектами вам может понадобиться увеличить значение этого параметра, чтобы размер кучи запросов позволял разместить эти большие объекты.

Параметр drda_heap_sz (размер кучи DRDA)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 128 [16 – 60000]

Единица измерения Страницы (4 Кбайта)

Когда размещается

- Сервер прикладных программ (AS) DRDA размещает кучу DRDA каждый раз, когда

реквестер прикладных программ (AR) DRDA соединяется с базой данных DB2
- DB2 Connect размещает кучу DRDA каждый раз, когда соединяется с AS DRDA.

Когда освобождается Когда AR DRDA отсоединяется от базы данных

Этот параметр задает число страниц памяти, которая отводится для DB2 Connect и DRDA Application Server Support Feature. На размер памяти, отводимый для кучи, влияют следующие факторы:
- Число указателей, открытых прикладной программой
- Число входных переменных хоста
- Число элементов в списке выбора
- Размер входных и выходных данных
- Длина связанных и подготовленных операторов SQL.

Рекомендация: Используйте значение по умолчанию, если не получаете кода ошибки, свидетельствующего, что памяти в куче DRDA недостаточно.

Размер совместной памяти пользовательских функций (udf_mem_sz)

Тип конфигурации Менеджер баз данных

Применяется к
- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 256 [128 – 60000]

Единица измерения Страницы (4 Кбайта)

Когда размещается При запуске пользовательской функции

Когда освобождается При завершении пользовательской функции

Этот параметр применяется как к изолированным, так и к неизолированным пользовательским функциям. Для изолированных пользовательских функций он задает выделяемую по умолчанию совместно используемую память для процесса баз данных и пользовательских функций. В среде одноразделных баз данных есть только один набор совместной памяти. В среде многораздельных баз данных совместная память есть на каждом сервере разделов баз данных, и все агенты и подагенты прикладных программ, запущенные на этом сервере, используют одну и ту же совместную память.

Для неизолированных пользовательских функций этот параметр задает размер собственной памяти. В среде одноразделных баз данных куча выделяется из собственной памяти. В среде многоразделных баз данных куча выделяется из глобальной памяти прикладных программ на каждом сервере раздела базы данных, и все агенты и подагенты, работающие для прикладной программы на этом сервере раздела базы данных, используют одну и ту же совместную память.

Как для изолированных, так и для неизолированных пользовательских функций эта память используется для передачи данных между пользовательской функцией и базой данных.

Если в программах не используются пользовательские функции, память не выделяется. Если в одной и той же программе изолированные пользовательские функции работают вместе с неизолированными, выделяется две памяти: одна для изолированных функций, а другая для неизолированных.

Дополнительную информацию о пользовательских функциях смотрите в книгах *Application Development Guide* и *SQL Reference*.

Рекомендация: Значение по умолчанию должно быть достаточным для всех случаев, кроме передачи данные большого объекта пользовательским функциям. Когда данные большого объекта передаются пользовательским функциям, объем выделяемой памяти может понадобиться увеличить. Заданное вами значение этого параметра должно минимум на две страницы превышать размер входных аргументов и результата внешней функции.

Примечание: Требования пользовательских функций к памяти в целом суммируются, поэтому чем больше пользовательских функций используется в прикладной программе, тем больше оптимальное значение данного параметра.

Размер стека агента (agent_stack_sz)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

	OS/2	64 [8 – 1000]
	Windows NT	16 [8 – 1000]
Единица измерения	Страницы (4 Кбайта)	
Когда размещается	Когда агент инициализируется для выполнения работы для прикладной программы	
Когда освобождается	Когда агент завершает работу для прикладной программы	

Стек агента - это виртуальная память, которую DB2 отводит каждому агенту. Эта память выделяется, когда требуется обработать оператор SQL. Этот параметр можно использовать для оптимизации использования памяти сервера для данного набора программ. Сложные запросы требуют больше места в стеке, чем простые.

Для платформ на базе UNIX этот параметр не применяется.

Рекомендация: в большинстве случаев удается использовать размер стека по умолчанию. Лишь в случае, когда ваша среда содержит много особенно сложных запросов, значение этого параметра придется увеличить. Если размер стека недостаточен для обработки вашего оператора SQL, в файл db2diag.log будет сделана запись об ошибке и будет выдан код SQL. Надо увеличить значение *agent_stack_sz* и перезапустить экземпляр базы данных.

Сократить размер стека с целью сделать больше адресного пространства доступным для других клиентов можно, если ваша среда удовлетворяет следующим условиям:
- Содержит только простые программы (например, простую OLTP), в которых полностью отсутствуют сложные запросы
- Требует относительно большого числа одновременно работающих клиентов (например, более 100).

Размер стека агента и число одновременно работающих клиентов связаны обратной зависимостью: при больших размерах стека число потенциальных одновременно работающих клиентов снижается. Это вызвано ограниченностью адресного пространства на платформах OS/2 и Windows NT. Например, предположим, что на OS/2 есть 400 Мбайт адресного пространства (этот объем зависит от файла `config.sys`). Если задать значение *agent_stack_sz* равным 1 Мбайту, можно будет иметь не более 400 агентов. (Фактически же из-за других ограничений адресного пространства, например, связанных с пулами буферов, реальное число агентов будет гораздо меньше.) Это означает, что, если задано более высокое значение *maxagents*, например, 5000), указанный предел никогда не будет достигнут.

Минимальная принятая частная память (min_priv_mem)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 32 [32 – 112000]

Единица измерения Страницы (4 Кбайта)

Когда размещается При запуске менеджера баз данных

Когда освобождается При остановке менеджера баз данных

Параметры, связанные с данным

"Порог собственной памяти (priv_mem_thresh)"

Этот параметр задает число страниц, которые процесс сервера баз данных резервирует в качестве собственной виртуальной памяти при запуске экземпляра менеджера баз данных (`db2start`). Если сервер требует больше собственной памяти, он попытается получить дополнительную память у операционной системы.

Для систем на базе UNIX этот параметр не применяется.

Рекомендация: Используйте значение по умолчанию.

Это значение следует менять, только если вы хотите f дать серверу баз данных больше памяти. Это сэкономит время размещения. Однако если вы зададите слишком большое значение, это может снизить производительность других (не DB2) программ.

Порог собственной памяти (priv_mem_thresh)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

	• Сервер сателлитных баз данных с удаленными клиентами
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	1296 [-1; 32 – 112000] 32 [-1; 32 – 112000] на сервер сателлитных баз данных с удаленными клиентами
Единица измерения	Страницы (4 Кбайта)
Параметры, связанные с данным	“Минимальная принятая частная память (min_priv_mem)” на стр. 392

Этот параметр используется для задания количества неиспользуемой собственной памяти агентов, которая размещается и готова к использованию при запуске новых агентов. Для платформ на базе UNIX этот параметр не применяется.

Когда агент прекращает работу, менеджер баз данных, вместо того, чтобы автоматически освободить всю память, использовавшуюся этим агентом, освобождает только *избыточную память*, определяемую следующей формулой:

```
Выделенная собственная память -
(используемая собственная память + priv_mem_thresh)
```

Если эта формула дает отрицательное значение, никаких действий не предпринимается.

В приводимой ниже таблице дается пример, поясняющий, когда память размещается и освобождается. В этом примере используется значение параметра *priv_mem_thresh* 100.

Описание действия	Выделенная память	Используемая память
Число запущенных агентов, для которых выделена память.	1000	1000
Запускается новый агент, использующий 100 страниц памяти.	1100	1100
Агент, использовавший 200 страниц памяти, прекращает работу. (Обратите внимание на то, что при этом 100 страниц освобождается, а еще 100 остаются размещенными для возможного использования в будущем.)	1000	900
Агент, использовавший 50 страниц памяти, прекращает работу. (Обратите внимание на то, что 50 страниц памяти освобождается, а 100 дополнительных страниц - не используемых существующими агентами - остаются размещенными.)	950	850
Запускается новый агент, требующий 150 страниц памяти. (100 из 150 страниц уже размещены, и менеджеру баз данных надо разместить только 50 дополнительных страниц для этого агента.)	1000	1000

При значении “-1” для данного параметра будет использоваться значение параметра *min_priv_mem*.

Рекомендация: При настройке этого параметра следует принять во внимание, как происходит соединение/отсоединение клиента, а также требования к памяти других процессов на том же компьютере.

Если период, в течение которого много клиентов одновременно соединено с базой данных, короткий, высокий порог препятствует освобождению неиспользованной памяти и ее доступности для других процессов. В этих условиях нарушается управление памятью, что может повлиять на другие процессы, использующие память.

Если число одновременно работающих клиентов в среднем постоянно, но локально часто колеблется, высокий порог поможет сохранять доступность памяти для процессов клиента и сократит расходы на размещение и освобождение памяти.

Память связи агента с прикладной программой

Следующие параметры влияют на объем памяти, который отводится для передачи данных между прикладной программой и процессами агентов:

- “Размер кучи слоя поддержки программ (aslheapsz)” на стр. 395

- “Параметр min_dec_div_3” на стр. 396
- “Размер блока ввода-вывода клиента (rqrioblk)” на стр. 398
- “Размер блока ввода-вывода реквестера DOS (dos_rqrioblk)” на стр. 399

Информацию о том, как совместная память агентов и программ связана с остальной памятью, отводимой менеджером баз данных, смотрите в разделе “Как DB2 использует память” на стр. 253.

Размер кучи слоя поддержки программ (aslheapsz)

Тип конфигурации Менеджер баз данных

Применяется к
- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 15 [1 – 524288]

Единица измерения Страницы (4 Кбайта)

Когда размещается Когда для локальной прикладной программы запускается агент менеджера баз данных

Когда освобождается Когда завершается процесс агента менеджера баз данных

Параметры, связанные с данным

“Размер кучи запросов (query_heap_sz)” на стр. 387

Куча слоя поддержки программ представляет собой буфер связи между локальной программой и связанным с ним агентом. Этот буфер размещается как совместно используемая память каждым запускаемым агентом менеджера баз данных.

Если требование менеджера баз данных или связанный с ним ответ не умещается в буфере, он будет разделен на две или большее число пар отправления-получения. Размер буфера надо установить таким, чтобы большинство требований использовали бы одну пару отправления-получения. Размер требования определяется на основе памяти, необходимой для:
- Входной SQLDA
- Всех соответствующих данных в SQLVAR
- Выходной SQLDA

- Других полей, обычно не превышающих 250 байт.

Кроме длины буфера связи, этот параметр определяет также размер блока ввода-вывода при открытии указателя с блокированием. Память для указателей выделяется из собственного адресного пространства прикладной программы, то есть вы можете задать оптимальное количество памяти для размещения каждой прикладной программой. Если клиент баз данных не может выделить пространство для указателя с блокированием из собственной памяти программы, открывается указатель без блокирования.

Данные, посылаемые локальной прикладной программой, принимаются менеджером баз данных в непрерывную область памяти, выделяемую из кучи запроса. Параметр *aslheapsz* используется для определения начального размера кучи запроса (как для локальных, так и для удаленных клиентов). Максимальный размер кучи запроса определяется параметром *query_heap_sz*.

Рекомендация: Если требования программы обычно невелики, и программа работает в системе с ограничениями по памяти, вы можете захотеть уменьшить значение этого параметра. Если ваши запросы обычно весьма велики, и для них требуется несколько требований отправки и приема, а ограничений по памяти в системе нет, вы можете захотеть увеличить значение этого параметра.

Чтобы вычислить минимальное число страниц для *aslheapsz*, используйте следующую формулу:

```
aslheapsz >= ( sizeof(входная SQLDA)
             + sizeof(каждая входная SQLVAR)
             + sizeof(выходная SQLDA)
             + 250 ) / 4096
```

где sizeof(x) - размер x в байтах, который определяет число страниц для данного входного или выходного значения.

Надо также учесть влияние этого параметра на число и потенциальный размер указателей с блокированием. Большие блоки строк могут повысить производительность, если число или размер передаваемых строк велики (например, объем данных больше 4096 байт). Однако увеличение блоков записей увеличивает размер памяти, необходимый для каждого соединения.

Большие размеры блоков могут также повысить число операций выборки, необходимых прикладной программе. Управлять числом требований выборки можно при помощи условия OPTIMIZE FOR оператора SELECT в вашей программе. Дополнительную информацию об условии OPTIMIZE FOR смотрите в разделе “Условие OPTIMIZE FOR n ROWS” на стр. 80.

Параметр min_dec_div_3

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] No [Yes, No]

С добавлением параметра конфигурации базы данных *min_dec_div_3* появился быстрый способ изменить вычисление масштаба десятичного деления в SQL. Этот параметр может иметь значение ″Yes″ или ″No″. Значение по умолчанию - ″No″.

Параметр конфигурации базы данных *min_dec_div_3* изменяет масштаб результата десятичной арифметической операции, где используется деление. Если задано значение ″No″, масштаб вычисляется как 31-p+s-s'. Дополнительную информацию смотрите в справочнике *SQL Reference*, Глава 3, ″Decimal Arithmetic in SQL″. Если задано значение ″Yes″, масштаб вычисляется как MAX(3, 31-p+s-s'). Это приводит к тому, что результат десятичного деления всегда имеет масштаб как минимум 3. Точность всегда равна 31.

Изменение этого параметра конфигурации базы данных может привести к изменениям в программах для существующих баз данных. Это может произойти, если изменение этого параметра конфигурации базы данных повлияет на полученный при десятичном делении масштаб. Ниже приведены некоторые возможные сценарии, которые могут влиять на программы. Эти сценарии следует учесть перед изменением параметра *min_dec_div_3* на сервере баз данных с действующими базами данных.

- При изменении масштаба результата одного из столбцов производной таблицы, которая определяется в среде с такой конфигурацией, при обращении к этой производной таблице после изменения данного параметра конфигурации базы данных может случиться ошибка с SQLCODE -344. Сообщение SQL0344N относится к рекурсивным общим табличным выражениям, однако если именем объекта (первый маркер) является производная таблица, чтобы избежать этой ошибки, вам необходимо отбросить эту производную таблицу и создать ее заново.
- Поведение статического пакета не изменится, пока не связать его повторно, неявно или явно. Например, после изменения значения с NO на YES дополнительные разряды масштаба не будут включены в результаты, пока не будет выполнено повторное связывание. Чтобы провести повторное связывание принудительно для любых измененных статических пакетов, можно воспользоваться явной командой повторного связывания (REBIND).
- Проверочное ограничение, в котором используется десятичное деление, может отклонить некоторые значения, допустимые ранее. Такие строки будут теперь нарушать ограничение, но они не будут обнаружены, пока не изменится один из столбцов, участвующих в строке проверочного ограничения, или не будет выполнен оператор SET INTEGRITY с опцией IMMEDIATE CHECKED. Для принудительной проверки такого ограничения выполните оператор ALTER TABLE, чтобы отбросить проверочное ограничение, а затем снова добавьте это ограничение, выполнив оператор ALTER TABLE еще раз.

Примечание: В DB2 Версии 7 действуют также следующие ограничения:

1. Параметр *min_dec_div_3* не выводится командой `GET DB CFG FOR DBNAME`. Лучший способ узнать текущее значение - посмотреть побочный эффект результата десятичного деления. Например, рассмотрим следующий оператор:

   ```
   VALUES (DEC(1,31,0)/DEC(1,31,5))
   ```

   Если этот оператор возвращает sqlcode SQL0419N, база данных не поддерживает параметр *min_dec_div_3*, или для него установлено значение ″No″. Если данный оператор возвращает `1,000`, для параметра *min_dec_div_3* установлено значение ″Yes″.
2. Если ввести приведенную ниже команду, параметр *min_dec_div_3* не выводится в списке ключевых слов конфигурации: `? UPDATE DB CFG`

Размер блока ввода-вывода клиента (rqrioblk)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 32767 [4096 – 65535]

Единица измерения Байты

Когда размещается

- Когда удаленная клиентская прикладная программа выдает требование соединения с базой данных сервера
- Когда открывается указатель с блокированием, на клиенте открываются дополнительные блоки

Когда освобождается

- Когда удаленная прикладная программа отсоединяется от базы данных сервера
- Когда закрывается указатель с блокированием

Параметры, связанные с данным

"Размер блока ввода-вывода реквестера DOS (dos_rqrioblk)"

Этот параметр задает размер буфера связи между удаленными программами и их агентами баз данных на сервере баз данных. Когда клиент баз данных требует соединения с удаленной базой данных, на клиенте размещается буфер связи такого размера. На сервере баз данных изначально размещается буфер связи размером 32767 байт, который используется, пока соединение не будет установлено и сервер не сможет определить значение *rqrioblk* на клиенте. Когда сервер будет знать это значение, он переразместит буфер связи, если длина буфера на клиенте не равна 32767 байтам.

Кроме длины буфера связи, этот параметр определяет также размер блока ввода-вывода на клиенте баз данных при открытии указателя с блокированием. Память для указателей выделяется из собственного адресного пространства прикладной программы, то есть вы можете задать оптимальное количество памяти для размещения каждой прикладной программой. Если клиент баз данных не может выделить пространство для указателя с блокированием из собственной памяти программы, открывается указатель без блокирования.

Рекомендация: Если используются указатели без блокирования, смысл увеличить значение этого параметра есть, если данные (например, данные большого объекта), передаваемые в одном операторе SQL, настолько велики, что значения по умолчанию недостаточно.

Надо также учесть влияние этого параметра на число и потенциальный размер указателей с блокированием. Большие блоки строк могут повысить производительность, если число или размер передаваемых строк велики (например, объем данных больше 4096 байт). Однако увеличение блоков записей увеличивает размер памяти, необходимый для каждого соединения.

Большие размеры блоков могут также повысить число операций выборки, необходимых прикладной программе. Управлять числом требований выборки можно при помощи условия OPTIMIZE FOR оператора SELECT в вашей программе. Дополнительную информацию об условии OPTIMIZE FOR смотрите в разделе "Условие OPTIMIZE FOR n ROWS" на стр. 80.

Размер блока ввода-вывода реквестера DOS (dos_rqrioblk)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент

- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 4096 [4096 – 65535]

Единица измерения Байты

Когда размещается

- Когда удаленный клиент DOS или Windows 3.1 требует соединения с базой данных сервера
- Когда открывается указатель с блокированием, на клиенте открываются дополнительные блоки

Когда освобождается

- Когда удаленная прикладная программа отсоединяется от базы данных
- Когда закрывается указатель с блокированием

Параметры, связанные с данным

"Размер блока ввода-вывода клиента (rqrioblk)" на стр. 398

Этот параметр задает размер буфера связи между программами DOS/Windows 3.1 и их агентами баз данных на сервере баз данных. Этот параметр подобен *rqrioblk*, однако его использование позволяет задать для клиентов DOS/Windows 3.1 другой размер блоков. В файле конфигурации DB2 можно задать и параметр *rqrioblk* (используемый для клиентов на 32-битных платформах Windows, OS/2 и UNIX), и параметр *dos_rqrioblk* (используемый для клиентов DOS и Windows 3.1).

Кроме длины буфера связи, этот параметр определяет также размер блока ввода-вывода на клиенте баз данных при открытии указателя с блокированием. Память для указателей выделяется из собственного адресного пространства прикладной программы, то есть вы можете задать оптимальное количество памяти для размещения каждой прикладной программой. Если клиент баз данных не может выделить пространство для указателя с блокированием из собственной памяти программы, открывается указатель без блокирования.

Рекомендация: Если используются указатели без блокирования, смысл увеличить значение этого параметра есть, если данные (например, данные

большого объекта), передаваемые в одном операторе SQL, настолько велики, что значения по умолчанию недостаточно.

Надо также учесть влияние этого параметра на число и потенциальный размер указателей с блокированием. Большие блоки строк могут повысить производительность, если число или размер передаваемых строк велики (например, объем данных больше 4096 байт). Однако увеличение блоков записей увеличивает размер памяти, необходимый для каждого соединения.

Большие размеры блоков могут также повысить число операций выборки, необходимых прикладной программе. Управлять числом требований выборки можно при помощи условия OPTIMIZE FOR оператора SELECT в вашей программе. Дополнительную информацию об условии OPTIMIZE FOR смотрите в разделе “Условие OPTIMIZE FOR n ROWS” на стр. 80.

Память экземпляра менеджера баз данных

Следующие параметры влияют на память, которая отводится и используется на уровне экземпляра:

- “Размер кучи монитора баз данных (mon_heap_sz)”
- “Поддержка кэша каталогов (dir_cache)” на стр. 403
- “Размер буфера аудита (audit_buf_sz)” на стр. 404
- “Максимальный размер кучи интерпретатора Java (java_heap_sz)” на стр. 405

Размер кучи монитора баз данных (mon_heap_sz)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

UNIX 56 [0 – 60000]

OS/2 и Windows NT, Сервер баз данных с локальными и удаленными клиентами и Сервер сателлитных баз данных с удаленными клиентами 32 [0 – 60000]

	OS/2 и Windows NT, Сервер баз данных с локальными клиентами 12 [0 – 60000]
Единица измерения	Страницы (4 Кбайта)
Когда размещается	Когда менеджер баз данных запускается командой *db2start*
Когда освобождается	Когда менеджер баз данных останавливается командой *db2stop*
Параметры, связанные с данным	"Переключатели системного монитора баз данных по умолчанию (dft_monswitches)" на стр. 498

Этот параметр задает объем памяти в страницах, отводимый для данных монитора баз данных. Память из кучи монитора выделяется, когда выполняются действия монитора, такие как снятие снимка, переключение переключателей монитора, сброс монитора или активация монитора событий.

Нулевое значение задает, что менеджер баз данных не будет собирать данные монитора баз данных.

Рекомендация: Объем памяти, необходимый для монитора, зависит от числа наблюдаемых программ (программ, снимающих снимки, или мониторов событий), установленных переключателей и уровня активности базы данных.

Оценить необходимое число страниц можно по следующей формуле:

```
( число наблюдаемых программ + 1 ) *
( число баз данных *
  (800 + ( число используемых таблиц * 20 )
   + ( ( число подключенных программ + 1) *
       (200 + (число табличных пространств * 100) ) ) ) )
/ 4096
```

Если доступная в этой куче память исчерпана, происходит одно из следующих событий:

- При первом соединении программы с базой данных, для которой определен этот монитор событий, в файлы `db2alert.log` и `db2diag.log` записывается сообщение об ошибке уровня 2.
- Если монитор событий, запускаемый динамически оператором SET EVENT MONITOR, завершается неудачно, вашей программе возвращается код ошибки.
- Если команда монитора или подпрограмма API завершается неудачно, вашей программе возвращается код ошибки.

Поддержка кэша каталогов (dir_cache)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] Yes [Yes; No]

Когда размещается

- Собственный кэш размещается, когда программа выполняет свое первое соединение
- Совместно используемый кэш размещается при запуске экземпляра монитора баз данных (`db2start`).

Когда освобождается

- Собственный кэш освобождается при завершении процесса прикладной программы
- Совместно используемый кэш освобождается при остановке экземпляра монитора баз данных (`db2stop`).

Если *dir_cache* имеет значение Yes, база данных, узел и файлы каталога DCS будут кэшироваться в памяти. Использование кэша каталога сокращает расходы, исключая необходимость ввода/вывода и минимизирует поиск в каталоге, требуемый для получении информации каталога. Существует два типа кэшей каталогов:

- Собственный кэш, который размещается и используется для каждого процесса прикладной программы на том компьютере, где она выполняется.
- Совместно используемый кэш, который размещается и используется для некоторых внутренних процессов монитора баз данных.

Примечание: В поддерживаемых средах Windows используется только собственный кэш.

В случае собственного кэша, когда прикладная программа выполняет свое первое соединение, читается каждый файл каталога и информация заносится в

кэш в собственной памяти этой программы. Этот кэш используется процессом прикладной программы при последующих требованиях на соединение и поддерживается в течение всей работы процесса. Если база данных не найдена в собственном кэше, ее поиск проводится в файлах каталога, но кэш при этом не корректируется. Если программа изменяет запись в каталоге, при следующем соединении с этой программой кэш для нее будет обновлен. Собственные кэши других программ при этом не обновляются. При завершении процесса прикладной программы собственный кэш освобождается. (Чтобы обновить кэш каталога, используемый сеансом процессора командной строки, введите команду `db2 terminate`.)

В случае совместно используемого кэша, когда запускается экземпляр монитора баз данных (`db2start`), читается каждый файл каталога и информация заносится в кэш в совместно используемой памяти. Этот кэш используется некоторыми процессами монитора баз данных и поддерживается, пока экземпляр не будет остановлен `db2stop`). Если запись каталога не найдена в этом кэше, ее поиск проводится в файлах каталога. Этот кэш во время работы экземпляра не обновляется.

Рекомендация: Используйте кэширование каталога, если файлы каталога обновляются редко, а производительность для вас критична.

Кроме того, для удаленных клиентов использование кэша может оказаться полезным, если прикладные программы выдают по несколько различных требований на соединение. В этом случае кэширование снижает число обращений программы к файлам каталога.

Кэширование каталога повышает также производительность снимков монитора баз данных. При этом вы должны явно указывать имя базы данных при вызове снимка, а не использовать алиасы.

Примечание: Если кэширование каталога включено, а базы данных вносились в каталог, исключались из каталога, создавались или отбрасывались после запуска монитора баз данных, при вызовах снимков могут происходить ошибки.

Размер буфера аудита (audit_buf_sz)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 0 [0 – 65000]

Единица измерения Страницы (4 Кбайта)

Когда размещается При запуске DB2

Когда освобождается При остановке DB2

Этот параметр задает размер буфера, используемого для аудита базы данных. Дополнительную информацию о возможности аудита смотрите в разделе "Аудит действий DB2" в книге *Administration Guide: Implementation*.

Значение этого параметра по умолчанию - ноль (0). Если значение равно нулю (0), буфер аудита не используется. Если значение больше нуля (0), пространство отводится под буфер для записей аудита, когда они генерируются возможностью аудита. Под буфер отводится указанное число страниц по 4 Кбайта. Буфер аудита не размещается динамически; после изменения значения этого параметра надо остановить и перезапустить DB2, чтобы новое значение вступило в силу.

Если задать для этого параметра значение больше нуля, утилита аудита пишет записи на диск асинхронно по отношению к выполнению операторов, генерирующих эти записи аудита. Это улучшает производительность DB2 по сравнению с отсутствием буфера (при нулевом значении). Если значение этого параметра равно нулю, при аудите записи помещаются на диск синхронно по отношению к выполнению операторов, генерирующих эти записи аудита. Такая работа может снизить производительность прикладных программ, выполняемых в DB2.

Максимальный размер кучи интерпретатора Java (java_heap_sz)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 512 [0 - 4096]

Единица измерения Страницы (4 Кбайта)

Когда размещается При запуске прикладной программы Java

Когда освобождается При завершении прикладной программы Java

Параметры, связанные с данным

"Путь установки Java Development Kit 1.1 (jdk11_path)" на стр. 507

Этот параметр задает максимальный размер кучи, используемой интерпретатором Java.

Используется одна куча для каждого процесса DB2 (одна для каждого агента и подагента на платформах на основе UNIX и одна для каждого экземпляра на других платформах), а также одна куча для каждого процесса изолированной пользовательской функции или изолированной хранимой процедуры. В любом случае эта память выделяется только для агентов или процессов, выполняющих пользовательские функции или хранимые процедуры Java. В системах многораздельных баз данных одно и то же значение используется для всех разделов.

Блокировки

Следующие параметры влияют на управление блокировкой в вашей среде:

- "Интервал проверки тупиковых ситуаций (dlchktime)"
- "Максимальный процент списка блокировок перед расширением (maxlocks)" на стр. 407
- "Срок ожидания блокировки (locktimeout)" на стр. 409

Смотрите также раздел "Максимальная память для списка блокировок (locklist)" на стр. 375.

"Блокировка" на стр. 52 содержит общий обзор того, как менеджер баз данных использует блокировки для поддержания целостности данных.

Интервал проверки тупиковых ситуаций (dlchktime)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 10000 (10 секунд) [1000 – 600000]

Единица измерения Миллисекунды

Параметры, связанные с данным

- "Максимальная память для списка блокировок (locklist)" на стр. 375

- “Максимальный процент списка блокировок перед расширением (maxlocks)”

Тупиковая ситуация возникает, когда две или несколько прикладных программ, соединенных с одной базой данных, бесконечно долго ожидают доступности ресурса. Это ожидание само никогда не завершается, поскольку каждая прикладная программа удерживает ресурс, который нужен для продолжения работы другой прикладной программе.

Интервал проверки тупиковых ситуаций задает частоту, с которой менеджер баз данных проверяет на тупиковые ситуации все программы, соединенные с базой данных.

Примечания:

1. В среде многораздельных баз данных этот параметр применяется только для узла каталога.
2. В среде многораздельных баз данных тупиковая ситуация отмечается только после второго цикла.

Рекомендация: При увеличении значения этого параметра уменьшается частота проверки тупиковых ситуаций, поэтому увеличивается время, затрачиваемое прикладной программой на ожидание, прежде чем тупиковая ситуация будет исправлена.

При уменьшении значения этого параметра увеличивается частота проверки тупиковых ситуаций, поэтому уменьшается время, затрачиваемое прикладной программой на ожидание, прежде чем будет исправлена тупиковая ситуация, но увеличивается время, затрачиваемое менеджером баз данных на проверку тупиковых ситуаций. Если интервал проверки тупиковых ситуаций слишком мал, может ухудшиться производительность времени выполнения, поскольку менеджер баз данных будет часто выполнять проверку тупиковых ситуаций. Уменьшая значения этого параметра для улучшения одновременности, убедитесь, что заданы подходящие значения параметров *maxlocks* и *locklist*, чтобы избежать ненужных расширений блокировок, которые могут привести к большему числу конфликтов блокировки и в результате к большему числу тупиковых ситуаций.

Максимальный процент списка блокировок перед расширением (maxlocks)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

UNIX 10 [1 – 100]

OS/2 и Windows NT
22 [1 – 100]

Единица измерения Проценты

Параметры, связанные с данным

- “Максимальная память для списка блокировок (locklist)” на стр. 375
- “Максимальное число активных прикладных программ (maxappls)” на стр. 419

Расширение блокировок - это процесс замены блокировок строк блокировками таблиц, в результате чего число блокировок в списке снижается. Этот параметр определяет процент заполнения списка блокировок, удерживаемых прикладной программой, после которого менеджер баз данных выполняет расширение блокировок. Когда число блокировок, удерживаемых какой-либо программой, достигает этого процента от размера всего списка, для блокировок, удерживаемых этой программой, происходит расширение. Расширение происходит также при нехватке места в списке блокировок.

Менеджер баз данных определяет, какие блокировки надо расширить, просматривая список блокировок для программы и отыскивая таблицу с наибольшим числом блокировок строк. Если после замены этих блокировок строк одной блокировкой таблицы значение *maxlocks* более не превышается, расширение блокировок прекращается. В противном случае процесс расширения будет продолжаться, пока процент заполнения списка блокировок не станет ниже значения *maxlocks*. Произведение значений параметров *maxlocks* и *maxappls* не может быть меньше 100.

Рекомендация: Следующая формула позволяет настроить параметр *maxlocks*, чтобы разрешить программе удерживать число блокировок в два раза больше среднего:

```
maxlocks = 2 * 100 / maxappls
```

Где 2 увеличивает среднее значение в два раза, а 100 - наибольший допустимый процент. Если параллельно работают всего несколько программ, вместо первой формулы можно использовать такую:

```
 maxlocks = 2 * 100 / (среднее число программ, работающих
параллельно)
```

Одна из особенностей настройки параметра *maxlocks*: его следует конфигурировать совместно с размером списка блокировок (*locklist*). Фактическое предельное число блокировок, удерживаемых программой перед расширением блокировок:

```
maxlocks * locklist * 4096 / (100 * 36)
```

Где 4096 - число байтов на странице, 100 - наибольший процент, допустимый для *maxlocks* и 36 - число байтов на блокировку. Если вы знаете, что одна из программ требует 1000 блокировок, и не хотите, чтобы произошло расширение блокировок, выберите в этой формуле значения для *maxlocks* и *locklist* так, чтобы результат был больше 1000. (Задав в этой формуле для *maxlocks* 10, а для *locklist* 100, в результате вы получите больше 1000 необходимых блокировок.)

Если для *maxlocks* установить слишком низкое значение, расширение блокировок произойдет, когда для других параллельных программ все еще будет оставаться достаточно пространства блокировок. Если для *maxlocks* установить слишком высокое значение, большую часть пространства блокировок могут задействовать несколько программ, и расширение блокировок придется выполнять другим программам. Такая потребность в расширении блокировок приводит в этом случае к плохому параллелизму.

Для отслеживания и настройки этого параметра конфигурации можно использовать системный монитор баз данных.

Срок ожидания блокировки (locktimeout)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] -1 [-1; 0 – 30000]

Единица измерения Секунды

Параметры, связанные с данным

- “Максимальная память для списка блокировок (locklist)” на стр. 375
- “Максимальный процент списка блокировок перед расширением (maxlocks)” на стр. 407

Этот параметр задает срок в секундах, который программа будет ждать получения блокировки. Это помогает избегать глобальных тупиковых ситуаций для программ.

Если задать значение этого параметра 0, ожидания блокировок не происходит. В этой ситуации, если во время требования доступные блокировки отсутствуют, программа немедленно получит код -911.

Если задать значение этого параметра -1, обнаружение истечения срока ожидания блокировок будет выключено. В этой ситуации ожидание блокировки будет происходить (если таковая еще не доступна во время запроса) до тех пор, пока:

- Блокировка не будет предоставлена
- Возникнет тупиковая ситуация.

Рекомендация: В среде обработки транзакций (OLTP) используйте в качестве начального значение 30 секунд. В среде только с запросами следует начать с более высокого значения. В обоих случаях при настройке данного параметра следует использовать методы тестирования производительности.

При работе с менеджером связей данных, обнаружив в файле db2diag.log экземпляра менеджера связей данных (dlfm) истечения срока ожидания блокировок, следует увеличить значение *locktimeout*. Возможно, придется увеличить также значение *locklist*.

Значение надо задать так, чтобы быстро выявлять случаи ожидания, вызванные ненормальной ситуацией, например, при ″приостановке″ транзакции (вероятно, в результате ухода пользователя с рабочей станции). Оно должно быть достаточно высоким, чтобы допустимые требования блокировок выполнялись до истечения срока ожидания из-за чрезмерной рабочей нагрузки, когда ожидание блокировок затягивается.

Чтобы отслеживать случаи истечения срока ожидания для какой-либо прикладной программы (соединения) или случаи, когда база данных обнаруживает такую ситуацию для всех соединенных с ней программ, можно использовать системный монитор баз данных. Дополнительную информацию смотрите в описании элемента монитора *locks_timeouts* (число истечений срока ожидания блокировки) в книге *System Monitor Guide and Reference*.

Высокие значения элемента монитора *lock_timeout* могут быть обусловлены:
- Слишком низким значением данного параметра конфигурации.
- Длительным удерживанием блокировок программой (транзакцией). Для дальнейшего исследования таких программ можно использовать системный монитор баз данных.
- Проблемами одновременности, которые могут быть вызваны расширениями блокировок (с блокировки уровня строки до блокировки уровня таблицы). Дополнительные сведения смотрите в разделах “Максимальный процент списка блокировок перед расширением (maxlocks)” на стр. 407 и “Максимальная память для списка блокировок (locklist)” на стр. 375.

Дополнительную информацию об использовании этого параметра смотрите в разделе “Ожидание блокировок и истечение сроков ожидания” на стр. 60.

Ввод/вывод и хранение

Следующие параметры могут влиять на стоимость ввода/вывода и хранения, связанную с работой вашей базы данных:

- “Порог числа измененных страниц (chngpgs_thresh)” на стр. 411
- “Число асинхронных чистильщиков страниц (num_iocleaners)” на стр. 412
- “Число серверов ввода/вывода (num_ioservers)” на стр. 413
- “Флаг сортировки индекса (indexsort)” на стр. 414

- “Флаг обнаружения последовательного чтения (seqdetect)” на стр. 415
- “Размер предварительной выборки по умолчанию (dft_prefetch_sz)” на стр. 415
- “Число контейнеров SMS по умолчанию (numsegs)” на стр. 416
- “Размер экстента по умолчанию для табличных пространств (dft_extent_sz)” на стр. 417
- “Размер сегмента расширения памяти (estore_seg_sz)” на стр. 417
- “Число сегментов расширения памяти (num_estore_segs)” на стр. 418

Порог числа измененных страниц (chngpgs_thresh)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 60 [5 – 99]

Единица измерения Проценты

Параметры, связанные с данным

“Число асинхронных чистильщиков страниц (num_iocleaners)” на стр. 412

Асинхронные чистильщики страниц записывают измененные страницы из пула (или пулов) буферов на диск до того, как пространство пула буферов потребуется агенту базы данных. В результате агенты баз данных не должны будут ждать записи измененных страниц и смогут использовать пространство в пуле буферов. Это повышает производительность программ баз данных в целом.

При помощи этого параметра можно задать долю (в процентах) измененных страниц, при которой запускаются асинхронные чистильщики страниц (если они не активны в текущий момент). При запуске чистильщиков страниц они составляют список страниц для записи на диск. Когда запись этих страниц будет выполнена, они снова становятся неактивными и ждут следующего срабатывания триггера.

В среде, где выполняется только чтение (например, для запросов), чистильщики страниц не используются.

Рекомендация: Для баз данных с интенсивными транзакциями изменения надо обеспечить наличие достаточного числа чистых страниц в пуле буферов, задав значение параметра равным значению по умолчанию или меньше его. Задание большего процента может повысить производительность, если ваша база данных содержит малое число очень больших таблиц.

Число асинхронных чистильщиков страниц (num_iocleaners)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 1 [0 – 255]

Единица измерения Счетчик

Параметры, связанные с данным

- “Размер пула буферов (buffpage)” на стр. 366
- “Порог числа измененных страниц (chngpgs_thresh)” на стр. 411

Этот параметр задает число асинхронных чистильщиков страниц для базы данных. Эти чистильщики страниц записывают измененные страницы из пула буферов на диск до того, как пространство пула буферов потребуется агенту базы данных. В результате агенты баз данных не должны будут ждать записи измененных страниц и смогут использовать пространство в пуле буферов. Это повышает производительность программ баз данных в целом.

Если задано нулевое (0) значение этого параметра, чистильщики страниц не запускаются, и все операции записи страниц из пула буферов на диск выполняют агенты базы данных. Этот параметр может существенно влиять на производительность базы данных, размещенной на многих физических устройствах, так как в этом случае повышается вероятность того, что одно из них будет незанято. Если чистильщики страниц не запущены, ваши прикладные программы могут периодически сталкиваться с состоянием заполнения журнала.

Если прикладные программы для базы данных состоят в основном из транзакций, изменяющих данные, увеличение числа чистильщиков страниц приведет к улучшению производительности. Увеличение числа чистильщиков страниц снижает также время восстановления после сбоев, например, перебоев в питании, так как содержимое базы данных на диске в любое время будет более свежим.

Рекомендация: При настройке этого параметра учитывайте следующие факторы:

- Тип прикладной программы
  - Если база данных работает только с запросами и в нее не вносятся изменения, задайте нулевое (0) значение этого параметра. Исключение составляет случай, когда обработка запросов приводит к созданию большого числа временных таблиц (что можно определить при помощи утилиты объяснения).
  - Если в базе данных выполняются транзакции, задайте значение этого параметра между единицей и числом физических устройств памяти для этой базы данных.
- Рабочая нагрузка

В средах с большим количеством транзакций, изменяющих данные, может потребоваться задать больше чистильщиков страниц.

- Размер пула буферов (*buffpage*)

  В средах с большими пулами буферов может также потребоваться задать больше чистильщиков страниц.

При настройке данного параметра конфигурации можно использовать собранную монитором баз данных информацию монитора событий об активности записи из пула буферов можно использовать:

- Значение параметра можно уменьшить, если выполняются оба следующих условия:
  - *pool_data_writes* примерно равно *pool_async_data_writes*
  - *pool_index_writes* примерно равно *pool_async_index_writes*.
- Значение параметра следует увеличить, если выполняется любое из следующих условий:
  - *pool_data_writes* намного превышает *pool_async_data_writes*
  - *pool_index_writes* намного превышает *pool_async_index_writes*.

Дополнительную информацию смотрите в описании элементов монитора в книге *System Monitor Guide and Reference*.

- *pool_data_writes* (записи данных пула буферов)
- *pool_index_writes* (записи индекса пула буферов)
- *pool_async_data_writes* (асинхронные записи данных пула буферов)
- *pool_async_index_writes* (асинхронные записи индекса пула буферов)

Число серверов ввода/вывода (num_ioservers)

Тип конфигурации	База данных
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	3 [1 – 255] 1 [1 – 255] на сервере сателлитной базы данных с локальными клиентами
Единица измерения	Счетчик
Когда размещается	Когда программа соединяется с базой данных
Когда освобождается	Когда программа отсоединяется от базы данных
Параметры, связанные с данным	• “Размер предварительной выборки по умолчанию (dft_prefetch_sz)” на стр. 415 • “Флаг обнаружения последовательного чтения (seqdetect)” на стр. 415

Серверы ввода/вывода используются для агентов баз данных и выполняют операции предварительного ввода/вывода и асинхронного ввода/вывода в таких утилитах, как утилиты резервного копирования и восстановления. Этот параметр задает число серверов ввода/вывода для базы данных. Не больше, чем это число операций ввода/вывода для предварительной выборки и утилит могут выполняться для базы данных в любой момент времени. Сервер ввода/вывода ждет, пока продолжается начатая им операция ввода/вывода. Операции ввода/вывода без предварительной выборки планируются прямо из агентов баз данных, и параметр *num_ioservers* их не ограничивает.

Рекомендация: Чтобы полностью использовать устройства ввода/вывода в системе, лучше задавать число, на один или два большее числа физических устройств, на которых расположена база данных. Выгоднее сконфигурировать дополнительные серверы ввода/вывода, поскольку с каждым таким сервером связаны минимальные ресурсы, а неиспользуемые серверы ввода/вывода просто простаивают.

Дополнительную информацию смотрите в разделах “Предварительная выборка данных в пул буферов” на стр. 269 и “Настройка серверов ввода/вывода для предварительной выборки и параллельного ввода/вывода” на стр. 273.

Флаг сортировки индекса (indexsort)

Тип конфигурации	База данных
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	Yes [Yes; No]

Этот параметр указывает, будет ли производиться сортировка индексных ключей при создании индекса. Производительности создания индекса повышается при предварительной сортировке, в особенности для индексов с низким отношением кластеризации или показателем кластеризации. Производительность запросов также может повыситься, если индексы создаются с сортировкой. Однако это достигается за счет увеличения необходимого дискового пространства для сортировки вдвое по сравнению с созданием индекса без предварительной сортировки.

Рекомендация: Если у вас достаточно места на диске, используйте значение по умолчанию (Yes). Пространство, необходимое для этой сортировки, примерно равно пространству, необходимому для оператора SELECT столбцов индекса из таблицы с условием ORDER BY по этим столбцам.

Если вы задаете для этого параметра значение No в симметричной мультипроцессорной среде, мультипроцессорная обработка не будет использоваться при создании индекса.

Флаг обнаружения последовательного чтения (seqdetect)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] Yes [Yes; No]

Параметры, связанные с данным

"Размер предварительной выборки по умолчанию (dft_prefetch_sz)"

Менеджер баз данных может отслеживать операции ввода/вывода и, обнаружив чтение последовательных страниц, активировать предварительную выборку. Этот тип последовательной предварительной выборки называется *обнаружением последовательного чтения*. Параметр конфигурации *seqdetect* позволяет управлять применением менеджером баз данных обнаружения последовательного чтения.

Если этот параметр имеет значение NO, предварительное чтение будет применяться, только когда менеджер баз данных знает, что оно будет эффективным, например, при сортировках таблиц, просмотрах таблиц или предварительном чтении списков.

Рекомендация: В большинстве случаев для этого параметра надо оставить значение по умолчанию. Отключайте обнаружение последовательного чтения, только если при помощи других мер не удается решить существующие проблемы с производительностью запросов.

Размер предварительной выборки по умолчанию (dft_prefetch_sz)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

UNIX 32 [0 – 32767]

OS/2 и Windows NT
16 [0 – 32767]

Единица измерения Страницы

Параметры, связанные с данным

- "Размер экстента по умолчанию для табличных пространств (dft_extent_sz)" на стр. 417
- "Число серверов ввода/вывода (num_ioservers)" на стр. 413

При создании табличного пространства можно (необязательно) указать PREFETCHSIZE n, где n - число страниц, которое будет читать менеджер баз данных при предварительном чтении. Если размер предварительной выборки в операторе CREATE TABLESPACE не задан, менеджер баз данных использует значение, задаваемое этим параметром.

Дополнительную информацию смотрите в разделе “Предварительная выборка данных в пул буферов” на стр. 269.

Рекомендация: При помощи средств мониторинга системы можно определить, остается ли процессор свободным в ожидании ввода/вывода. Увеличение значения этого параметра может помочь, если для используемых табличных пространств размер предварительной выборки не задан.

Этот параметр задает умолчание для всей базы данных, и может оказаться неподходящим для отдельных табличных пространств в этой базы данных. Например, значение 32 может подходить для табличного пространства с размером экстента 32 страницы, но не годится для табличного пространства с размером экстента 25 страниц. В идеальном случае следует явно задавать размер предварительной выборки для каждого табличного пространства.

Чтобы свести к минимуму число операций ввода/вывода для табличных пространств, использующих размер экстента по умолчанию (*dft_extent_sz*), надо задать этот параметр как делитель или кратное значения параметра*dft_extent_sz*. Например, если значение *dft_extent_sz* - 32, можно задать для *dft_prefetch_sz* значение 16 (делитель 32) или 64 (кратное 32). Когда размер предварительной выборки кратен размеру экстента, менеджер баз данных может выполнять ввод/вывод параллельно, если:

- Экстенты предварительной выборки находятся на разных физических устройствах
- Сконфигурировано несколько серверов ввода/вывода (*num_ioservers*).

Число контейнеров SMS по умолчанию (numsegs)

Тип конфигурации	База данных
Тип параметра	Информационный
Единица измерения	Счетчик

Этот параметр применим только для табличных пространств SMS и задает число контейнеров, которые будут создаваться для табличных пространств по умолчанию. Этот параметр показывает информацию, заданную при создании базы данных (явно или неявно в команде CREATE DATABASE). Оператор CREATE TABLESPACE никак **не** использует этот параметр.

Дополнительную информацию смотрите в главе “Физические каталоги баз данных” книги *Administration Guide: Planning*.

Размер экстента по умолчанию для табличных пространств (dft_extent_sz)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 32 [2 – 256]

Единица измерения Страницы

Параметры, связанные с данным

“Размер предварительной выборки по умолчанию (dft_prefetch_sz)” на стр. 415

При создании табличного пространства можно (необязательно) указать `EXTENTSIZE n`, где `n` - размер экстента. Если вы не указываете размер экстента в операторе CREATE TABLESPACE, менеджер баз данных использует заданное значение этого параметра.

Дополнительную информацию смотрите в главе “Проектирование и выбор табличных пространств” в книге *Administration Guide: Planning*.

Рекомендация: Во многих случаях вы захотите явно задавать размер экстента при создании табличного пространства. Перед тем, как выбирать значение для этого параметра, надо понять, как вы будете явно задавать размер экстента в операторе CREATE TABLESPACE. Дополнительную информацию смотрите в разделе “Влияние табличного пространства на оптимизацию запросов” на стр. 98.

Размер сегмента расширения памяти (estore_seg_sz)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 16000 [0 – 1048575]

Единица измерения Страницы

Параметры, связанные с данным

“Число сегментов расширения памяти (num_estore_segs)” на стр. 418

Этот параметр задает число страниц в каждом сегменте расширения памяти в базе данных. Этот параметр используется только в случае, когда у вашего компьютера больше реальной адресуемой памяти, чем максимальный объем виртуальной адресуемой памяти.

Рекомендация: Этот параметр применяется, только если доступно расширение памяти; он используется вместе с параметром *num_estore_segs*. Задавая число

страниц, используемых в каждом сегменте расширения памяти, стоит также рассмотреть число сегментов расширения памяти (посмотрев и изменив значение параметра *num_estore_segs*). Дополнительную информацию о расширении памяти смотрите в разделе “Расширение памяти” на стр. 296.

Число сегментов расширения памяти (num_estore_segs)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 0 [0 – 2147483647]

Параметры, связанные с данным

“Размер сегмента расширения памяти (estore_seg_sz)” на стр. 417

Этот параметр задает число сегментов расширения памяти, доступных для базы данных.

По умолчанию сегменты расширения памяти не используются.

Рекомендация: Используйте этот параметр для задания использования сегментов расширения памяти, только если объем памяти на используемой платформе превышает максимальное адресное пространство и желательно использовать эту память. Задавая число сегментов, следует также рассмотреть размер каждого сегмента (посмотрев и изменив значение параметра *estore_seg_sz*).

Если заданы оба параметра конфигурации *num_estore_segs* и *estore_seg_sz*, нужно с помощью операторов CREATE/ALTER BUFFERPOOL задать пулы буферов, которые будут использоваться для расширения памяти. Дополнительную информацию о расширении памяти смотрите в разделе “Расширение памяти” на стр. 296.

Агенты

Следующие параметры могут влиять на число программ, которые могут одновременно выполняться при достижении оптимальной производительности:

- “Максимальное число активных прикладных программ (maxappls)” на стр. 419
- “Среднее число активных программ (avg_appls)” на стр. 420
- “Максимальное число файлов баз данных, открытых для программы (maxfilop)” на стр. 421
- “Максимальное число открытых файлов (maxtotfilop)” на стр. 422
- “Приоритет агентов (agentpri)” на стр. 423
- “Максимальное число агентов (maxagents)” на стр. 425
- “Максимальное число одновременных агентов (maxcagents)” на стр. 426

- “Максимальное число координирующих агентов (max_coordagents)” на стр. 427
- “Максимальное число логических агентов (max_logicagents)” на стр. 428
- “Размер пула агентов (num_poolagents)” на стр. 428
- “Начальное число агентов в пуле (num_initagents)” на стр. 430

Максимальное число активных прикладных программ (maxappls)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

UNIX 40 [1 – 60000]

OS/2 и Windows NT, Сервер баз данных с локальными и удаленными клиентами 20 [1 – 60000]

OS/2 и Windows NT, Сервер баз данных с локальными клиентами 10 [1 – 60000]

Единица измерения Счетчик

Параметры, связанные с данным

- “Максимальное число агентов (maxagents)” на стр. 425
- “Максимальное число координирующих агентов (max_coordagents)” на стр. 427
- “Максимальный процент списка блокировок перед расширением (maxlocks)” на стр. 407
- “Максимальная память для списка блокировок (locklist)” на стр. 375
- “Среднее число активных программ (avg_appls)” на стр. 420

Этот параметр задает максимальное число прикладных программ (локальных и удаленных), одновременно соединенных с базой данных. Поскольку для каждой прикладной программы, подсоединяющейся к базе данных, надо выделить некоторую собственную память, много одновременно работающих программ могут потребовать большого объема памяти.

Значение этого параметра должно быть не меньше суммы числа соединенных с базой данных программ и числа тех из них, которые могут одновременно участвовать в процессе выполнения двухфазного принятия или отката. Добавьте затем к этой сумме ожидаемое число неоднозначных транзакций в любой данный момент времени. Дополнительную информацию по неоднозначным

транзакциям смотрите в разделе “Восстановление после ошибок в ходе двухфазного принятия” книги *Administration Guide: Planning*.

Если программа пытается соединиться с базой данных, но значение *maxappls* уже достигнуто, программе возвращается сообщение об ошибке, указывающее, что с базой данных уже соединено максимально возможное число программ.

Чем больше программ используют менеджер связей данных, тем выше должно быть значение *maxappls*. Рассчитайте нужное значение по следующей формуле:

```
<maxappls> = 5 * (число узлов) + (наибольшее число активных программ,
                  использующих менеджер связей данных)
```

Максимальное поддерживаемое значение для менеджера связей данных - 2000.

В среде многораздельных баз данных это максимальное число программ, которые могут быть одновременно активны по отношению к разделу базы данных. Этот параметр ограничивает число активных прикладных программ, работающих с разделом базы данных на сервере разделов баз данных, независимо от того, является ли этот сервер узлом координатора или нет. Узел каталога в среде многораздельных баз данных требует более высокого значения *maxappls*, чем при других типах сред, так как в среде многораздельных баз данных каждой программе требуется соединяться с узлом каталога.

Рекомендация: Увеличивая значение этого параметра без уменьшения значения параметра *maxlocks* или увеличения значения параметра *locklist*, можно в результате вместо предельного числа программ достичь предела по блокировкам для базы данных (*locklist*) и столкнуться с частыми расширениями блокировок.

В определенной степени на максимальное число программ может влиять параметр *maxagents*. Программа может соединиться с базой данных только при наличии доступного соединения (*maxappls*), а также доступного агента (*maxagents*). Кроме того, на максимальное число прикладных программ влияет также параметр конфигурации *max_coordagents*, поскольку если значение *max_coordagents* достигнуто, новые прикладные программы (для которых требуются координирующие агенты) запустить нельзя.

Среднее число активных программ (avg_appls)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 1 [1 – maxappls]

Единица измерения Счетчик

Параметры, связанные с данным

- “Максимальное число активных прикладных программ (maxappls)” на стр. 419

Этот параметр используется оптимизатором SQL для оценки степени доступности пула буферов для выбранного плана доступа во время выполнения.

Рекомендация: Если DB2 работает в многопользовательской среде, в особенности со сложными запросами и большим пулом буферов, вы можете захотеть сообщить оптимизатору SQL, что к системе обращаются с запросами много пользователей, чтобы оптимизатор был осторожнее в своих предположениях о доступности пула буферов.

Задавая этот параметр, надо оценить число прикладных программ со сложными запросами, обычно использующих эту базу данных. Простые программы OLTP в эту оценку включать не надо. Если такую оценку сделать трудно, можно перемножить:

- Среднее число всех программ, работающих с вашей базой данных. Монитор баз данных может дать сведения о числе программ в каждый момент времени; по этим данным вы можете вычислить среднее число программ за некоторый период. Информация от монитора баз данных включает в себя как программы OLTP, так и прочие программы.
- Вашу оценку доли прикладных программ со сложными запросами.

Как и при настройке других параметров конфигурации, влияющих на оптимизатор, меняйте при настройке этот параметр понемногу. Это позволит минимизировать различие выбора путей.

После изменения этого параметра, возможно, следует выполнить повторное связывание прикладных программ (используя команду REBIND PACKAGE).

Максимальное число файлов баз данных, открытых для программы (maxfilop)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

UNIX 64 [2 – 1950]

OS/2 и Windows NT
64 [2 – 32768]

Единица измерения Счетчик

Параметры, связанные с данным

- “Максимальное число открытых файлов (maxtotfilop)” на стр. 422

- “Максимальное число активных прикладных программ (maxappls)” на стр. 419

Этот параметр задает максимальное число хэндлов файлов, которые можно открыть для каждого агента базы данных. Если при открытии файла это значение превышается, закрываются некоторые файлы, используемые этим агентом. Если задано слишком маленькое значение *maxfilop*, расходы на открытие и закрытие файлов (для предотвращения превышения этого максимального значения) станут значительными и могут ухудшить производительность.

При взаимодействии менеджера баз данных с операционной системой файловые контейнеры табличных пространств SMS и табличных пространств DMS воспринимаются как файлы и для них требуются хэндлы файлов. Для табличных пространств SMS обычно используется больше файлов, чем для файловых табличных пространств DMS. Поэтому при использовании табличных пространств SMS для этого параметра нужно задавать большее значение, чем при использовании файловых табличных пространств DMS.

Этот параметр можно также использовать, чтобы гарантировать, что общее число хэндлов, используемых менеджером баз данных, не превосходит максимального их числа для операционной системы, ограничив число хэндлов для каждого агента некоторым числом; это число хэндлов будет зависеть от числа одновременно работающих агентов.

Максимальное число открытых файлов (maxtotfilop)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 16000 [100 – 32768]

Единица измерения Счетчик

Параметры, связанные с данным

“Максимальное число файлов баз данных, открытых для программы (maxfilop)” на стр. 421

Этот параметр задает максимальное число файлов, которые могут быть открыты всеми агентами и другими потоками, выполняемыми в одном экземпляре менеджера баз данных. Если при открытии файла превышается это значение, прикладной программе возвращается код ошибки.

Примечание: Для платформ на базе UNIX этот параметр не применяется.

Рекомендация: При задании этого параметра нужно учесть число хэндлов файлов, которые могут использоваться для каждой базы данных в этом экземпляре менеджера баз данных. Чтобы оценить максимальное значение этого параметра:

1. С помощью следующей формулы вычислите максимальное число хэндлов файлов, которые могут использоваться для каждой базы данных в этом экземпляре:

   ```
   maxappls * maxfilop
   ```

2. Вычислите сумму полученных значений и проверьте, чтобы это значение не превосходило заданного этим параметром максимального значения.

Если создана новая база данных, нужно заново вычислить значение этого параметра.

Приоритет агентов (agentpri)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию[Диапазон]

AIX -1 [41 - 125]

Другие системы UNIX
-1 [41 - 128]

Windows NT
-1 [0 - 6]

OS/2 -1 [200 - 231; 300 - 331; 400 - 431]

Этот параметр управляет приоритетом, который планировщик операционной системы дает всем агентам и другим процессам и потокам экземпляра

менеджера баз данных. В среде многораздельной базы данных сюда также включаются координирующие агенты и подагенты, параллельные системные контроллеры и демоны FCM. Этот приоритет определяет, как процессорное время предоставляется процессам DB2, агентам и потокам относительно других процессов и потоков, запускаемых на данном компьютере. Если для данного параметра устанавливается значение -1, никаких специальных действий не выполняется, и планирование для менеджера баз данных осуществляется так же, как и для других процессов и потоков в операционной системе. Если для данного параметра установить значение, отличное от -1, менеджер баз данных будет создавать свои процессы и потоки со статическим приоритетом, равным значению данного параметра. Тем самым этот параметр позволяет управлять приоритетом, с которым процессы и потоки менеджера баз данных выполняются на данном компьютере.

Этот параметр может использоваться для повышения пропускной способности менеджера баз данных. Значения для настройки этого параметра определяются операционной системой, в которой запускается менеджер баз данных. Например, в среде на основе UNIX низкие числовые значения означают высокие приоритеты. Если для данного параметра установлено значение от 41 до 125, менеджер баз данных создает свои агенты со статическим приоритетом UNIX, равным значению данного параметра. Это важно в средах на основе UNIX, так как низкие числовые значения дают для менеджера баз данных высокие приоритеты, однако при этом другие процессы (включая программные и пользовательские) могут испытывать задержки, поскольку не могут получить необходимое им процессорное время. При настройке этого параметра следует учесть интересы других процессов, выполняемых на данном компьютере.

В среде OS/2 более высокие числовые значения дают более высокие приоритеты.

Рекомендация: Начните со значения по умолчанию. Это значение обеспечивает хороший компромисс между временем ответа для других программ (пользователей) и пропускной способностью менеджера баз данных.

Если для вас важна производительность базы данных, можно воспользоваться методами измерения производительности, чтобы определить для этого параметра оптимальное значение. Увеличивать приоритет диспетчера баз данных надо осторожно, так как производительность других пользовательских процессов может сильно упасть, особенно при очень высокой занятости процессора. Увеличение приоритета процессов и потоков менеджера баз данных может дать существенную выгоду в производительности.

Примечание: Если на платформах на основе UNIX установить для данного параметра значение, отличающееся от значения по умолчанию, нельзя будет управлять приоритетами агентов с помощью программы ограничения ресурсов.

Максимальное число агентов (maxagents)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию[Диапазон] 200 [1 – 64000]

400 [1 – 64000] на сервере многораздельных баз данных с локальными и удаленными клиентами

10 [1 – 64000] на сервере сателлитных баз данных с удаленными клиентами

Единица измерения Счетчик

Параметры, связанные с данным

- “Максимальное число активных прикладных программ (maxappls)” на стр. 419
- “Максимальное число одновременных агентов (maxcagents)” на стр. 426
- “Максимальное число координирующих агентов (max_coordagents)” на стр. 427
- “Максимальное число процессов DARI (maxdari)” на стр. 432
- “Минимальная принятая частная память (min_priv_mem)” на стр. 392
- “Размер пула агентов (num_poolagents)” на стр. 428

Этот параметр задает максимальное число агентов менеджера баз данных (агентов координатора и подагентов), которые в любой заданный момент времени готовы принимать запросы от прикладной программы. Если нужно ограничить число координирующих агентов, используйте параметр *max_coordagents*.

В средах с ограниченным объемом памяти этот параметр может быть полезен для ограничения общего использования памяти менеджером баз данных, поскольку для каждого дополнительного агента требуется дополнительная память.

Рекомендация: Значение *maxagents* не должно быть меньше суммы значений *maxappls* в каждой базе данных, для которой разрешен одновременный доступ. Если число баз данных больше значения параметра *numdb*, безопаснее всего использовать произведение значения *numdb* на наибольшее из значений *maxappls*.

Для каждого дополнительного агента требуются некоторые дополнительные ресурсы, которые выделяются при запуске менеджера баз данных.

Максимальное число одновременных агентов (maxcagents)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] -1 (*max_coordagents*) [-1; 1 – *max_coordagents*]

Единица измерения Счетчик

Параметры, связанные с данным

- "Максимальное число активных прикладных программ (maxappls)" на стр. 419
- "Максимальное число агентов (maxagents)" на стр. 425
- "Максимальное число координирующих агентов (max_coordagents)" на стр. 427

Максимальное число агентов менеджера баз данных, которые могут одновременно выполнять транзакцию менеджера баз данных. Этот параметр используется для управления загрузкой системы в периоды высокой одновременной активности прикладных программ. Например, в системе, которая должна поддерживать большое число соединений и в которой ограничен объем памяти для обслуживания этих соединений. Этот параметр может быть полезен в таких средах, когда в периоды высокой одновременной активности может возникать чрезмерная загрузка операционной системы.

Этот параметр не ограничивает число прикладных программ, которые могут соединяться с базой данных. Он ограничивает только число агентов менеджера

баз данных, одновременно используемых менеджером баз данных, ограничивая тем самым использование системных ресурсов в периоды максимальной нагрузки.

Значение –1 указывает, что максимальное число агентов равно значению параметра *max_coordagents*.

Рекомендация: В большинстве случаев для этого параметра можно использовать значение по умолчанию. Если большая степень одновременности прикладных программ вызывает ошибки, можно использовать измерение производительности, чтобы настроить этот параметр для оптимизации производительности базы данных.

Максимальное число координирующих агентов (max_coordagents)

Тип конфигурации
Менеджер баз данных

Применяется к
- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра
Конфигурируемый

По умолчанию [Диапазон]
-1 (*maxagents – num_initagents*)

[-1, 0 – *maxagents*]

Для сред многораздельных баз данных и для сред, где *intra_parallel* имеет значение Yes, значение по умолчанию равно *maxagents* минус *num_initagents*; иначе значение по умолчанию равно *maxagents*. Это означает, что в среде одноразделных баз данных *max_coordagents* всегда равен *maxagents*, если только система не сконфигурирована для внутриразделного параллелизма.

Если у вас среда одноразделных баз данных и параметр *intra_parallel* не включен, *max_coordagents* должен быть равен *maxagents*.

Параметры, связанные с данным
- “Начальное число агентов в пуле (num_initagents)” на стр. 430

- “Размер пула агентов (num_poolagents)”
- “Максимальное число агентов (maxagents)” на стр. 425
- “Разрешение внутриразделного параллелизма (intra_parallel)” на стр. 494

Этот параметр задает максимальное число координирующих агентов, которые могут одновременно существовать на сервере в среде многораздельных или одноразделных баз данных.

По одному координирующему агенту запрашивается для каждой локальной или удаленной программы, которая соединяется с базой данных или подключается к экземпляру. Подключения к экземпляру требуют команды CREATE DATABASE, DROP DATABASE и команды монитора систем баз данных.

Максимальное число логических агентов (max_logicagents)

Тип конфигурации Менеджер баз данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] -1 (*max_coordagents*) [-1; *max_coordagents* – 64000]

Этот параметр определяет максимальное число прикладных программ, которые могут соединиться с экземпляром. Обычно каждой программе назначается агент координатора. Агент облегчает операции между программой и базой данных. Когда используется значение этого параметра по умолчанию, возможность концентратора не активируется. В результате каждый агент работает в своей памяти, и все агенты совместно используют глобальные ресурсы менеджера баз данных и базы данных, в частности, пул буферов. Если для этого параметра задано значение больше значения по умолчанию, активируется возможность концентратора. Использование концентратора позволяет снизить ресурсы сервера, необходимый клиентской программе, до такого уровня, что шлюз DB2 Connect может обслуживать более 10000 клиентских соединений.

Дополнительную информацию и примеры использования DB2 Connect в качестве концентратора поддержки транзакций XA смотрите в руководстве *DB2 Connect. Руководство пользователя*.

Значение -1 указывает, что предельное значение равно *max_coordagents*.

Размер пула агентов (num_poolagents)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами

- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра

Конфигурируемый

По умолчанию [Диапазон]

-1 [-1, 0 — *maxagents*]

По умолчанию для сервера с одноразделной базой данных и локальными клиентами это значение - максимум из *maxagents*/50 и *max_querydegree*.

Для сервера с одноразделной базой данных и локальными и удаленными клиентами это значение по умолчанию - максимум из *maxagents*/50 x *max_querydegree* и *maxagents* - *max_coordagents*.

По умолчанию это значение для сервера раздела базы данных равно большему из значений *maxagents*/10 x *max_querydegree* и *maxagents* - *max_coordagents*.

Параметры, связанные с данным

- “Начальное число агентов в пуле (num_initagents)” на стр. 430
- “Максимальное число агентов (maxagents)” на стр. 425
- “Максимальная степень параллелизма запросов (max_querydegree)” на стр. 493
- “Максимальное число координирующих агентов (max_coordagents)” на стр. 427

Этот параметр задает максимальный размер пула агентов (он заменяет параметр *max_idleagents*, который использовался в DB2 Версии 2).

В пуле агентов находятся подагенты и незанятые агенты. Незанятые агенты могут использоваться как параллельные подагенты и как подагенты координатора. Если создается больше агентов, чем значение этого параметра, они прекращают работать, как только выполнят свои текущие требования, и не возвращаются в пул.

Если значение этого параметра - 0, агенты создаются по необходимости, и работа их прекращается, как только они выполнят свои текущие требования.

Если значение равно *maxagents*, а пул заполнен связанными подагентами, сервер не сможет использоваться как узел координатора, так как нельзя будет создать новые агенты координатора.

Рекомендация: Если вы запускаете среду поддержки решений с небольшим числом параллельно соединяющихся программ, установите для параметра *num_poolagents* небольшое значение, чтобы избежать большого количества наличия незанятых агентов в пуле.

При работе в среде обработки транзакций, в которой параллельно соединяется большое число программ, увеличьте значение параметра *num_poolagents*, чтобы избежать затрат, связанных с частым созданием и уничтожением агентов.

Начальное число агентов в пуле (num_initagents)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 0 [0 – *num_poolagents*]

Параметры, связанные с данным

- "Максимальное число агентов (maxagents)" на стр. 425
- "Размер пула агентов (num_poolagents)" на стр. 428
- "Максимальное число координирующих агентов (max_coordagents)" на стр. 427

Этот параметр задает начальное число незанятых агентов, которые создаются в пуле агентов при запуске DB2.

Хранимые процедуры (DARI)

Следующие параметры могут влиять на прикладные программы DARI (Database Application Remote Interface - удаленный интерфейс прикладных программ баз данных):

- "Индикатор сохранения процесса DARI (keepdari)" на стр. 431
- "Максимальное число процессов DARI (maxdari)" на стр. 432
- "Инициализировать процессы DARI в JVM (initdari_jvm)" на стр. 433

- “Начальное число изолированных процессов DARI в пуле (num_initdaris)” на стр. 434

Примечание: Термин DARI - синоним хранимой процедуры.

Индикатор сохранения процесса DARI (keepdari)

Тип конфигурации Менеджер баз данных

Применяется к
- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] Yes [Yes; No]

Параметры, связанные с данным

“Максимальное число процессов DARI (maxdari)” на стр. 432

Этот параметр задает, сохранять ли процесс DARI после завершения вызова DARI. Процессы DARI создаются как отдельные системные объекты с целью изолировать написанный пользователем код DARI от процесса агента менеджера баз данных. Этот параметр применяется только для серверов баз данных.

Если значение *keepdari* - *no*, новые процессы DARI создаются для каждого вызова DARI, а затем уничтожаются. Если значение *keepdari* - *yes*, процессы DARI используются для последующих вызовов DARI. При остановке менеджера баз данных все незавершенные процессы DARI будут прерваны.

Значение *yes* для этого параметра ведет к потреблению менеджером баз данных дополнительных системных ресурсов для каждого активируемого процесса DARI вплоть до значения, определяемого параметром *maxdari*. Однако такой рост происходи только при отсутствии доступного процесса DARI для обработки очередного вызова DARI. Этот параметр игнорируется, если задано значение *maxdari*, равное 0.

Рекомендация: В среде, где число требований DARI велико по сравнению с прочими требованиями, а ресурсов достаточно, для этого параметра можно задать значение *yes*. Это улучшит производительность DARI за счет исключения расходов на создании исходного процесса DARI, поскольку для обработки вызова будет использован существующий процесс DARI.

Например, в программе OLTP обработки банковских операций со счетами код для выполнения каждой транзакции можно быть оформлен в виде хранимой процедуры, выполняющей процесс DARI. В этой программе основную рабочую нагрузку выполняют процессы DARI. Если задано значение параметра *no*, каждая транзакция вызовет дополнительные расходы на создание нового процесса DARI, что приведет к значительному ухудшению производительности. Если же задано значение *yes*, каждая транзакция будет пытаться использовать существующий процесс DARI, что позволяет избежать этих расходов.

Максимальное число процессов DARI (maxdari)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] -1 (*max_coordagents*) [-1; 0 – *max_coordagents*]

Единица измерения Счетчик

Параметры, связанные с данным

- "Максимальное число агентов (maxagents)" на стр. 425
- "Индикатор сохранения процесса DARI (keepdari)" на стр. 431
- "Начальное число изолированных процессов DARI в пуле (num_initdaris)" на стр. 434
- "Максимальное число координирующих агентов (max_coordagents)" на стр. 427

Этот параметр задает максимальное число процессов DARI на сервере базы данных. Когда этот предел достигнут, новые требования DARI не могут выполняться. Этот параметр применяется только для серверов баз данных.

Для каждого координирующего агента может быть активным только один процесс DARI, поэтому максимальное число процессов DARI ограничивается также максимальным числом координирующих агентов (*max_coordagents*).

Рекомендация: Если в вашей среде используется средство DARI с менеджером баз данных, этот параметр можно использовать для обеспечения необходимого

числа процессов DARI для обработки вызовов DARI, сделанных в любой момент времени в менеджере баз данных.

Если этот параметр имеет значение –1, максимальное число процессов DARI будет равно значению параметра *max_coordagents*.

Если оказывается, что значение по умолчанию не подходит для вашей среды, так как на процессы DARI затрачивается слишком много системных ресурсов и это влияет на производительность менеджера баз данных, настройку этого параметра можно начать с такого значения:

```
maxdari = число прикладных программ, которые могут одновременно делать вызовы DARI
```

Если параметр *keepdari* имеет значение *YES*, каждый созданный процесс DARI будет продолжать существовать и использовать системные ресурсы даже после завершения обработки вызова DARI и возврата к агенту.

Если в вашей среде сильно ограничен объем ресурсов и вы не можете позволить использование ресурсов процессами DARI, можно запретить DARI, задав для этого параметра значение ноль (0).

Инициализировать процессы DARI в JVM (initdari_jvm)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] No [Yes; No]

Параметры, связанные с данным

- “Максимальное число процессов DARI (maxdari)” на стр. 432
- “Начальное число изолированных процессов DARI в пуле (num_initdaris)” на стр. 434
- “Индикатор сохранения процесса DARI (keepdari)” на стр. 431

Этот параметр задает, будет ли любой изолированный процесс DARI загружать при запуске виртуальную Java-машину. Задание этого параметра сокращает время начального запуска для изолированных хранимых процедур, особенно

когда он применяется в сочетании с параметром *num_initdaris*. Этот параметр может увеличить время начальной загрузки для изолированных хранимых процедур, написанных не на языке Java, поскольку для них JVM не нужна.

Начальное число изолированных процессов DARI в пуле (num_initdaris)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 0 [0 – *maxdari*]

Параметры, связанные с данным

- “Максимальное число процессов DARI (maxdari)” на стр. 432
- “Инициализировать процессы DARI в JVM (initdari_jvm)” на стр. 433
- “Индикатор сохранения процесса DARI (keepdari)” на стр. 431

Этот параметр задает начальное число незанятых изолированных процессов DARI, которые создаются в пуле DARI при запуске DB2. Задание этого параметра сокращает время начального запуска для изолированных хранимых процедур. Этот параметр игнорируется, если не задано *keepdari*.

Ведение журнала и восстановление

Восстановление среды имеет большое значение для предотвращения потерь важных данных. Ряд параметров служит для управления средой и помогает обеспечить правильное восстановление данных и транзакций. Эти параметры разделены на следующие категории:

- “Файлы журналов баз данных” на стр. 435
- “Активность журналов баз данных” на стр. 441
- “Восстановление” на стр. 447
- “Восстановление распределенной единицы работы” на стр. 453.

Файлы журналов баз данных

Следующие параметры содержат информацию о числе, размере и состоянии файлов, используемых для ведения журналов баз данных:

- “Размер файлов журнала (logfilsiz)”
- “Число первичных файлов журнала (logprimary)” на стр. 436
- “Число вторичных файлов журнала (logsecond)” на стр. 438
- “Изменить путь журнала базы данных (newlogpath)” на стр. 439
- “Положение файлов журналов (logpath)” на стр. 441
- “Первый активный файл журнала (loghead)” на стр. 441

Размер файлов журнала (logfilsiz)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

UNIX	1000 [4 – 65535]
Windows NT	250 [4 – 65535]
OS/2	250 [4 – 65535]

Единица измерения Страницы (4 Кбайта)

Параметры, связанные с данным

- “Число первичных файлов журнала (logprimary)” на стр. 436
- “Число вторичных файлов журнала (logsecond)” на стр. 438
- “Диапазон восстановления и интервал мягких контрольных точек (softmax)” на стр. 443

Этот параметр определяет размер каждого первичного и вторичного файла журнала. Размер этих файлов определяет число записей, которые могут быть занесены в журнал до момента его заполнения и открытия нового журнала.

Использование первичных и вторичных файлов журнала и действия, предпринимаемые при заполнении журнала, зависят от типа выполняемой записи в журнал:

- Циклическая запись

  Первичный файл журнала можно использовать повторно, если записанные в него изменения были приняты. Если размер файла журнала невелик, а прикладные программы обработали много изменений в базе данных без принятия этих изменений, первичный файл журнала может быстро

заполниться. При заполнении всех первичных файлов журнала менеджер баз данных разместит вторичные файлы журнала для хранения новых записей.

- Регистрация хранения журналов

  При заполнении первичного файла журнала журнал архивируется, и размещается новый первичный файл журнала.

Рекомендация: Надо найти компромисс между размером файлов журнала и числом первичных файлов журнала:

- Если для базы данных выполняется большое число транзакций изменения, удаления и/или вставки, из-за чего файл журнала очень быстро заполняется, значение *logfilsiz* надо увеличить.

  Примечание: Общее ограничение на размер файлов журнала составляет 32 Гбайта. Это означает, что число файлов журнала (*logprimary* + *logsecond*), умноженное на размер каждого файла в байтах (*logfilsiz* * 4096), не должно превышать 32 Гбайт.

  Слишком маленький размер файла журнала может снизить производительность системы из-за ненужных затрат на архивирование старых файлов журнала, размещение новых файлов и ожидание пригодного для использования файла журнала.
- При недостатке места на диске значение *logfilsiz* следует уменьшить, так как производится предварительное размещение первичных журналов с заданным размером.

  При слишком большом размере файла журнала может снизиться гибкость управления архивированными файлами журнала и копиями файлов журнала, так как на некоторые носители не удастся записать файл журнала целиком.

Если используется запись с сохранением, текущий активный файл журнала закрывается и усекается при отсоединении от базы данных последней прикладной программы. При очередном соединении с базой данных используется уже следующий файл журнала. Поэтому если вы точно знаете требования ваших одновременно используемых прикладных программ к записи журнала, можно определить размер файла журнала, при котором не будет выделяться лишнее дисковое пространство.

Дополнительную информацию об этом параметре смотрите в разделе "Параметры конфигурации для записи журнала базы данных" в книге *Administration Guide: Implementation*.

Число первичных файлов журнала (logprimary)

Тип конфигурации	База данных
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	3 [2 – 128]

Единица измерения Счетчик

Когда размещается

- При создании базы данных
- При перемещении журнала в другое место (что происходит при изменении параметра *logpath*)
- При увеличении значения этого параметра (*logprimary*) во время следующего соединения с базой данных после отсоединения всех пользователей
- При архивировании файла журнала и размещении нового файла (если разрешены *logretain* или *userexit*)
- При изменении параметра *logfilsiz* во время следующего соединения с базой данных и изменении размер активных файлов журнала после отсоединения всех пользователей.

Когда освобождается Не освобождается, если только значение этого параметра не уменьшается. Если значение уменьшается, ненужные файлы журнала удаляются при следующем соединении с базой данных.

Параметры, связанные с данным

- “Размер файлов журнала (logfilsiz)” на стр. 435
- “Число вторичных файлов журнала (logsecond)” на стр. 438
- “Разрешение хранения журнала (logretain)” на стр. 445
- “Разрешение обработчика пользователя (userexit)” на стр. 446

Первичные файлы журналов устанавливают определенный объем памяти, отводимой для файлов журналов восстановления. Этот параметр позволяет задавать количество предварительно размещаемых первичных файлов журнала.

При циклической записи первичные журналы используются несколько раз по очереди. Поэтому, если журнал заполнен, используется следующий первичный журнал в последовательности, если он доступен. Журнал считается доступным, если для всех записанных в нем единиц работы выполнено принятие или откат. Если следующий первичный журнал в последовательности недоступен, размещается и используется вторичный журнал. Дополнительные вторичные журналы размещаются и используются до тех пор, пока не станет доступен

следующий первичный журнал в последовательности или не будет достигнут предел, установленный параметром *logsecond*. Эти вторичные файлы журнала динамически освобождаются, так как они больше не требуются менеджеру баз данных.

Число первичных и вторичных файлов журнала должно удовлетворять следующему неравенству:
- (*logprimary* + *logsecond*) <= 128

Рекомендация: Выбор значения для этого параметра зависит от ряда факторов, включая используемый тип записи, размер файлов журнала и тип среды обработки (например, длину транзакций и частоту принятий).

Увеличение этого значения приводит к возрастанию требований к месту на диске для журналов, так как первичные файлы журнала предварительно размещаются при самом первом соединении с базой данных.

Обнаружив, что часто размещаются вторичные файлы журнала, можно попытаться улучшить производительность системы, увеличив размера файлов журнала (*logfilsiz*) или число первичных файлов журнала.

Для баз данных, к которым обращаются редко, для экономии места на диске задайте значение этого параметра 2. Для баз данных, где поддерживается восстановление с повтором транзакций, задайте более высокое значение этого параметра, чтобы избежать излишних затрат на частое размещение новых журналов.

Управлять размером первичных файлов журнала можно с помощью системного монитора баз данных.

Дополнительную информацию смотрите в описании элементов монитора в книге *System Monitor Guide and Reference*.
- *sec_log_used_top (максимальное используемое вторичное пространство журнала)*
- *tot_log_used_top (максимальное используемое общее пространство журнала)*
- *sec_logs_allocated (число размещенных на данный момент вторичных журналов)*

Наблюдения за этими значениями монитора за некоторый промежуток времени полезны при решении вопросов настройки, так как средние значения могут лучше соответствовать вашим долгосрочным требованиям.

Число вторичных файлов журнала (logsecond)

Тип конфигурации	База данных
Тип параметра	Конфигурируемый

По умолчанию [Диапазон] 2 [0 – 126]

Единица измерения Счетчик

Когда размещается При необходимости, когда *logprimary* недостаточен (смотрите подробности ниже)

Когда освобождается Когда менеджер баз данных определяет, что он более не требуется.

Параметры, связанные с данным

- “Размер файлов журнала (logfilsiz)” на стр. 435
- “Число первичных файлов журнала (logprimary)” на стр. 436
- “Разрешение хранения журнала (logretain)” на стр. 445
- “Разрешение обработчика пользователя (userexit)” на стр. 446

Этот параметр задает число вторичных файлов журнала, которые создаются и используются для файлов журнала восстановления (только при необходимости). Когда первичные файлы журнала заполняются, по мере необходимости по одному выделяются вторичные файлы журнала (с размером, указанным *logfilsiz*) вплоть до максимального числа, задаваемого этим параметром. Если вторичных файлов журнала требуется больше, чем разрешено этим параметром, прикладной программе будет возвращена ошибка, и база данных будет закрыта.

Более подробно о том, как используются вторичные файлы журнала, смотрите в разделе “Число первичных файлов журнала (logprimary)” на стр. 436.

Рекомендация: Используйте вторичные файлы журнала для баз данных, которым периодически требуются большие объемы пространства журнала. Например, если для программы, запускаемой раз в месяц, может потребоваться больше пространства журнала, чем отведено для первичных файлов журнала. Поскольку вторичные файлы журнала не требуют места постоянно, в ситуациях такого типа их применение может оказаться выгодным.

Изменить путь журнала базы данных (newlogpath)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] Пусто [любой допустимый путь или устройство]

Параметры, связанные с данным

- “Положение файлов журналов (logpath)” на стр. 441

• “Согласованность базы данных (database_consistent)” на стр. 465

В этом параметре можно задать строку длиной до 242 байт, чтобы изменить место хранения файлов журнала. Строка может указывать как на путь, так и на непосредственное устройство. Если в строка задается путь, надо использовать полное, а не относительное имя пути.

Примечание: В среде многораздельных баз данных к пути автоматически добавляется номер узла. Это делается для сохранения единого пути в нескольких конфигурациях логических узлов.

Чтобы задать устройство, введите строку, которую операционная система распознает в качестве устройства. Например:

- В Windows NT - `\\.\d:` или `\\.\PhysicalDisk5`

  Примечание: Чтобы записывать журналы на устройство, необходимо иметь Windows NT Версии 4.0 с установленным Service Pack 3 или более поздним.

- Для платформ на основе UNIX - `/dev/rdblog8`

Примечание: Устройство можно задавать только для платформ AIX, Windows 2000, Windows NT, Solaris, HP-UX, NUMA-Q и Linux.

Новое параметр не станет значением *logpath*, пока не будут выполнены оба следующих условия:

- База данных находится в согласованном состоянии, что определяется значением параметра *database_consistent*.
- От базы данных отключены все пользователи

Когда с базой данных будет произведено первое соединение, менеджер баз данных переместит журналы в место, заданное параметром *logpath*.

На прежнем месте могут остаться файлы журнала. Эти файлы могут быть еще не заархивированы. Их может понадобиться архивировать вручную. Кроме того, если вы запускаете репликацию для этой базы данных, для репликации могут понадобиться файлы журнала, записанные до изменения пути. Если база данных сконфигурирована с возможностью использования обработчика пользователя (параметр конфигурации базы данных *userexit* имеет значение Yes), и если все файлы журнала заархивированы (автоматически DB2 или же вручную), DB2 сможет извлечь эти файлы журнала и выполнить процесс репликации. В противном случае можно скопировать эти файлы из старого пути в новый.

Рекомендация: В идеале файлы журнала должны находиться на физическом диске, с которым **не** выполняется интенсивных операций ввода-вывода. В

частности, не стоит помещать файлы журнала на один диск с операционной системой или большой базой данных. Это позволит эффективно вести запись при минимальных расходах, таких как ожидание ввода-вывода.

Можно использовать системный монитор баз данных, чтобы определить число операций ввода-вывода, связанных с записью журнала.

Дополнительную информацию смотрите в описании следующих элементов монитора в книге *System Monitor Guide and Reference*:
- *log_reads* (число прочитанных страниц журнала)
- *log_writes* (число записанных страниц журнала).

Эти элементы данных возвращают показатели активности ввода/вывода, связанной с записью журнала. Можно воспользоваться средствами монитора операционной системы для сбора информации о других дисковых операциях ввода-вывода, а затем сравнить полученные для этих видов операций результаты.

Положение файлов журналов (logpath)

Тип конфигурации База данных

Тип параметра Информационный

Параметры, связанные с данным

“Изменить путь журнала базы данных (newlogpath)” на стр. 439

Этот параметр содержит текущий путь для журналов регистрации. Нельзя напрямую изменить этот параметр, так как он задается менеджером баз данных после вступления в силу изменения значения параметра *newlogpath*.

После создания базы данных для нее создается файл журнала восстановления в подкаталоге каталога, содержащего эту базу данных. По умолчанию используется подкаталог с именем SQLOGDIR в каталоге, созданном для этой базы данных.

Первый активный файл журнала (loghead)

Тип конфигурации База данных

Тип параметра Информационный

Этот параметр содержит имя текущего активного файла журнала.

Активность журналов баз данных

Следующие параметры могут влиять на тип и производительность ведения журналов баз данных:

- “Число принятий для группировки (mincommit)” на стр. 442

- “Диапазон восстановления и интервал мягких контрольных точек (softmax)” на стр. 443
- “Разрешение хранения журнала (logretain)” на стр. 445
- “Разрешение обработчика пользователя (userexit)” на стр. 446

Число принятий для группировки (mincommit)

Тип конфигурации	База данных
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	1 [1 – 25]
Единица измерения	Счетчик

Этот параметр позволяет откладывать помещение записей журнала на диск до выполнения минимального числа принятий. Такая задержка позволяет сократить расходы менеджера баз данных, связанные с записью в журнал. Это улучшит производительность, когда несколько прикладных программ требуют много принятий за очень короткий отрезок времени.

Такая группировка принятий будет происходить только тогда, когда значение этого параметра больше 1 и число связанных с базой данных прикладных программ не меньше значения этого параметра. При группировке принятий выполнение требований прикладных программ на принятия откладывается либо до истечения одной секунды, либо до того времени, когда число требований на принятие сравняется со значением этого параметра (если это произойдет раньше).

Изменения этого параметра вступают в силу немедленно; для этого не надо ждать отсоединения всех программ от базы данных.

Рекомендация: Увеличьте значение этого параметра выше значения по умолчанию, если несколько программ чтения и записи обычно одновременно требуют принятия в базе данных. Это позволит эффективнее использовать ввод/вывод, так как операции будут проводиться реже, записывая за один раз больше записей журнала.

Можно также определить число транзакций в секунду и настроить этот параметр в соответствии с пиковым значение числа транзакций в секунду (или некоторым процентом от него). Расчет на пиковую активность минимизирует расходы на запись журнала в периоды интенсивной нагрузки.

Если вы увеличиваете *mincommit*, может также потребоваться увеличить параметр *logbufsz*, чтобы в период интенсивной нагрузки запись буфера не происходила из-за его переполнения. В таком случае *logbufsz* надо задать как:

```
mincommit * (используемое пространство журнала на транзакцию в среднем)
```

Настроить этот параметр можно с помощью системного монитора баз данных:

- Определить число транзакций в секунду для пиковой нагрузки:

  Получив снимки монитора за типичный день, можно выделить периоды тяжелой нагрузки. Полное число транзакций можно получить, добавив:
  - *commit_sql_stmts (число попыток для операторов commit)*
  - *rollback_sql_stmts (число попыток для операторов rollback)*

  При помощи этой информации и отметок времени можно рассчитать число транзакций в секунду.
- Расчет пространства журнала на одну транзакцию:

  По снимкам за некоторый период времени и число транзакций, можно рассчитать средний размер пространства журнала по следующему элементу монитора:
  - *log_space_used (пространство журнала, используемое единицей работы)*

  Дополнительную информацию о мониторе баз данных смотрите в книге *System Monitor Guide and Reference*.

Диапазон восстановления и интервал мягких контрольных точек (softmax)

Тип конфигурации — База данных

Тип параметра — Конфигурируемый

По умолчанию [Диапазон] — 100 [1 – 100 * *logprimary*]

Единица измерения — Проценты от размера одного первичного файла журнала

Параметры, связанные с данным

- “Размер файлов журнала (logfilsiz)” на стр. 435
- “Число первичных файлов журнала (logprimary)” на стр. 436

Назначение этого параметра:

- Управление числом журналов, необходимых для восстановления после аварии (например, после отключения питания). Например, если используется значение по умолчанию, менеджер баз данных будет пытаться оставить один журналов для восстановления. Если вы задаете значение параметра 300, менеджер баз данных будет пытаться сохранить 3 журнала для восстановления.

  Чтобы управлять числом журналов, необходимых для восстановления после аварии, менеджер баз данных использует этот параметр для запуска чистильщиков страниц и записи на диск тех страниц, которые старше заданного интервала восстановления.
- Определения частоты мягких контрольных точек.

Когда в работе базы данных происходит сбой, например, при отключении питания, с базой данных могут произойти изменения, которые :
- Не приняты, однако информация в пуле буферов была изменена
- Приняты, но не записаны с пула буферов на диск
- Приняты и записаны с пула буферов на диск.

При перезапуске базы данных файлы журналов используются для восстановления после аварии базы данных; это восстановление обеспечивает согласованное состояние базы данных (то есть все принятые транзакции применены к базе данных, а все непринятые транзакции - не применены).

Чтобы определить, какие записи из файла журнала нужно применить к базе данных, менеджер баз данных использует управляющий файл журнала. Этот файл периодически записывается на диск, и в зависимости от того, насколько часто это делается, менеджер баз данных может применять записи журнала для принятых транзакций или записи журнала, описывающие изменения, которые уже были записаны с пула буферов на диск. Эти записи журнала никак не влияют на базу данных, и их применение влечет за собой некоторые расходы на процесс перезапуска баз данных.

Управляющий файл журнала всегда записывается на диск, когда файл журнала заполнен, а также в мягких контрольных точках. Данный параметр конфигурации позволяет задать дополнительные мягкие контрольные точки.

Временные характеристики мягких контрольных точек основаны на различии между “текущим состоянием” и “записанным состоянием”, выраженным в процентах от *logfilsiz*. “Записанное состояние” определяется по самой ранней допустимой записи журнала, указанной в управляющем файле журнала на диске, а “текущее состояние” определяется по информации управления журналом в памяти. (Самая ранняя допустимая запись журнала - это первая запись журнала, читаемая при процессе восстановления.) Мягкая контрольная точка выполняется, если значение, полученное по следующей формуле, будет не меньше значения данного параметра:

```
( (разность между текущим и записанным состояниями) / logfilsiz ) * 100
```

Рекомендация: Можно увеличить или сократить значение этого параметра в зависимости от размера приемлемого окна восстановления (если он больше или меньше размера файла журнала). Уменьшение значения этого параметра приведет к тому, что менеджер баз данных будет чаще запускать чистильщики страниц и чаще выполнять мягкие контрольные точки. Эти действия могут сократить число необходимых и избыточных записей журнала, которые обрабатываются во время восстановления.

Однако отметьте, что увеличение числа триггеров очистки страниц и мягких контрольных точек увеличивают расходы, связанные с записью в журнал базы данных, что может также повлиять на производительность менеджера баз

данных. Кроме того, увеличение частоты мягких контрольных точек может не сократить время, требуемое для перезапуска базы данных, если у вас:

- Очень длинные транзакции с очень немногими точками принятия.
- Очень большой пул буферов и страницы, содержащие принятые транзакции, записываются на диск редко. (Заметим, что избежать этой ситуации можно путем использования асинхронных чистильщиков страниц. Смотрите раздел “Число асинхронных чистильщиков страниц (num_iocleaners)” на стр. 412.)

В обоих этих случаях хранимая в памяти управляющая информация журнала меняется редко, и частая ее запись на диск, если она не была изменена, не даст преимуществ.

Разрешение хранения журнала (logretain)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] No [Recovery; Capture; No]

Параметры, связанные с данным

- “Разрешение обработчика пользователя (userexit)” на стр. 446
- “Индикатор состояния сохранения журналов (log_retain_status)” на стр. 466
- “Индикатор отложенного резервного копирования (backup_pending)” на стр. 465

Используются следующие значения:

- No - указывает, что журналы не хранятся.
- Recovery - задает, что журналы хранятся и могут использоваться для восстановления с повтором транзакций.Кроме того, при использовании репликации данных программа Capture может записывать изменения из журналов в таблицу изменений.
- Capture - задает, что журналы хранятся только для того, чтобы программа Capture могла записывать изменения в таблицу изменений. Эти журналы могут использоваться для восстановления с повтором транзакций, если они не были сокращены после их использования программой репликации данных Capture.

Если *logretain* имеет значение Recovery или *userexit* имеет значение Yes, файлы активных журналов будут сохранены и станут оперативными архивными файлами журналов для использования в восстановлении с повтором транзакций. Это называется регистрацией хранения журналов.

Если для *logretain* задается значение Recovery или для *userexit* задается значение Yes (или оба они), нужно сделать полную резервную копию базы данных. На это состояние укажет параметр флага *backup_pending*.

Если и для *logretain*, и для *userexit* заданы значения No, восстановление с повтором транзакций для базы данных невозможно.

Если для *logretain* задается значение Capture, программа репликации данных Capture вызывает после завершения работы команду PRUNE LOGFILE, чтобы удалить файлы журналов. Если вы собираетесь выполнять для базы данных восстановление с повтором транзакций, не следует задавать значение Capture для *logretain*.

Если и для *logretain*, и для *userexit* заданы значения No, журналы не хранятся. В этой ситуации менеджер баз данных удаляет все журналы в каталоге *logpath* (включая и оперативные файлы архивных журналов), выделяет новые файлы активных журналов и возвращается к циклической записи.

Разрешение обработчика пользователя (userexit)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] No [Yes; No]

Параметры, связанные с данным

- “Разрешение хранения журнала (logretain)” на стр. 445
- “Индикатор состояния обработчика пользователя (user_exit_status)” на стр. 466
- “Индикатор отложенного резервного копирования (backup_pending)” на стр. 465

Если этот параметр включен, запись журналов с сохранением выполняется независимо от значения параметра *logretain*. Этот параметр также указывает, что для архивации и восстановления файлов журнала должен использоваться обработчик пользователя. Файлы журналов архивируются, когда менеджер баз данных закрывает файл журнала. Они извлекаются из архива, когда становятся нужны утилите ROLLFORWARD для восстановления базы данных.

Если включаются *logretain*, *userexit* или оба этих параметра, надо сделать полную копию базы данных. На это состояние указывает параметр флага *backup_pending*.

Если оба этих параметра выключаются, повтор транзакций для базы данных становится невозможным, поскольку журналы для нее более не сохраняются. В этом случае менеджер баз данных удаляет все файлы журналов из каталога

logpath (в том числе оперативные архивные файлы журналов), размещает новые активные файлы журналов и возвращается к циклической записи.

Дополнительную информацию об обработчиках пользователя смотрите в разделе “Обработчики пользователя для восстановления баз данных” книги *Administration Guide: Implementation*.

Восстановление

Следующие параметры влияют на различные аспекты восстановления баз данных:

- “Разрешение автоматического перезапуска (autorestart)”
- “Время пересоздания индекса (indexrec)” на стр. 448
- “Число сеансов по умолчанию при восстановлении загрузки (dft_loadrec_ses)” на стр. 449
- “Число резервных копий базы данных (num_db_backups)” на стр. 450
- “Период хранения хронологии восстановления (rec_his_retentn)” на стр. 450
- “Разрешить отслеживание измененных страниц (trackmod)” на стр. 451

Смотрите также раздел “Восстановление распределенной единицы работы” на стр. 453.

Следующие параметры используются при работе с TSM (Tivoli Storage Manager):

- “Класс управления Tivoli Storage Manager (tsm_mgmtclass)” на стр. 452
- “Пароль Tivoli Storage Manager (tsm_password)” на стр. 452
- “Имя узла Tivoli Storage Manager (tsm_nodename)” на стр. 452
- “Имя владельца Tivoli Storage Manager (tsm_owner)” на стр. 453

Разрешение автоматического перезапуска (autorestart)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] ON [ON; OFF]

Если этот параметр имеет значение ON, менеджер баз данных при необходимости автоматически вызывает утилиту перезапуска базы данных при соединении прикладной программы с базой данных. *Аварийное восстановление* - это операция, выполняемая утилитой перезапуска базы данных. Она выполняется, если работа базы данных была завершена аварийно, когда с ней были соединены прикладные программы. Аварийное завершение работы базы данных может быть вызвано сбоем питания или сбоем в работе системного программного обеспечения. Аварийное восстановление применяет принятые транзакции, которые находились в пуле буферов базы данных, но не были

записаны на диск во время сбоя. Оно также отменяет все непринятые транзакции, которые могли быть записаны на диск.

Если параметр *autorestart* имеет значение OFF, прикладная программа, пытающаяся соединиться с базой данных, для которой требуется выполнение аварийного восстановления (то есть для которой требуется перезапуск), получит сообщение об ошибке SQL1015N. В этом случае утилиту перезапуска базы данных может вызвать прикладная программа или вы сами можете перезапустить базу данных, выбрав операцию перезапуска утилиты восстановления.

Время пересоздания индекса (indexrec)

Тип конфигурации База данных и менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

UNIX, Менеджер баз данных
restart [restart; access]

OS/2 и Windows NT, Менеджер баз данных
access [restart; access]

База данных system [system; restart; access]

Параметры, связанные с данным

“Разрешение автоматического перезапуска (autorestart)” на стр. 447

Этот параметр указывает, когда менеджер баз данных будет пытаться пересоздать неверные индексы. Возможны три значения:

SYSTEM *использовать значение, заданное системой* - неверные индексы пересоздаются в момент времени, заданный в файле конфигурации менеджера баз данных. (Примечание: это значение допустимо лишь для конфигураций баз данных.)

ACCESS *при обращении к индексу* - неверные индексы пересоздаются при первом обращении к ним.

RESTART *при перезапуске базы данных* - неверные индексы

перестраиваются при явном или неявном выполнении команды RESTART DATABASE. Обратите внимание на то, что команда RESTART DATABASE выполняется неявно, если включен параметр *autorestart*.

Числовые эквиваленты и константы API для этих значений приводятся в книге *Administrative API Reference*.

Причиной неверных индексов могут быть неисправимые ошибки диска. Если это происходит с самими данными, они могут быть утрачены. Однако если это случается с индексом, его можно восстановить путем пересоздания. Если индекс пересоздается, пока пользователи соединены с базой данных, возможны ошибки двух типов:

- При пересоздании файла индекса возможно неожиданное ухудшение времени ответа. Пользователи, обращающиеся к таблице и использующие данный индекс, должны будут дожидаться его пересоздания.
- После пересоздания индекса могут сохраняться непредвиденные блокировки, в особенности, если пользовательская транзакция, вызвавшая пересоздание индекса, никогда не выполняла принятие или откат.

Рекомендация: На сервере с большим числом пользователей, если время перезапуска не играет большой роли, для этой опции лучше всего выбрать пересоздание индекса во время работы утилиты DATABASE RESTART как часть процесса восстановления связи с базой данных после аварии.

Если для данного параметра задано значение “ACCESS”, это приведет к снижению производительности менеджера базы данных в процессе пересоздания индекса. Каждый пользователь, обращающийся к данному индексу или таблице, должен будет ждать пересоздания индекса.

Если для параметра задано значение “RESTART”, время, затрачиваемое на перезапуск базы данных, увеличится из-за пересоздания индекса, но процесс обработки пойдет нормально с момента восстановления связи с базой данных.

Число сеансов по умолчанию при восстановлении загрузки (dft_loadrec_ses)

Тип конфигурации	База данных
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	1 [1 – 30000]
Единица измерения	Счетчик

Этот параметр задает число сеансов по умолчанию, которое будет использоваться при восстановлении загрузки таблицы. Следует задать оптимальное число сеансов ввода/вывода для получения копии загрузки.

Получение копии загрузки - операция, подобная восстановлению. Этот параметр можно переопределить через записи в файле положения копии, заданном переменной среды DB2LOADREC.

Число буферов по умолчанию, используемых для восстановления загрузки, на два больше значения этого параметра. Это число буферов также можно переопределить в файле положения копии.

Этот параметр применим, только если разрешено восстановление с повтором.

Дополнительную информацию о восстановлении загрузки смотрите в руководстве *Data Movement Utilities Guide and Reference*.

Число резервных копий базы данных (num_db_backups)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 12 [1 – 32768]

Параметры, связанные с данным

"Период хранения хронологии восстановления (rec_his_retentn)"

Этот параметр задает число резервных копий, сохраняемых для базы данных. Когда заданное число достигнуто, старые резервные копии помечаются в файле хронологии восстановления как копии с истекшим сроком хранения. Записи файла хронологии восстановления для резервных копий табличных пространств и копий загрузок, связанные с этими копиями, также помечаются как копии с истекшим сроком хранения. Копия с пометкой об истечении срока хранения может быть удалена со своего носителя (например, диска, ленты, ADSM). При следующем резервном копировании базы данных эти записи будут удалены из файла хронологии восстановления.

Когда копия базы данных помечена в файле хронологии как копия с истекшим сроком хранения, все соответствующие резервные копии файлов, связанные через DB2 Data Links Manager, будут удалены с архивного сервера.

Значение параметра конфигурации *rec_his_retentn* должно быть согласовано со значением *num_db_backups*. Например, если для *num_db_backup* задано большое значение, значение *rec_his_retentn* должно быть достаточно большим для поддержки такого числа резервных копий.

Период хранения хронологии восстановления (rec_his_retentn)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 366 [-1; 0 – 30000]

Единица измерения Дни

Параметры, связанные с данным

“Число резервных копий базы данных (num_db_backups)” на стр. 450

Этот параметр задает срок в днях, в течение которого должна храниться хронологическая информация резервных копий. Если файл хронологии восстановления не требуется для слежения за снятием резервных копий, восстановлениями и загрузками, для этого параметра можно задать небольшое значение.

Если этот параметр имеет значение -1, файл хронологии восстановления можно сокращать только вручную при помощи доступных команд или API. Если значение отлично от -1, файл хронологии восстановления сокращается после каждого снятия полной резервной копии.

Значение этого параметра переопределяет значение параметра *num_db_backups*, однако *rec_his_retentn* и *num_db_backups* надо использовать вместе. Если значение *num_db_backups* велико значение *rec_his_retentn* должно быть достаточным для поддержки большого числа резервных копий.

Независимо от заданного периода хранения последняя полная резервная копия и ее набор восстановления сохраняются, если только вы не использовали утилиту PRUNE с опцией FORCE. Дополнительную информацию об этой утилите смотрите в справочнике *Command Reference*.

Разрешить отслеживание измененных страниц (trackmod)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] No [Yes, No]

Если для этого параметра задано значение ″Yes″, менеджер баз данных отслеживает изменения базы данных, так что утилита резервного копирования может узнать, какие страницы базы данных надо проверить при снятии инкрементной резервной копии и в потенциале включить в образ резервной копии. После установки для этого параметра значения ″Yes″ нужно создать резервную копию базы данных, чтобы иметь основу для инкрементных резервных копий. Кроме того, если этот параметр включен и табличное пространство создано, необходимо снять резервную копию, содержащую это табличное пространство. Это может быть копия базы данных или копия табличного пространства. После этого инкрементные резервные копии смогут содержать это табличное пространство.

Класс управления Tivoli Storage Manager (tsm_mgmtclass)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] Null [любая строка]

Класс управления Tivoli Storage Manager задает, как сервер TSM должен управлять резервными версиями копируемых объектов.

По умолчанию класс управления TSM не задан.

Этот класс управления назначается администратором Tivoli Storage Manager. Если вы назначаете класс, задайте для этого параметра имя класса управления. При выполнении резервного копирования TSM менеджер баз данных использует этот параметр для передачи класса управления TSM.

Дополнительную информацию о Tivoli Storage Manager смотрите в главе “Tivoli Storage Manager” книги *Data Recovery and High Availability Guide and Reference*.

Пароль Tivoli Storage Manager (tsm_password)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] Null [любая строка]

Этот параметр используется для переопределения пароля, связанного с TSM (Tivoli Storage Manager). Пароль необходим для восстановления базы данных, если резервная копия была сделана на TSM с другого узла.

Примечание: Если *tsm_nodename* переопределен при резервном копировании в DB2 (например, при помощи команды BACKUP DATABASE), может также понадобится задать *tsm_password*.

Параметр по умолчанию позволяет восстанавливать базу данных с TSM на том же узле, с которого была снята резервная копия. Возможно, что *tsm_nodename* был переопределен при резервном копировании, выполненном в DB2.

Дополнительную информацию о Tivoli Storage Manager смотрите в главе “Tivoli Storage Manager” книги *Data Recovery and High Availability Guide and Reference*.

Имя узла Tivoli Storage Manager (tsm_nodename)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] Null [любая строка]

Этот параметр используется для переопределения имени узла, связанного с TSM (Tivoli Storage Manager). Имя узла необходимо для восстановления базы данных, если резервная копия была сделана на TSM с другого узла.

Параметр по умолчанию позволяет восстанавливать базу данных с TSM на том же узле, с которого была снята резервная копия. *tsm_nodename* можно переопределить при резервном копировании в DB2 (например, при помощи команды BACKUP DATABASE).

Дополнительную информацию о Tivoli Storage Manager смотрите в главе “Tivoli Storage Manager” книги *Data Recovery and High Availability Guide and Reference*.

Имя владельца Tivoli Storage Manager (tsm_owner)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] Null [любая строка]

Этот параметр используется для переопределения владельца, связанного с TSM (Tivoli Storage Manager).Имя владельца необходимо для восстановления базы данных, если резервная копия была сделана на ADSM с другого узла. *tsm_owner* можно переопределить при резервном копировании в DB2 (например, при помощи команды BACKUP DATABASE).

Примечание: Имя пользователя регистрозависимо.

Параметр по умолчанию позволяет восстанавливать базу данных с TSM на том же узле, с которого была снята резервная копия.

Дополнительную информацию о Tivoli Storage Manager смотрите в главе “Tivoli Storage Manager” книги *Data Recovery and High Availability Guide and Reference*.

Восстановление распределенной единицы работы

Следующие параметры влияют на восстановление транзакций распределенной единицы работы (DUOW):

- “Имя базы данных менеджера транзакций (tm_database)” на стр. 454
- “Интервал ресинхронизации транзакций (resync_interval)” на стр. 454
- “Путь файлов журнала менеджера точек синхронизации (spm_log_path)” на стр. 455
- “Имя менеджера точек синхронизации (spm_name)” на стр. 456
- “Размер файлов журнала менеджера точек синхронизации (spm_log_file_sz)” на стр. 456
- “Предельное число агентов ресинхронизации менеджера точек синхронизации (spm_max_resync)” на стр. 457

Имя базы данных менеджера транзакций (tm_database)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 1ST_CONN [любое правильное имя базы данных]

Этот параметр задает имя базы данных менеджера транзакций (TM) для каждого экземпляра DB2. База данных TM может быть:

- Локальной базой данных DB2 Universal Database
- Удаленная база данных DB2 Universal Database, которая не находится на системе хоста или AS/400
- База данных DB2 for OS/390 Версии 5, если обращение производится по протоколу TCP/IP и не используется менеджер точек синхронизации (SPM).

База данных TM используется для регистрации и координации, и применяется при восстановлении для неоднозначных транзакций.

Можно задать для этого параметра значение **1ST_CONN**, при этом базой данных TM станет первая база, к которой подключится пользователь.

Дополнительную информацию о распределенных единицах работы смотрите в главе “Распределенные базы данных” руководства *Administration Guide: Planning*.

Рекомендация: Чтобы упростить управление и работу, можно создать несколько баз данных для многих экземпляров и использовать их исключительно как базы данных TM.

Интервал ресинхронизации транзакций (resync_interval)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами

- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 180 [1 – 60000]

Единица измерения Секунды

Этот параметр задает интервал времени в секундах, через который менеджер транзакций (TM), менеджер ресурсов (RM) или менеджер точек синхронизации (SPM) должны предпринимать попытки восстановления ожидающих неоднозначных транзакций, обнаруженных в TM, RM или SPM. Этот параметр применяется, если у вас есть транзакции, выполняемые в среде распределенных единиц работы (DUOW).

Дополнительную информацию о распределенных единицах работы смотрите в главе “Распределенные базы данных” руководства *Administration Guide: Planning*.

Рекомендация: Если в вашей среде неоднозначные транзакции не мешают другим транзакциям базы данных, вы можете захотеть увеличить значение этого параметра. Если для доступа к серверам прикладных программ DRDA2 используется шлюз DB2 Connect, надо учесть возможное влияние неоднозначных транзакций на серверы прикладных программ, даже если они и не мешают локальному доступу к данным. Если неоднозначных транзакций нет, влияние на производительность будет минимальным.

Путь файлов журнала менеджера точек синхронизации (spm_log_path)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию sqllib/spmlog [любой допустимый путь или устройство]

Этот параметр задает каталог, в который записываются журналы менеджера точек синхронизации (SPM). По умолчанию они записываются в каталог

sqllib/spmlog, что при большом количестве транзакций может замедлять ввод/вывод. Данный параметр позволяет разместить файлы журнала SPM на более быстром диске, чем текущий каталог sqllib/spmlog. Это улучшит параллельную работу агентов SPM.

Дополнительную информацию о менеджере точек синхронизации смотрите в книге *Дополнение по установке и конфигурированию*.

Дополнительную информацию о восстановлении неоднозначных транзакций DRDA смотрите в разделе "Восстановление неоднозначных транзакций на хосте" в книге *Administration Guide: Planning*.

Имя менеджера точек синхронизации (spm_name)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию Определяется по имени хоста TCP/IP

Этот параметр задает имя экземпляра менеджера точек синхронизации (SPM) для менеджера баз данных.

Дополнительную информацию о менеджере точек синхронизации смотрите в книге *Дополнение по установке и конфигурированию*.

Дополнительную информацию о восстановлении неоднозначных транзакций DRDA смотрите в разделе "Восстановление неоднозначных транзакций на хосте" в книге *Administration Guide: Planning*.

Размер файлов журнала менеджера точек синхронизации (spm_log_file_sz)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами

- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 256 [4 – 1000]

Единица измерения Страницы

Этот параметр задает размер файла журнала менеджера точек синхронизации (SPM) в страницах по 4 Кбайта. Этот файл журнала находится в подкаталоге `spmlog` каталога `sqllib` и создается при первом запуске SPM.

Дополнительную информацию о менеджере точек синхронизации смотрите в книге *Дополнение по установке и конфигурированию*.

Дополнительную информацию о восстановлении неоднозначных транзакций DRDA смотрите в разделе “Восстановление неоднозначных транзакций на хосте” в книге *Administration Guide: Planning*.

Рекомендация: Файл журнала менеджера точек синхронизации должен быть достаточно большим, чтобы обеспечить хорошую производительность, но не должен занимать лишнее место. Его размер зависит от количества транзакций, использующих защищенные диалоги, и от частоты использования операторов COMMIT и ROLLBACK.

Чтобы изменить размер файла журнала SPM:
1. При помощи команды LIST DRDA INDOUBT TRANSACTIONS убедитесь, что нет неоднозначных транзакций.
2. Если это так, остановить менеджер баз данных.
3. Укажите в конфигурации менеджера баз данных новое значение размера файла журнала SPM.
4. Перейдите в каталог `$HOME/sqllib` и введите команду `rm -fr spmlog`, чтобы удалить текущий журнал SPM. (Примечание: Эта команда используется в AIX. В других системах команда remove или delete может выглядеть по-другому.)
5. Запустите менеджер баз данных. При его запуске будет создан новый журнал SPM указанного размера.

Предельное число агентов ресинхронизации менеджера точек синхронизации (spm_max_resync)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами

- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 20 [10 – 256]

Этот параметр задает число агентов, которые могут одновременно выполнять операции ресинхронизации.

Дополнительную информацию о восстановлении неоднозначных транзакций DRDA смотрите в разделе “Восстановление неоднозначных транзакций на хосте” в книге *Administration Guide: Planning*.

Дополнительную информацию о менеджере точек синхронизации смотрите в книге *Дополнение по установке и конфигурированию*.

Управление базами данных

Ряд параметров содержит информацию о вашей базе данных или влияет на управление ей. Они разбиты на следующие группы:

- “Query Enabler”
- “Атрибуты” на стр. 459
- “DB2 Data Links Manager” на стр. 462
- “Состояние” на стр. 465
- “Настройка компилятора” на стр. 467.

Query Enabler

Следующие параметры содержат информацию об управлении Query Enabler:

- “Управление запросами динамического SQL (dyn_query_mgmt)”

Управление запросами динамического SQL (dyn_query_mgmt)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 0 (DISABLE) [1(ENABLE), 0 (DISABLE)]

Этот параметр используется, если установлен DB2 Query Patroller. Если параметр конфигурации базы данных *dyn_query_mgmt* имеет значение “ENABLE” и стоимость динамического запроса превышает trap_threshold для этого пользователя или группы (заданный в таблице профилей пользователей DB2 Query Patroller), такой запрос перехватывает DB2 Query Patroller. Порог

trap_threshold - триггер на основе стоимости для перехвата запросов, устанавливаемый DB2 Query Patroller для пользователя. После перехвата динамического запроса открывается диалоговое окно, в котором пользователь задает параметры выполнения.

Если *dyn_query_mgmt* имеет значение “DISABLE”, перехвата запросов не происходит.

Атрибуты

Следующие параметры содержат общую информацию о базе данных:

- “Уровень выпуска файла конфигурации (release)”
- “Уровень выпуска базы данных (database_level)”
- “Территория для базы данных (territory)” на стр. 460
- “Код страны для базы данных” на стр. 460
- “Кодовый набор для базы данных (codeset)” на стр. 460
- “Кодовая страница для базы данных (codepage)” на стр. 461
- “Информация упорядочения (collate_info)” на стр. 461
- “Включение защиты от копирования (copyprotect)” на стр. 462

За исключением *copyprotect*, эти параметры служат чисто информационным целям.

Уровень выпуска файла конфигурации (release)

Тип конфигурации Менеджер баз данных, база данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Информационный

Параметры, связанные с данным

“Уровень выпуска базы данных (database_level)”

Этот параметр задает уровень выпуска файла конфигурации.

Уровень выпуска базы данных (database_level)

Тип конфигурации База данных

Тип параметра Информационный

Параметры, связанные с данным

"Уровень выпуска файла конфигурации (release)" на стр. 459

Этот параметр указывает уровень выпуска менеджера баз данных, который может использовать эту базу данных. Если перенастройка не завершена или завершилась неудачно, этот параметр будет отражать уровень выпуска базы данных до перенастройки и может при этом отличаться от параметра *release* (уровень выпуска файла конфигурации базы данных). В прочих случаях значение *database_level* будет совпадать со значением параметра *release*.

Территория для базы данных (territory)

Тип конфигурации База данных

Тип параметра Информационный

Параметры, связанные с данным

"Код страны для базы данных"

Этот параметр показывает территорию, использованную для создания базы данных. Территория используется менеджером баз данных для определения значений параметра *country*. Дополнительную информацию о том, как менеджер баз данных использует территорию, смотрите в приложении Поддержка национальных языков в книге *Administration Guide: Planning*.

Код страны для базы данных

Тип конфигурации База данных

Тип параметра Информационный

Параметры, связанные с данным

"Территория для базы данных (territory)"

Этот параметр показывает код страны, использованный для создания базы данных. Параметр *country* определяется на основе параметра *territory*. Дополнительную информацию о том, как менеджер баз данных использует код страны, смотрите в приложении Поддержка национальных языков в книге *Administration Guide: Planning*.

Кодовый набор для базы данных (codeset)

Тип конфигурации База данных

Тип параметра Информационный

Параметры, связанные с данным

"Кодовая страница для базы данных (codepage)" на стр. 461

Этот параметр показывает кодовый набор, использованный для создания базы данных. Кодовый набор пользуется менеджером баз данных для определения значений параметра *codepage*. Дополнительную информацию о том, как менеджер баз данных использует кодовый набор, смотрите в приложении Поддержка национальных языков в книге *Administration Guide: Planning*.

Кодовая страница для базы данных (codepage)

Тип конфигурации База данных

Тип параметра Информационный

Параметры, связанные с данным

"Кодовый набор для базы данных (codeset)" на стр. 460

Этот параметр показывает кодовую страницу, использованную для создания базы данных. Параметр *codepage* определяется на основе параметра *codeset*. Дополнительную информацию о том, как менеджер баз данных использует кодовую страницу, смотрите в приложении Поддержка национальных языков в книге *Administration Guide: Planning*.

Информация упорядочения (collate_info)

Этот параметр можно вывести только с помощью API GET DATABASE CONFIGURATION. Через процессор командной строки или из Центра управления вывести его **нельзя**.

Тип конфигурации База данных

Тип параметра Информационный

Этот параметр задает 260 байтов информации упорядочения базы данных. Первые 256 байтов задают последовательность упорядочения базы данных, где байт "n" содержит вес сортировки кода символа с десятичным представлением "n" в кодовой странице базы данных.

Последние 4 байта содержат внутреннюю информацию о типе последовательности упорядочения. Его можно рассматривать как целое число в формате платформы базы данных. Возможны три значения:

- **0** – Последовательность содержит неуникальные веса
- **1** – Все веса последовательности уникальны
- **2** – Последовательность является последовательностью идентификации, строки которой сравниваются побайтно.

Если эта внутренняя информация типа используется, необходимо учитывать различный порядок следования байтов при получении информации для базы данных на различных платформах.

Последовательность упорядочения можно задать во время создания базы данных.

Включение защиты от копирования (copyprotect)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] No [Yes; No]

Этот параметр включает атрибут защиты от копирования; по умолчанию он отключен. До Версии 2 менеджера баз данных атрибут защиты от копирования был по умолчанию включен.

Этот параметр не применяется для сред на основе UNIX.

Параметр *copyprotect* не влияет на утилиты резервного копирования и восстановления баз данных. Можно сделать резервную копию базы данных, для которой включена защита от копирования, восстановить эту базу данных на другой рабочей станции и затем внести ее в каталог и обращаться к ней.

Внимание: Отключите защиту от копирования для всех баз данных перед переустановкой менеджера баз данных или операционной системы. Если не отключить защиту от копирования, при попытке обратиться к базе данных будет выдано сообщение об ошибке. Завершив переустановку, можно включить защиту от копирования.

DB2 Data Links Manager

Следующие параметры относятся к менеджеру связей данных DB2:

- “Срок действия маркера связей данных (dl_expint)”
- “Число копий связей данных (dl_num_copies)” на стр. 463
- “Время после отбрасывания связей данных (dl_time_drop)” на стр. 463
- “Алгоритм маркера связей данных (dl_token)” на стр. 464
- “Верхний регистр в маркере связей данных (dl_upper)” на стр. 464
- “Поддержка связей данных (datalinks)” на стр. 464

Срок действия маркера связей данных (dl_expint)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 60 [-1, 1 – 31536000]

Единица измерения Секунды

Этот параметр задает временной интервал (в секундах), в течение которого действует сгенерированный маркер доступа к файлу. Срок начинается с момента генерации маркера. Data Links Filesystem Filter проверяет, не истек ли срок действия маркера.

Информацию о маркерах доступа к файлу смотрите в руководстве *DB2 Data Links Manager Quick Beginnings*.

Значение этого параметра по умолчанию - 60 секунд. Если для этого параметра задано значение ″-1″, срок маркера управления доступом истекает. Вместо этого можно задать для этого параметра максимальное значение 31536000 (секунд). Это соответствует сроку истечения 1 год, чего должно хватить для всех программ.

Этот параметр применяется к столбцам DATALINK, для которых задано “READ PERMISSION DB”.

Число копий связей данных (dl_num_copies)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 0 [0 – 15]

Этот параметр задает число дополнительных копий файла, создаваемых на сервере архива (например, на сервере TSM), когда файл связывается с базой данных.

Значение этого параметра по умолчанию - ноль (0).

Этот параметр применяется к столбцам DATALINK, для которых задано “Recovery=Yes”.

Время после отбрасывания связей данных (dl_time_drop)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 1 [0 – 365]

Единица измерения Дни

Этот параметр задает срок хранения в днях файлов на архивном сервере (например, на сервере TSM) после выполнения команды DROP DATABASE.

Значение этого параметра по умолчанию - один (1) день. Значение ноль (0) задает немедленное удаление файлов с архивного сервера после выполнения

команды DROP. (Сам файл при этом не удаляется, если только для столбца DATALINK не задан параметр ON UNLINK DELETE.)

Этот параметр применяется к столбцам DATALINK, для которых задано “Recovery=Yes”.

Алгоритм маркера связей данных (dl_token)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] MAC0 [MAC0; MAC1]

Этот параметр задает алгоритм генерации маркеров управления доступом к файлу DATALINK. Значение MAC1 (message authentication code - код аутентификации сообщений) задает генерацию более защищенного кода аутентификации сообщений, чем MAC0, но при этом может снизиться производительность.

Информацию о маркерах доступа к файлу смотрите в руководстве *DB2 Data Links Manager Quick Beginnings*.

Этот параметр применяется к столбцам DATALINK, для которых задано “READ PERMISSION DB”.

Верхний регистр в маркере связей данных (dl_upper)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] NO [YES; NO]

Этот параметр указывает, используется ли в маркерах управления доступом к файлам символы верхнего регистра. Значение “YES” задает, что в маркерах управления доступом используются только символы верхнего регистра. Значение “NO” задает, что маркер может содержать символы как верхнего, так и нижнего регистра.

Информацию о маркерах доступа к файлу смотрите в руководстве *DB2 Data Links Manager Quick Beginnings*.

Этот параметр применяется к столбцам DATALINK, для которых задано “READ PERMISSION DB”.

Поддержка связей данных (datalinks)

Тип конфигурации Менеджер баз данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] NO [YES; NO]

Этот параметр задает, включена ли поддержка связей данных. Значение “YES” включает поддержку связей данных менеджером связей данных для связывания файлов, хранящихся в собственной файловой системе (например, JFS в AIX). Значение “NO” задает, что поддержка связей данных не включена.

Состояние

Следующие параметры содержат информацию о состоянии базы данных:

- “Индикатор отложенного резервного копирования (backup_pending)”
- “Согласованность базы данных (database_consistent)”
- “Индикатор отложенного повтора транзакций (rollfwd_pending)” на стр. 466
- “Индикатор состояния сохранения журналов (log_retain_status)” на стр. 466
- “Индикатор состояния обработчика пользователя (user_exit_status)” на стр. 466
- “Отложенное восстановление (restore_pending)” на стр. 467
- “Многостраничное размещение файлов разрешено (multipage_alloc)” на стр. 467

Индикатор отложенного резервного копирования (backup_pending)

Тип конфигурации База данных

Тип параметра Информационный

Значение ON этого параметра указывает, что перед обращением к базе данных необходимо сделать ее полную резервную копию. Этот параметр принимает значение ON, только если была изменена конфигурация базы данных, в результате чего база данных перешла из невосстановимого состояния в восстановимое (то есть сначала оба параметра *logretain* и *userexit* имели значение NO, затем одному из них или обоим было присвоено значение YES и изменения конфигурации базы данных были приняты).

Согласованность базы данных (database_consistent)

Тип конфигурации База данных

Тип параметра Информационный

Этот параметр указывает, находится ли база данных в согласованном состоянии.

YES указывает, что для всех транзакций выполнено принятие или откат, и данные находятся в согласованном состоянии. Если “отказ” системы

происходит, когда база данных находится в согласованном состоянии, никаких специальных действий по возврату базы данных к работе делать не надо.

NO указывает, что имеется незавершенная транзакция или какое-либо другое задание не завершено для данной базы, и данные в настоящее время находятся в несогласованном состоянии. Если "отказ" системы произойдет, когда база данных находится в этом состоянии, базу данных придется перезапустить при помощи команды RESTART DATABASE, чтобы сделать ее пригодной к использованию. Дополнительную информацию о команде RESTART DATABASE смотрите в руководстве *Command Reference*.

Индикатор отложенного повтора транзакций (rollfwd_pending)

Тип конфигурации База данных

Тип параметра Информационный

Этот параметр может указывать на одно из следующих состояний:
- **DATABASE**, означает, что необходимо произвести восстановление с повтором транзакций для этой базы данных.
- **TABLESPACE**, означает, что требуется восстановление с повтором для одного или нескольких табличных пространств.
- **NO**, означает, что база данных пригодна для использования и повтор транзакций не требуется.

Восстановление (при помощи ROLLFORWARD DATABASE) должно быть выполнено, прежде чем вы сможете использовать базу данных или табличное пространство. Дополнительную информацию о ROLLFORWARD DATABASE смотрите в книге *Command Reference*.

Индикатор состояния сохранения журналов (log_retain_status)

Тип конфигурации База данных

Тип параметра Информационный

Параметры, связанные с данным

"Разрешение хранения журнала (logretain)" на стр. 445

Если установлен, этот параметр указывает, что файлы журналов сохраняются для использования при восстановлении с повтором транзакций.

Этот параметр задается в тех случаях, когда параметр *logretain* имеет значение Recovery.

Индикатор состояния обработчика пользователя (user_exit_status)

Тип конфигурации База данных

Тип параметра Информационный

Параметры, связанные с данным

"Разрешение обработчика пользователя (userexit)" на стр. 446

Если этот параметр имеет значение Yes, это означает, что менеджер баз данных допускает восстановление с повтором и что для архивирования и восстановления файлов журнала будет использоваться обработчик пользователя, вызываемый менеджером баз данных.

Отложенное восстановление (restore_pending)

Тип конфигурации База данных

Тип параметра Информационный

Этот параметр указывает, существует ли в базе данных состояние RESTORE PENDING.

Многостраничное размещение файлов разрешено (multipage_alloc)

Тип конфигурации База данных

Тип параметра Информационный

Многостраничное размещение файлов используется для увеличения производительности операций вставки. Это относится только к табличным пространствам SMS. Если оно разрешено, это влияет на все табличные пространства SMS: нельзя разрешить его только для отдельных табличных пространств SMS.

Значение этого параметра по умолчанию - No (многостраничное размещение файлов не разрешено).

После создания базы данных этому параметру можно присвоить значение Yes, которое указывает, что многостраничное размещение файлов разрешено. Чтобы сделать это, используйте утилиту *db2empfa*. После задания для этого параметра значения Yes его нельзя изменить назад на No.

Дополнительную информацию смотрите в руководстве *Command Reference*.

Настройка компилятора

Следующие параметры содержат информацию, влияющую на работу компилятора:

- "Продолжение при арифметических исключительных ситуациях (dft_sqlmathwarn)" на стр. 468
- "Степень параллелизма по умолчанию (dft_degree)" на стр. 469

- “Класс оптимизации запросов по умолчанию (dft_queryopt)” на стр. 470
- “Срок обновления по умолчанию (dft_refresh_age)” на стр. 471
- “Число сохраняемых частых значений (num_freqvalues)” на стр. 471
- “Число квантилей для столбцов (num_quantiles)” на стр. 472

Продолжение при арифметических исключительных ситуациях (dft_sqlmathwarn)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] No [No, Yes]

Этот параметр задает значение по умолчанию, которое определяет, как обрабатывать при компиляции операторов SQL арифметические ошибки и ошибки преобразования (как предупреждения или же как ошибки). Для статических операторов SQL значение этого параметра связывается с пакетом во время связывания. Для динамических операторов DML SQL значение этого параметра используется после подготовки оператора.

Внимание: Если изменить значение *dft_sqlmathwarn* для базы данных, поведение проверочных ограничений, триггеров и производных таблиц, включающих арифметические выражения, может измениться. Это, в свою очередь, может повлиять на целостность данных в базе данных. Меняя значение *dft_sqlmathwarn* для базы данных, надо тщательно продумать, как новые арифметические исключительные ситуации могут повлиять на проверочные ограничения, триггеры и производные таблицы. Все последующие изменения требуют не менее тщательной оценки.

В качестве примера рассмотрим следующее проверочное ограничение, содержащее арифметическую операцию деления:

```
A/B > 0
```

Если задано значение *dft_sqlmathwarn* “No” и предпринимается попытка применить оператор INSERT с B=0, деление на нуль обрабатывается как арифметическая ошибка. Операция вставки завершается неудачно, так как DB2 не может проверить ограничение. Если значение *dft_sqlmathwarn* изменено на “Yes”, деление на нуль обрабатывается как арифметическое предупреждение с результатом NULL. Пустой результат приводит к оценке предиката “>” как неизвестного, и операция вставки завершается успешно. Если значение параметра *dft_sqlmathwarn* снова изменить на “No”, попытка вставить ту же строку будет неудачной, поскольку деление на ноль мешает DB2 оценить ограничение. Строка, вставленная при B=0 в момент, когда *dft_sqlmathwarn* имел значение “Да” остается в таблице, и ее можно выбрать. Изменения в строке, приводящие к оценке ограничения, будут неудачными, а изменения в строке, не требующие повторной оценки ограничения, будут успешными.

Прежде, чем изменить значение *dft_sqlmathwarn* с “No” на “Yes”, необходимо переписать ограничение так, чтобы оно явно обрабатывало пустые значения в арифметических выражениях. Например:

```
  ( A/B > 0 ) AND ( CASE
                      WHEN A IS NULL THEN 1
                      WHEN B IS NULL THEN 1
                      WHEN A/B IS NULL THEN 0
                      ELSE 1
END
                    = 1 )
```

можно использовать, если и A, и B допускают отсутствие значения. Если же A или B не допускают пустых значений, соответствующее условие IS NULL WHEN можно удалить.

Прежде чем изменить значение *dft_sqlmathwarn* с “Yes” на “No”, следует вначале проверить, не может ли возникнуть несогласованность в данных, например с помощью показанных ниже предикатов:

```
   WHERE A IS NOT NULL AND B IS NOT NULL AND A/B IS NULL
```

Когда несогласованные строки обнаружены, надо предпринять соответствующие действия, чтобы исправить это несоответствие до изменения *dft_sqlmathwarn*. Можно также проверить ограничения с арифметическими выражениями вручную после изменения. Для этого сначала переведите затронутые таблицы в состояние ожидания проверки (с помощью условия OFF оператора SET CONSTRAINTS), затем потребуйте проверить таблицы (с помощью условия IMMEDIATE CHECKED оператора SET CONSTRAINTS). Несогласованные данные будут отмечены арифметической ошибкой, которая препятствует оценке ограничений.

Рекомендация: Если вам специально не требуется обрабатывать требования, включающие арифметические ошибки, используйте значение по умолчанию ″No″. В противном случае задайте значение ″Yes″. Эта ситуация может иметь место при обработке операторов SQL, которые в других менеджерах баз данных выдают результаты независимо от возможных арифметических ошибок.

Степень параллелизма по умолчанию (dft_degree)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 1 [-1, 1 – 32767]

Параметры, связанные с данным

“Максимальная степень параллелизма запросов (max_querydegree)” на стр. 493

Этот параметр задает значение по умолчанию для специального регистра CURRENT DEGREE и опции связывания DEGREE.

Значение по умолчанию - 1.

1 означает, что внутриразделный параллелизм не используется. Значение -1 означает, что степень внутриразделного параллелизма определяется оптимизатором на основе числа процессоров и типа запроса.

Степень внутриразделного параллелизма для оператора SQL задается при компиляции этого оператора с помощью специального регистра CURRENT DEGREE или опции связывания DEGREE. Максимальная степень внутриразделного параллелизма во время выполнения для активной прикладной программы задается командой SET RUNTIME DEGREE. Параметр конфигурации *max_querydegree* (максимальная степень параллелизма запроса) задает максимальную степень внутриразделного параллелизма для всех запросов SQL.

Реальная степень параллелизма во время выполнения - это минимум из:
- Значения параметра конфигурации *max_querydegree*
- Степень параллелизма времени выполнения прикладной программы
- Степень параллелизма, заданная при компиляции оператора SQL

Класс оптимизации запросов по умолчанию (dft_queryopt)

Тип конфигурации	База данных
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	5 [0 – 9]
Единица измерения	Класс оптимизации запроса (смотрите ниже)

Класс оптимизации запроса используется, чтобы задавать для оптимизатора различные уровни оптимизации при компиляции запросов SQL. Этот параметр обеспечивает дополнительную гибкость, задавая класс оптимизации запросов по умолчанию, используемый, когда не применяются ни оператор SET CURRENT QUERY OPTIMIZATION, ни команда связывания QUERYOPT.

В настоящее время определены следующие классы оптимизации:

0 - минимальная оптимизация запросов.

1 - примерно сравнимая с DB2 Версии 1.

2 - небольшая оптимизация.

3 - умеренная оптимизация запросов.

5 - существенная оптимизация запроса с эвристическими решениями для ограничения расходов на выбор плана. Эта опция принимается по умолчанию.

7 - существенная оптимизация запроса.

9 - максимальная оптимизация запросов.

Рекомендация: Дополнительную информацию о выборе подходящего класса оптимизации запросов смотрите в разделе “Настройка класса оптимизации” на стр. 70.

Дополнительную информацию о том, как программа может прочитать и изменить параметры конфигурации базы данных, смотрите в книге *Administrative API Reference*.

Срок обновления по умолчанию (dft_refresh_age)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 0 [0, 99999999999999 (ANY)]

Значение этого параметра используется по умолчанию для REFRESH AGE, если не задан специальный регистр CURRENT REFRESH AGE. Параметр задает значение длительности времени с типом данных DECIMAL(20,6). Этот интервал задает максимальный срок с момента выполнения оператора REFRESH TABLE для определенной таблицы сводки REFRESH DEFERRED, в течении которого эту таблицу сводки можно использовать для оптимизации запросов. Если CURRENT REFRESH AGE содержит значение 99999999999999 (ANY), а класс QUERY OPTIMIZATION - 5 или выше, таблицы сводок REFRESH DEFERRED будут применяться для оптимизации обработки динамических запросов SQL.

Число сохраняемых частых значений (num_freqvalues)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 10 [0 – 32767]

Единица измерения Счетчик

Параметры, связанные с данным

- “Число квантилей для столбцов (num_quantiles)” на стр. 472
- “Размер кучи статистики (stat_heap_sz)” на стр. 386

Этот параметр позволяет задать число “часто встречающихся значений”, которые будут собираться, когда в команде RUNSTATS задана опция WITH DISTRIBUTION. Увеличение значения этого параметра увеличивает размер кучи статистики (*stat_heap_sz*), используемой для сбора статистики.

Статистика “частых значений” помогает оптимизатору понимать распределение значений данных в столбце. При более высоких значениях оптимизатору SQL доступно больше информации, но требуется дополнительное место в каталоге. Если задано значение 0, статистика частых значений не хранится, даже если вы требуете собрать статистику распределения.

Изменение этого параметра помогает оптимизатору получать более точные оценки селективности для некоторых предикатов (=, <, >, IS NULL, IS NOT NULL), применяемых к данным с неравномерным распределением. Более точная оценка селективности может привести к выбору более эффективного плана доступа.

Изменив значение этого параметра, надо:
- Запустить команду RUNSTATS, когда все пользователи отсоединятся от базы данных, а вы соединитесь с ней повторно
- Повторно связать все пакеты, содержащие статический SQL.

Дополнительную информацию смотрите в разделе “Сбор и использование статистики распределения” на стр. 127.

Рекомендация: Чтобы изменить значение этого параметра, надо определить степень неоднородности для самых важных столбцов (из самых важных таблиц), к которым часто применяются предикаты выбора. Для этого можно использовать оператор SQL SELECT, который ранжирует значения столбца по числу вхождения. Не следует обращать внимание на столбцы с однородным распределением, столбцы уникальности, столбцы длинных значений и больших объектов. Разумные значения этого параметра находятся в диапазоне от 10 до 100.

Обратите внимание на то, что процесс сбора статистики частых значений требует много ресурсов процессора и памяти (*stat_heap_sz*).

Число квантилей для столбцов (num_quantiles)

Тип конфигурации	База данных
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	20 [0 – 32767]
Единица измерения	Счетчик
Параметры, связанные с данным	• “Число сохраняемых частых значений (num_freqvalues)” на стр. 471 • “Размер кучи статистики (stat_heap_sz)” на стр. 386

Этот параметр управляет числом квантилей, которые будут собраны, если в команде RUNSTATS задана опция WITH DISTRIBUTION. Увеличение значения этого параметра приводит к увеличению размера используемой кучи статистики (*stat_heap_sz*) при сборе.

Статистика “квантилей” помогает оптимизатору понять распределение значений данных в столбце. При более высоких значениях оптимизатору SQL доступно больше информации, но требуется дополнительное место в каталоге. Если задано значение 0 или 1, статистики квантилей не сохраняются, даже если затребован сбор статистики распределения.

При неоднородном распределении данных изменение этого параметра позволяет точнее оценить селективность для предикатов диапазона. Среди решений оптимизатора эта информация сильно влияет на выбор просмотра индекса или таблицы. (При обращении к диапазону часто встречающихся значений более эффективен просмотр таблицы, а при обращении к диапазону редких значений - просмотр индекса.)

После изменения значения данного параметра надо:
- Запустить команду RUNSTATS, когда все пользователи отсоединятся от базы данных, а вы соединитесь с ней повторно
- Повторно связать все пакеты, содержащие статический SQL.

Дополнительную информацию смотрите в разделе “Сбор и использование статистики распределения” на стр. 127.

Рекомендация: Значение по умолчанию для данного параметра гарантирует максимальную ошибку оценки около 2,5% для односторонних предикатов диапазона (>, >=, < или <=) и около 5% для предикатов BETWEEN. Простой способ оценить число квантилей:
- Определите максимальную ошибку (в процентах), приемлемую при оценке числа строк для какого-либо запроса диапазона - P
- Число квантилей должно составлять примерно 100/P, если большинство ваших предикатов - типа BETWEEN, и 50/P, если большинство - предикаты диапазона других типов (<, <=, > или >=).

Например, 25 квантилей дадут максимальную ошибку оценки 4% для предикатов BETWEEN и 2% для предикатов ″>″. На практике разумно использовать значение данного параметра от 10 до 50.

Связь

Следующие группы параметров содержат информацию об использовании DB2 в среде клиент/сервер:

- “Настройка протокола клиента”
- “Распределенные службы” на стр. 478
- “программа поиска DB2” на стр. 483

Настройка протокола клиента

Следующие параметры служат для настройки клиента баз данных и сервера баз данных:

- “Имя рабочей станции NetBIOS (nname)”
- “Имя службы TCP/IP (svcename)” на стр. 475
- “Имя программы транзакций APPC (tpname)” на стр. 476
- “Имя файл-сервера IPX/SPX (fileserver)” на стр. 476
- “Имя объекта сервера DB2 IPX/SPX (objectname)” на стр. 477
- “Номер гнезда IPX/SPX (ipx_socket)” на стр. 478

Имя рабочей станции NetBIOS (nname)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию Пустое

Этот параметр позволяет приписать уникальное имя экземпляру базы данных на рабочей станции в сетевой среде NetBIOS. На основе этого *nname* создаются реальные имена NetBIOS, которые будут регистрироваться для NetBIOS на рабочей станции.

Поскольку протокол NetBIOS устанавливает соединения с использованием этих имен NetBIOS, параметр *nname* должен быть задан и на клиенте, и на сервере.

Клиентские прикладные программы должны знать *nname* сервера, где находится база данных, к которой они хотят обращаться. *nname* на сервере должен быть внесен в каталог на узле клиента как параметр “server-nname” при помощи команды CATALOG NETBIOS NODE.

Если *nname* на узле сервера меняется, на всех клиентах, которые обращаются к базам данных на этом сервере, надо занести в каталоги новое имя для этого сервера.

Имя службы TCP/IP (svcename)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию Пустое

Этот параметр содержит имя порта TCP/IP, который будет использоваться сервером баз данных для ожидания связи от удаленных узлов клиентов. Это имя должно быть первым из двух идущих подряд портов, зарезервированных для использования менеджером баз данных; второй порт используется для обработки запросов прерываний от клиентов нижнего уровня.

Чтобы принимать запросы соединения от клиента базы данных через TCP/IP, сервер базы данных должен ожидать запроса через порт, назначенный для этого сервера. Системный администратор сервера базы данных должен зарезервировать порт (номер *n*) и определить связанное с ним имя службы TCP/IP в файле служб на этом сервере. Если сервер базы данных должен поддерживать запросы от клиентов нижнего уровня, в файле служб на этом сервере нужно определить второй порт (номер *n+1*, для запросов прерываний).

Порт сервера базы данных (номер *n*) и его имя службы TCP/IP должны быть определены в файле служб на клиенте базы данных. На клиентах нижнего уровня в файле служб должен быть также определен порт прерываний (номер *n+1*).

Положение файла служб зависит от конкретной операционной среды. Например:
- В UNIX – /etc/services
- В OS/2 – \tcpip\etc\services

- В OS/2 Warp – \mptn\etc\services.

В параметре *svcename* нужно задать имя службы, связанное с основным портом связи, чтобы после запуска сервера базы данных он мог определить, через какой порт нужно ожидать получения входящих запросов соединения. Имя службы для порта прерываний для клиентов нижнего уровня не сохраняется в файле конфигурации. Номер порта прерываний определяется на основе номера основного порта соединений (*номер порта прерываний* = *номер основного порта соединений* + 1).

Дополнительную информацию о настройке TCP/IP для серверов баз данных смотрите в руководстве *Дополнение по установке и конфигурированию*.

Имя программы транзакций APPC (tpname)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию Пустое

Этот параметр определяет имя удаленной программы транзакций, которую клиент базы данных должен использовать при выдаче требования на размещение серверу баз данных при использовании протокола связи APPC. Параметр должен быть задан с файле конфигурации на сервере баз данных.

Значение этого параметра должно совпадать с именем программы транзакций, заданном в определении программы транзакций SNA. Дополнительную информацию о конфигурировании APPC для вашего продукта DB2 смотрите в руководстве *Дополнение по установке и конфигурированию*.

Рекомендация: В этом имени допустимы только следующие символы:
- Латинские буквы (от A до Z и от a до z)
- Цифры (от 0 до 9)
- Знак доллара ($), знак номера (#), символ @ и точка (.)

Имя файл-сервера IPX/SPX (fileserver)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию Пустое

Параметры, связанные с данным

- “Имя объекта сервера DB2 IPX/SPX (objectname)”
- “Номер гнезда IPX/SPX (ipx_socket)” на стр. 478

Этот параметр задает имя файл-сервера NetWare**, где зарегистрирован межсетевой адрес менеджера баз данных. Межсетевой адрес менеджера баз данных хранится на файл-сервере NetWare. Если зарегистрированное имя файл-сервера меняется, все клиенты, которые обращаются к этому экземпляра сервера, должны:
- Удалить из каталога (UNCATALOG) узел сервера
- Занести в каталог (CATALOG) узел сервера, указав новое имя файл-сервера.

Дополнительную информацию смотрите в руководстве *Дополнение по установке и конфигурированию*.

Имя объекта сервера DB2 IPX/SPX (objectname)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию Пустое

Параметры, связанные с данным

- “Имя файл-сервера IPX/SPX (fileserver)” на стр. 476
- “Номер гнезда IPX/SPX (ipx_socket)” на стр. 478

Этот параметр задает имя экземпляра менеджера баз данных в сети IPX/SPX. У каждого экземпляра сервера, зарегистрированного на файл-сервере NetWare, должно быть уникальное имя. Если это имя на сервере баз данных меняется, все клиенты, обращающиеся к этому серверу, должны удалить из каталога узел сервера и внести его снова, задав новое имя объекта.

Дополнительную информацию о конфигурировании IPX/SPX для вашего продукта DB2 смотрите в руководстве *Дополнение по установке и конфигурированию*.

Номер гнезда IPX/SPX (ipx_socket)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 879E [879E – 87A2] Чтобы исключить возможные конфликты, эти пять номеров гнезд (с 879E по 87A2) зарегистрированы на Novell для использования DB2.

Параметры, связанные с данным

- “Имя файл-сервера IPX/SPX (fileserver)” на стр. 476
- “Имя объекта сервера DB2 IPX/SPX (objectname)” на стр. 477

Этот параметр задает “общеизвестный” номер гнезда и представляет конечную точку связи в межсетевом адресе сервера DB2. Номера гнезд должны быть уникальны для каждого экземпляра сервера DB2 на данном компьютере среди всех программ Novell** IPX/SPX, выполняемых на нем. Это гарантирует, что сервер DB2 может ожидать входящие соединения IPX/SPX на гнезде с этим номером.

Дополнительную информацию о конфигурировании IPX/SPX для вашего продукта DB2 смотрите в руководстве *Дополнение по установке и конфигурированию*.

Распределенные службы

Следующие параметры служат для настройки клиента баз данных и сервера баз данных, чтобы использовать службы каталогов DCE:

- “Тип служб каталога (dir_type)”
- “Каталог в пространстве имен DCE (dir_path_name)” на стр. 480
- “Имя объекта в пространстве имен DCE (dir_obj_name)” на стр. 480
- “Имя объекта информации маршрутизации (route_obj_name)” на стр. 481
- “Протокол связи клиента по умолчанию (dft_client_comm)” на стр. 482
- “Номер адаптера клиента по умолчанию (dft_client_adpt)” на стр. 483

Информацию о том, как DB2 использует каталоги DCE, смотрите в теме “Использование службы каталогов DCE” руководства *Administration Guide: Implementation*.

Тип служб каталога (dir_type)

Тип конфигурации Менеджер баз данных

Применяется к
- Сервер баз данных с локальными и удаленными клиентами
- UNIX и OS/2, Клиент
- UNIX и OS/2, Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] NONE [NONE; DCE]

Параметры, связанные с данным
- “Имя объекта в пространстве имен DCE (dir_obj_name)” на стр. 480
- “Каталог в пространстве имен DCE (dir_path_name)” на стр. 480
- “Имя объекта информации маршрутизации (route_obj_name)” на стр. 481
- “Протокол связи клиента по умолчанию (dft_client_comm)” на стр. 482
- “Номер адаптера клиента по умолчанию (dft_client_adpt)” на стр. 483

Этот параметр задает, будут ли использоваться службы каталогов DCE.

Если этот параметр имеет значение NONE, поиск файлов для требований CONNECT или ATTACH будет выполняться только в локальном каталоге. Однако вы по-прежнему можете использовать параметров *dir_path_name* и *dir_obj_name* для задания имени вашего экземпляра баз данных и баз данных в пространстве имен DCE.

Если этот параметр имеет значение DCE, в случаях, когда программа, выполняемая на этом экземпляре менеджера баз данных не может найти назначение требований CONNECT или ATTACH, будет выполнен поиск в каталоге DCE.

Числовые эквиваленты и константы API для этих значений приводятся в книге *Administrative API Reference*.

Каталог в пространстве имен DCE (dir_path_name)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- UNIX и OS/2, Клиент
- UNIX и OS/2, Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию /.:/subsys/database/

Параметры, связанные с данным

- "Имя объекта в пространстве имен DCE (dir_obj_name)"
- "Тип служб каталога (dir_type)" на стр. 479
- "Имя объекта информации маршрутизации (route_obj_name)" на стр. 481

Уникальное имя экземпляра менеджера баз данных в глобальном пространстве имен составляется из этого значения и значения параметра *dir_obj_name*.

Все клиентские программы, работающие на этом экземпляре, используют его также как путь по умолчанию для требований CONNECT или ATTACH, если только он не переопределен значением переменной среды DB2DIRPATHNAME.

Рекомендация: Используйте имя, которое вам сообщит администратор DCE.

Имя объекта в пространстве имен DCE (dir_obj_name)

Тип конфигурации Менеджер баз данных, база данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- UNIX и OS/2, Клиент

- UNIX и OS/2, Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию Пустое

Параметры, связанные с данным

- “Тип служб каталога (dir_type)” на стр. 479
- “Каталог в пространстве имен DCE (dir_path_name)” на стр. 480

Имя объекта представляет экземпляр менеджера баз данных (или базу данных) в каталоге. Конкатенация этого значения и значения *dir_path_name* дает глобальное имя в пространстве имен, которое уникально идентифицирует экземпляр менеджера баз данных или базу данных в пространстве имен, управляемом службами каталога, которые заданы параметром *dir_type*.

Этот параметр имеет смысл, только если задан параметр *dir_path_name*.

Суммарная длина значений параметров конфигурации *dir_path_name* и *dir_obj_name* должна быть менее 255 символов.

Дополнительную информацию смотрите в разделе “Использование служб каталогов DCE” руководства *Administration Guide: Implementation*.

Имя объекта информации маршрутизации (route_obj_name)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию Пустое

Параметры, связанные с данным

- “Каталог в пространстве имен DCE (dir_path_name)” на стр. 480
- “Тип служб каталога (dir_type)” на стр. 479

Этот параметр задает имя записи объекта информации маршрутизации по умолчанию, который будет использоваться при всех попытках обращения программ клиента к серверу DRDA. Он применяется только к средам на основе UNIX и OS/2.

Если значение этого параметра начинается с */.:/* или с */.../*, оно используется, как задано. В остальных случаях для получения полного имени объекта информации маршрутизации оно добавляется к значению параметра *dir_path_name* (или переменной среды DB2DIRPATHNAME).

Для переопределения этого значения по умолчанию можно использовать переменную среды DB2ROUTE.

Этот параметр имеет смысл, только если параметр *dir_type* имеет значение DCE.

Дополнительную информацию смотрите в разделе “Использование служб каталогов DCE” руководства *Administration Guide: Implementation*.

Протокол связи клиента по умолчанию (dft_client_comm)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] Null [Null; TCPIP; APPC; IPXSPX (только для OS/2); NETBIOS (только для OS/2)]

Параметры, связанные с данным

“Тип служб каталога (dir_type)” на стр. 479

Этот параметр задает протоколы связи, которые программы клиента на данном экземпляре могут использовать для удаленных соединений. Его значение - символьная строка из одного или нескольких названий протоколов. Если вы задаете несколько протоколов, разделяйте их запятыми. Порядок протоколов задает порядок предпочтения.

Этот параметр используется только с DCE и применяется только для сред на основе UNIX и OS/2.

Значение этого параметра можно временно переопределить, задав переменную среды DB2CLIENTCOMM.

Если значение этого параметра - NULL, и переменная среды не задана, используется первый протокол, заданный на объекте глобального каталога сервера.

Этот параметр игнорируется, если для *dir_type* задано значение NONE.

Рекомендация: Поставьте на первое место тот протокол, который используется чаще.

Номер адаптера клиента по умолчанию (dft_client_adpt)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 0 [0–15]

Параметры, связанные с данным

- “Протокол связи клиента по умолчанию (dft_client_comm)” на стр. 482.
- “Тип служб каталога (dir_type)” на стр. 479. (Когда dir_type - DCE.)

Этот параметр задает номер адаптера клиента по умолчанию для протокола NETBIOS, для которого nname сервера получается от служб каталогов ячейки DCE (CDS). Этот параметр применяется только в среде OS/2.

Этот параметр используется только с DCE.

Значение этого параметра можно временно переопределить, задав переменную среды DB2CLIENTADPT. Если этот параметр содержит нечисловое или выходящее за границы диапазона значение, используется номер адаптера 0 (ноль).

программа поиска DB2

Следующие параметры служат для задания программы поиска DB2:

- “Поиск баз данных (discover_db)” на стр. 484
- “Режим поиска (discover)” на стр. 484
- “Протоколы связи поиска (discover_comm)” на стр. 485
- “Экземпляр сервера Discover (discover_inst)” на стр. 486

Поиск баз данных (discover_db)

Тип конфигурации База данных

Тип параметра Конфигурируемый

По умолчанию [Диапазон] Enable [Disable, Enable]

Этот параметр используется, чтобы информация о базе данных не возвращалась клиенту, когда требование поиска принимается сервером.

По умолчанию этот параметр разрешает находить базу данных при поиске.

Изменив значение параметра на “Disable”, можно скрыть базы данных с важной информацией от процесса поиска. Это можно сделать в качестве дополнительной меры к другим элементам защиты в базе данных.

Числовые эквиваленты и константы API для этих значений приводятся в книге *Administrative API Reference*.

Режим поиска (discover)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] SEARCH [DISABLE, KNOWN, Search]

Параметры, связанные с данным

“Протоколы связи поиска (discover_comm)” на стр. 485

С точки зрения сервера администратора этот параметр конфигурации определяет тип режима поиска, который запускается при запуске DB2ADMIN.

- Если при запуске DB2ADMIN *discover* = SEARCH, запускаются менеджеры соединений поиска для каждого из протоколов, указанных в *discover_comm*. Кроме того, запускаются менеджеры соединений для каждого из протоколов, указанных в переменной реестра DB2COMM. Это позволяет серверу администратора обслуживать требования поиска ″search″ от клиентов. Тип

SEARCH обеспечивает более широкие функции по сравнению с типом KNOWN. Если *discover* = SEARCH, сервер администратора будет обслуживать требования поиска ″search″ и ″known″ от клиентов.

- Если при запуске DB2ADMIN *discover* = KNOWN, будут запущены только менеджеры соединений, указанные в переменной реестра DB2COMM. Эти менеджеры соединений обрабатывают требования поиска KNOWN.
- Если при запуске DB2ADMIN *discover* = DISABLE, сервер администратора не будет обрабатывать никаких требований поиска.

С точки зрения экземпляра сервера, если *discover* = DISABLE, информация об этом экземпляре сервера скрыта от клиентов. Сервер администратора не отправит информацию об этом экземпляре, когда получит требование поиска ″known″ для этой системы от какого-либо клиента.

С точки зрения клиента возможны следующие случаи:

- Если *discover* = SEARCH, этот клиент может использовать требования поиска ″search″ для поиска систем серверов DB2 в сети. Тип поиска ″search″ обеспечивает более широкие функции по сравнению с типом KNOWN. Если *discover* = SEARCH, клиент может использовать требования поиска ″search″ и ″known″.
- Если *discover* = KNOWN, клиент может использовать только требования поиска ″known″. При указании некоторой информации о соединении для сервера администратора на конкретной системе вся информация об экземплярах и базах данных системы DB2 будет возвращена клиенту.
- Если *discover* = DISABLE, поиск для этого клиента отключен.

По умолчанию используется режим поиска SEARCH.

Числовые эквиваленты и константы API для этих значений приводятся в книге *Administrative API Reference*.

Дополнительную информацию о поиске DB2 смотрите в руководстве *Быстрый старт* для вашей платформы.

Протоколы связи поиска (discover_comm)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] None [Любое сочетание NETBIOS к TCPIP]

Параметры, связанные с данным

“Режим поиска (discover)” на стр. 484

С точки зрения сервера администратора этот параметр определяет менеджеры поиска, запускаемые при запуске DB2ADMIN. Эти менеджеры обслуживают требования поиска от клиентов.

Примечание: Протоколы, определенные в *discover_comm*, должны быть указаны также в переменной реестра DB2COMM.

С точки зрения клиента этот параметр определяет протоколы, которые будут использовать клиенты для требований поиска.

Можно задать несколько протоколов через запятую или оставить параметр пустым.

По умолчанию значение этого параметра - ″None″, то есть протоколы связи для поиска не используются.

Экземпляр сервера Discover (discover_inst)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] ENABLE [ENABLE, DISABLE]

Этот параметр определяет, должна ли программа поиска DB2 находить данный экземпляр. Значение по умолчанию, ENABLE, задает, что экземпляр может быть найден; значение DISABLE задает, что экземпляр найден не будет.

Числовые эквиваленты и константы API для этих значений приводятся в книге *Administrative API Reference*.

Дополнительную информацию о поиске DB2 смотрите в руководстве *Быстрый старт* для вашей платформы.

Многораздельная база данных

Следующие группы параметров содержат информацию о параллельных операциях и среде многораздельных баз данных:

- “Связь”
- “Параллельная обработка” на стр. 493.

Связь

Следующие параметры содержат информацию о связи в среде многораздельных баз данных:

- “Затраченное время соединения (conn_elapse)”
- “Число привязок сообщений FCM (fcm_num_anchors)” на стр. 488
- “Число буферов FCM (fcm_num_buffers)” на стр. 488
- “Число записей соединений FCM (fcm_num_connect)” на стр. 490
- “Число блоков требований FCM (fcm_num_rqb)” на стр. 490
- “Число попыток соединений с узлом (max_connretries)” на стр. 491
- “Максимальная разница времени между узлами (max_time_diff)” на стр. 492
- “Срок ожидания запуска и остановки (start_stop_time)” на стр. 492.

Затраченное время соединения (conn_elapse)

Тип конфигурации Менеджер баз данных

Применяется к Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 10 [0–100]

Единица измерения Секунды

Параметры, связанные с данным

“Число попыток соединений с узлом (max_connretries)” на стр. 491

Этот параметр задает срок (в секундах) установки соединения TCP/IP между двумя серверами разделов баз данных. Если за заданное этим параметром время попытка установки соединения достигает успеха, связь будет установлена. Если установить соединение не удается, будет предпринята другая попытка установки связи. Если число неудачных попыток достигнет значения параметра *max_connretries*, будет выдано сообщение об ошибке.

Число привязок сообщений FCM (fcm_num_anchors)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] -1 [-1, 128–*fcm_num_rqb*]

В одноразделыных системах баз данных параметр *intra_parallel* должен быть активен, чтобы использовать данный параметр.

Параметры, связанные с данным

- “Число блоков требований FCM (fcm_num_rqb)” на стр. 490
- “Разрешение внутриразделыного параллелизма (intra_parallel)” на стр. 494

Привязки сообщений FCM. Агенты используют привязки сообщений для отправки сообщений друг другу. Значение по умолчанию (-1) соответствует 75 процентам от значения, заданного для *fcm_num_rqb*.

Число буферов FCM (fcm_num_buffers)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 512, 1024 или 4096 [128 – *fcm_num_rqb*]

- Сервер баз данных с локальными и удаленными клиентами: значение по умолчанию - 1024
- Сервер баз данных с локальными клиентами: значение по умолчанию - 512
- Сервер многораздельных баз данных с локальными и удаленными клиентами: значение по умолчанию - 4096

В системах одноразделных баз данных параметр *intra_parallel* должен быть активен, чтобы использовался этот параметр.

Этот параметр задает число 4-Кбайтных буферов, которые используются для внутренней связи (сообщений) как между серверами баз данных, так и внутри самих серверов баз данных.

Дополнительную информацию по FCM смотрите в теме “Включение связи FCM” в книге *Administration Guide: Implementation*.

Если на процессоре используется несколько логических узлов, может потребоваться увеличить значение этого параметра. Кроме того, увеличение значения этого параметра может потребоваться при исчерпании буферов сообщений из-за большого числа пользователей системы, большого числа серверов разделов баз данных в системе или сложности программ.

При использовании нескольких логических узлов (кроме систем AIX) один пул буферов *fcm_num_buffers* используется совместно всеми логическими узлами на одном компьютере. В AIX:

- При достаточном объеме общей памяти, используемой менеджером баз данных, из этой памяти будут выделена куча буферов FCM. В этом случае каждый сервер разделов баз данных получит *fcm_num_buffers* собственных буферов; серверы разделов баз данных не будут использовать совместно пул буферов FCM (эта особенность появилась DB2 Версии 5).
- При недостаточном объеме общей памяти, используемой менеджером баз данных, куча буферов FCM будет выделена из отдельной области памяти (совместно используемой памяти AIX), которая используется всеми логическими узлами на одном компьютере. Один пул *fcm_num_buffers* будет совместно использоваться всеми логическими узлами, работающими на одном компьютере. Так же ведут себя и другие системы (не AIX), и DB2 Parallel Edition Версии 1.2 в AIX.

Совет пользователям Parallel Edition в AIX: Если вы используете несколько логических узлов, значение *fcm_num_buffers*, используемое в Parallel Edition Версии 1.2, теперь может привести к значительно большему расходу памяти на

каждом компьютере. Например, конфигурация множества логических узлов с четырьмя узлами может использовать в четыре раза больше буферов FCM, чем прежде.

Перепроверьте используемое вами значение; посмотрите, сколько всего буферов FCM будет выделено на компьютере (или компьютерах), где работают несколько логических узлов. Попробуйте изменить значение *fcm_num_buffers*, чтобы учесть описанное выше поведение.

Число записей соединений FCM (fcm_num_connect)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] -1 [-1, 128 – *fcm_num_rqb*]

В одноразделыных системах баз данных параметр *intra_parallel* должен быть активен, чтобы использовать данный параметр.

Параметры, связанные с данным

"Число блоков требований FCM (fcm_num_rqb)"

Этот параметр задает число серверов *записей соединений* FCM. Агенты используют записи соединений для передачи данных друг другу. Значение по умолчанию (-1) соответствует 75 процентам от значения, заданного для *fcm_num_rqb*.

Число блоков требований FCM (fcm_num_rqb)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

32-битные платформы UNIX
256, 512, 2048 [128 – 120000]

64-битные платформы UNIX
256, 512, 2048 [128 – 524288]

OS/2 и Windows NT
10000 [250 – 2097152]

- Сервер баз данных с локальными и удаленными клиентами: по умолчанию - 512
- Сервер баз данных с локальными клиентами: по умолчанию - 256
- Сервер многораздельных баз данных с локальными и удаленными клиентами: по умолчанию - 2048

В одноразделных системах баз данных параметр *intra_parallel* должен быть активен, чтобы использовать данный параметр.

Этот параметр задает число серверов *блоков требований* FCM. Через блоки требований передается информация между демоном FCM и агентом, или между агентами.

Необходимо число требований зависит от числа пользователей в системе, числа серверов разделов баз данных в системе и сложности обрабатываемых запросов. Начните со значения по умолчанию и используйте для тонкой настройки этого параметра результаты монитора систем баз данных.

Число попыток соединений с узлом (max_connretries)

Тип конфигурации Менеджер баз данных

Применяется к Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 5 [0–100]

Параметры, связанные с данным

“Затраченное время соединения (conn_elapse)” на стр. 487

При неудачной попытке установления связи между двумя серверами разделов баз данных (например, если превышен срок установки соединения, заданный параметром *conn_elapse*) параметр *max_connretries* задает максимальное число попыток соединения с сервером раздела базы данных. Если превышено заданное этим параметром значение, возвращается сообщение об ошибке.

Максимальная разница времени между узлами (max_time_diff)

Тип конфигурации	Менеджер баз данных
Применяется к	Сервер многораздельных баз данных с локальными и удаленными клиентами
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	60 [1–1440]
Единица измерения	Минуты

У каждого сервера раздела базы данных есть свои системные часы. Этот параметр задает максимальную разность времени, допускаемую между серверами разделов баз данных, перечисленными в файле конфигурации узла.

Если с транзакцией связаны несколько серверов разделов, и показания их часов расходятся сильнее, чем на значение этого параметра, транзакция не выполняется и сообщение об ошибке записывается в файл `db2diag.log`. (Транзакция отклоняется, только если она связана с изменением данных.)

DB2 Universal Database Enterprise - Extended Edition использует *согласованное универсальное время* (UTC), поэтому разница часовых поясов в данном случае не играет роли. Согласованное универсальное время - это то же самое, что время по Гринвичу.

Срок ожидания запуска и остановки (start_stop_time)

Тип конфигурации	Менеджер баз данных
Применяется к	Сервер многораздельных баз данных с локальными и удаленными клиентами
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	10 [1 – 1440]
Единица измерения	Минуты

Этот параметр применяется только в в среде многораздельных баз данных. Он задает время в минутах, за которое все серверы разделов баз данных должны ответить на команду DB2START или DB2STOP. Он используется также, как срок ожидания для операции ADDNODE.

Серверы разделов баз данных, которые не ответили на команду DB2START за заданное время, посылают сообщение в журнал ошибок `db2start` в подкаталоге `log` подкаталога `sqllib` домашнего каталога экземпляра. Перед перезапуском таких узлов на них надо выполнить команду DB2STOP.

Серверы разделов баз данных, которые не ответили на команду DB2STOP за заданное время, посылают сообщение в журнал ошибок `db2stop` в подкаталоге `log` подкаталога `sqllib` домашнего каталога экземпляра. Вы можете выполнить команду DB2STOP либо для каждого не ответившего сервера раздела базы данных, либо для всех них. (Уже остановленные серверы сообщат при этом, что они остановлены.)

Параллельная обработка

Следующие параметры содержат информацию о параллельной обработке:

- "Максимальная степень параллелизма запросов (max_querydegree)"
- "Разрешение внутрираздельного параллелизма (intra_parallel)" на стр. 494.

Максимальная степень параллелизма запросов (max_querydegree)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] -1 (ANY) [ANY, 1 – 32767] (ANY означает ″определяемый системой″)

Параметры, связанные с данным

- "Степень параллелизма по умолчанию (dft_degree)" на стр. 469
- "Разрешение внутрираздельного параллелизма (intra_parallel)" на стр. 494

Этот параметр задает максимальную степень внутрираздельного параллелизма для любого оператора SQL, выполняемого на этом экземпляре менеджера баз данных. При выполнении оператора SQL не будет использоваться большее число параллельных операций в разделе, чем задано этим числом. Чтобы

разрешить использование внутриразделного параллелизма на разделе базы данных, нужно задать для параметра конфигурации *intra_parallel* значение “YES”.

По умолчанию для этого параметра конфигурации используется значение -1. Это значение указывает, что система использует степень параллелизма, определенную оптимизатором; в противном случае используется заданное пользователем значение.

Примечание: Степень параллелизма для оператора SQL можно задать во время компиляции оператора, используя специальный регистр CURRENT DEGREE или опцию связывания DEGREE.

Максимальную степень параллелизма запросов для активной прикладной программы можно изменить с помощью команды SET RUNTIME DEGREE. Во время выполнения в качестве степени параллелизма используется наименьшее из следующих значений:
- Значение параметра конфигурации *max_querydegree*
- Степень параллелизма времени выполнения для этой прикладной программы
- Степень параллелизма, заданная при компиляции оператора SQL

Это правило определения используемой степени параллелизма не действует при создании индекса. В этом случае если параметр *intra_parallel* имеет значение “YES” и таблица достаточно велика, чтобы было выгодно использовать несколько процессоров, используемая для создания индекса степень параллелизма равна числу активных процессоров (до 6) плюс один. При этом на степень параллелизма не будут влиять указанные выше параметр, опция связывания или специальный регистр.

Разрешение внутриразделного параллелизма (intra_parallel)

Тип конфигурации Менеджер баз данных

Применяется к
- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] NO (0) [SYSTEM (-1), NO (0), YES (1)]

Значение -1 позволяет системе задать значение “YES” или “NO” на основе аппаратных характеристик среды, где работает менеджер баз данных.

Параметры, связанные с данным

“Максимальная степень параллелизма запросов (max_querydegree)” на стр. 493

Этот параметр задает, может ли менеджер баз данных использовать внутриразделный параллелизм.

Некоторые операции, в том числе запросы к базам данных и создание индексов, могут выполняться быстрее, если задать для этого параметра значение ″YES″.

Примечание: Если значение параметра изменяется, пакеты могут быть пересвязаны с базой данных. При этом при пересвязывании может упасть производительность.

Управление экземпляром

Ряд параметров служит для управления экземплярами менеджера баз данных. Эти параметры разделены на следующие категории:

- “Диагностика”
- “Параметры монитора баз данных” на стр. 498
- “Управление системой” на стр. 500
- “Управление экземпляром” на стр. 508

Диагностика

Следующие параметры служат для управления диагностической информацией, которую может давать менеджер баз данных:

- “Уровень захвата диагностических сообщений (diaglevel)”
- “Путь каталога данных диагностики (diagpath)” на стр. 496
- “Уровень оповещения (notifylevel)” на стр. 497.

Уровень захвата диагностических сообщений (diaglevel)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами

- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 3 [0 – 4]

Параметры, связанные с данным

"Путь каталога данных диагностики (diagpath)"

Этот параметр задает тип ошибок диагностики, записываемых в файл db2diag.log. Возможные значения:

0 – Не захватывать данные диагностики

1 – Только серьезные ошибки

2 – Все ошибки

3 – Все ошибки и предупреждения

4 – Все ошибки, предупреждения и информационные сообщения

Параметр конфигурации *diagpath* используется для задания каталога, куда будет помещен файл ошибок, журнал событий (только в Windows NT), файл журнала оповещений и любых файлов дампа, которые могут быть сгенерированы на основе значения параметра *diaglevel*.

Рекомендация: Можно увеличить значение этого параметра, чтобы собрать дополнительную диагностическую информацию для разрешения проблемы.

Путь каталога данных диагностики (diagpath)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] Пустое [любой допустимое имя пути]

Параметры, связанные с данным

"Уровень захвата диагностических сообщений (diaglevel)" на стр. 495

Этот параметр позволяет задать полный путь для информации диагностики DB2. В этот каталог могут записываться файлы дампов, файлы перехватов, файл журнала ошибок и журнала оповещений в зависимости от платформы.

Если этот параметр имеет значение null, диагностическая информация будет записываться в файлы в следующих каталогах или папках:

- Для OS/2 и поддерживаемых сред Windows:
  - Если переменная среды (или параметр) DB2INSTPROF **не заданы**, информация будет записана в каталог `x:\SQLLIB\DB2INSTANCE`, где `x:\SQLLIB` - буква диска и каталог, заданный в переменной среды или переменной реестра DB2PATH, а `DB2INSTANCE` - имя владельца экземпляра.

    Примечание: Этот каталог нельзя называть SQLLIB.
  - Если переменная среды или параметр DB2INSTPROF **заданы**, информация будет записана в каталог `x:\DB2INSTPROF\DB2INSTANCE`, где DB2INSTPROF - каталог профиля экземпляра, а `DB2INSTANCE` - имя владельца экземпляра.
- Для сред на основе UNIX: `INSTHOME/sqllib/db2dump`, где `INSTHOME` - начальный каталог владельца экземпляра.
- Для среды Macintosh: папка DB2.

Рекомендация: Используйте значение по умолчанию или же задайте единый diagpath для нескольких экземпляров.

В среде многораздельной базы данных задаваемый путь должен находиться в совместно используемой файловой системе.

Уровень оповещения (notifylevel)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами в Windows NT
- Клиент в Windows NT
- Сервер баз данных с локальными клиентами в Windows NT
- Сервер многораздельных баз данных с локальными и удаленными клиентами в Windows NT
- Сервер сателлитных баз данных с удаленными клиентами в Windows 95, Windows 98 и Windows NT

Тип параметра Конфигурируемый

По умолчанию [Диапазон] 2 [0 – 4]

Этот параметр задает тип оповещающих сообщений об ошибках, записываемых в файл. Для сервера узла сателлитного типа ошибки записываются в файл оповещения *instance*.nfy . Для узлов других типов этот параметр доступен только на платформе Windows NT, и ошибки записываются в журнал Windows NT. Эти ошибки могут записывать DB2, программы Capture и Apply, а также пользовательские программы.

Допустимые значения этого параметра:

0 — Не захватывать данные диагностики

1 — Только серьезные ошибки

2 — Все ошибки

3 — Все ошибки и предупреждения

4 — Все ошибки, предупреждения и информационные сообщения

Чтобы пользовательская программа могла записывать в файл оповещений или в журнал событий Windows NT, она должна вызывать API db2AdminMsgWrite. Дополнительную информацию об этом API смотрите в руководстве *Administrative API Reference*.

Рекомендация: Можно увеличить значение этого параметра, чтобы собрать дополнительную диагностическую информацию для разрешения проблемы.

Параметры монитора баз данных

Следующий параметр позволяет управлять различными аспектами монитора баз данных:

- "Переключатели системного монитора баз данных по умолчанию (dft_monswitches)"

Переключатели системного монитора баз данных по умолчанию (dft_monswitches)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию Все переключатели выключены

Этот параметр уникален, поскольку позволяет включить любое число переключателей, каждый из которых внутренне представляется битом данного параметра. В зависимости от интерфейса, используемого для изменения конфигурации менеджера баз данных, может существовать возможность изменять этот параметр непосредственно. Кроме того, можно независимо изменить любой из этих переключателей, задав следующие параметры:

dft_mon_uow	Положение по умолчанию переключателя единицы работы (UOW) для монитора снимков
dft_mon_stmt	Положение по умолчанию переключателя оператора для монитора снимков
dft_mon_table	Положение по умолчанию переключателя таблицы для монитора снимков
dft_mon_bufpool	Положение по умолчанию переключателя пула буферов для монитора снимков
dft_mon_lock	Положение по умолчанию переключателя блокировок для монитора снимков
dft_mon_sort	Положение по умолчанию переключателя сортировок для монитора снимков

Изменение положений любых из этих переключателей системного монитора баз данных вступает в силу немедленно, то есть не требуется остановка и перезапуск менеджера баз данных.

Примечание: Существующая программа мониторинга не будет автоматически использовать для этих переключателей новые значения по умолчанию. Чтобы новые значения использовались, эта программа должна отсоединиться от экземпляра и вновь соединиться с ним.

Дополнительную информацию о мониторе снимков и использовании им переключателей монитора смотрите в книге *System Monitor Guide and Reference*.

Рекомендация: Любой переключатель во включенном положении (ON) инструктирует менеджер баз данных собрать данные слежения, связанные с этим переключателем. Сбор дополнительных данных монитора увеличивает затраты менеджера баз данных, что может ухудшить производительность системы.

Каждая программа мониторинга наследует эти установки переключателей по умолчанию, когда выдает свое первое требование мониторинга (например, положение переключателя, активация монитора событий, снятие снимка). Включать переключатель в файле конфигурации следует только в том случае, если нужно собрать данные, начав с момента запуска менеджера баз данных.

(Иначе каждая программа мониторинга может включить свои собственные переключатели, и собираемые ею данные будут задаваться переключателями этой программы.)

Управление системой

Следующие параметры относятся к управлению системой:

Пропускная способность связи (comm_bandwidth)

Тип конфигурации	Менеджер баз данных
Применяется к	Сервер многораздельных баз данных с локальными и удаленными клиентами
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	-1 [.1 – 100000] Значение -1 вызывает сброс параметра к значению по умолчанию. Значение по умолчанию вычисляется на основе того, используется ли высокоскоростной переключатель.
Единица измерения	Мегабиты в секунду

Значение определяет пропускную способность линии связи в мегабитах в секунду и используется оптимизатором SQL для оценки стоимости определенных операций между серверами разделов базы данных в системе многораздельных баз данных. Оптимизатор не моделирует стоимость связи между клиентом и сервером, так что этот параметр должен отражать только номинальную пропускную способность между серверами разделов базы данных, если она задана.

Это значение можно задать явно, чтобы смоделировать производственную среду в системе тестирования или чтобы определить влияние модернизации аппаратного обеспечения.

Рекомендация: Изменяйте этот параметр, только если нужно смоделировать другую среду.

Пропускная способность связи используется оптимизатором для определения путей доступа. После изменения этого параметра, возможно, следует выполнить повторное связывание прикладных программ (используя команду REBIND PACKAGE).

Скорость процессора (cpuspeed)

Тип конфигурации	Менеджер баз данных
Применяется к	• Сервер баз данных с локальными и удаленными клиентами • Сервер баз данных с локальными клиентами • Сервер многораздельных баз данных с локальными и удаленными клиентами • Сервер сателлитных баз данных с удаленными клиентами
Тип параметра	Конфигурируемый
По умолчанию [Диапазон]	-1 [1^{-10} – 1] Если задано значение -1, этот параметр автоматически вычисляется заново тестовой программой.
Единица измерения	Секунды

Значение скорости процессора (в миллисекундах на команду) используется оптимизатором SQL для оценки затрат на выполнение конкретных операций. При установке менеджера баз данных значение этого параметра задается автоматически на основе результатов работы программы, измеряющей скорость процессора. Эта программа выполняется, если результаты измерения производительности недоступны по одной из следующих причин:
- На платформе нет поддержки файла `db2spec.dat`
- Файл `db2spec.dat` **не** найден
- В этом файле не найдены данные для IBM RISC System/6000 модели 530H
- В этом файле не найдены данные для вашего компьютера.

Это значение можно задать явно, чтобы смоделировать производственную среду в системе тестирования или чтобы определить влияние модернизации аппаратного обеспечения. Если задать значение -1, значение параметра *cpuspeed* будет измерено заново.

Рекомендация: Изменяйте этот параметр, только если нужно смоделировать другую среду.

Скорость процессора используется оптимизатором при определении путей доступа. После изменения этого параметра, возможно, следует выполнить повторное связывание прикладных программ (используя команду REBIND PACKAGE).

Максимальное число одновременно активных баз данных (numdb)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

UNIX 8 [1 – 256]

OS/2 и Windows NT, Сервер баз данных с локальными и удаленными клиентами
8 [1 – 256]

OS/2 и Windows NT, Сервер баз данных с локальными клиентами и Сервер сателлитных баз данных с удаленными клиентами
3 [1 – 256]

Единица измерения Счетчик

Этот параметр задает число локальных баз данных, которые могут быть активными (то есть иметь подключенные к ним программы) одновременно. В среде многораздельных баз данных этот параметр ограничивает число активных разделов баз данных на сервере разделов базы данных, независимо от того, является ли данный сервер узлом координатора или нет.

Поскольку каждая база данных требует места, а активная база данных использует новый сегмент совместной памяти, вы можете ограничить использование системных ресурсов, задав предельное число отдельных баз данных на вашем компьютере. Однако мы не рекомендуем механически снижать число баз данных. Объединение всех данных, независимо от связей между ними, в одну базу сокращает использование места на диске, однако неудобно для работы. Обычно правильнее объединять в одну базу только функционально связанную между собой информацию.

Рекомендация: Обычно лучше задать это значение равным реальному числу баз, уже определенных в менеджере баз данных, плюс некоторый запас для роста на ближайший период (скажем, полгода или год). Реальное увеличение не должно быть особенно велико, однако это позволит вам добавлять новые базы, не меняя часто этот параметр.

Изменение значения параметра *numdb* может влиять на общий объем выделяемой памяти. Часто менять значение этого параметра не рекомендуется. Изменив его, надо рассмотреть другие параметры, влияющие на выделение памяти для базы данных или для программы, соединенной с базой данных, в том числе:

- “Размер пула буферов (buffpage)” на стр. 366
- “Максимальная память для списка блокировок (locklist)” на стр. 375
- “Размер кучи прикладных программ (applheapsz)” на стр. 385
- “Размер кучи управления прикладной программы (app_ctl_heap_sz)” на стр. 379
- “Размер кучи сортировки (sortheap)” на стр. 381
- “Размер кучи операторов (stmtheap)” на стр. 385
- “Размер кучи слоя поддержки программ (aslheapsz)” на стр. 395
- “Куча базы данных (dbheap)” на стр. 369
- “Размер кучи монитора баз данных (mon_heap_sz)” на стр. 401
- “Размер кучи статистики (stat_heap_sz)” на стр. 386

Имя монитора процессора транзакций (tp_mon_name)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию Значения по умолчанию нет

Допустимые значения

- CICS
- MQ
- ENCINA
- CB
- SF

- TUXEDO
- TOPEND
- пусто или некоторые другие значения (для UNIX, OS/2 и Windows NT; для Solaris и SINIX других значений нет)

Этот параметр задает имя используемого монитора обработки транзакций (TP).

- Если прикладные программы выполняются в среде WebSphere Enterprise Edition CICS, этот параметр должен иметь значение “CICS”
- Если прикладные программы выполняются в среде WebSphere Enterprise Edition Encina, этот параметр должен иметь значение “ENCINA”
- Если прикладные программы выполняются в среде WebSphere Enterprise Edition Component Broker, этот параметр должен иметь значение “CB”
- Если прикладные программы выполняются в среде IBM MQSeries, этот параметр должен иметь значение “MQ”
- Если прикладные программы выполняются в среде BEA Tuxedo, этот параметр должен иметь значение “TUXEDO”
- Если прикладные программы выполняются в среде IBM San Francisco, этот параметр должен иметь значение “SF”.

Пользователи **IBM WebSphere EJB** и **Microsoft Transaction Server** не должны конфигурировать значения этого параметра.

Если не используется ни один из перечисленных продуктов, параметр не следует конфигурировать; его надо оставить пустым.

В предыдущих версиях DB2 Universal Database в средах OS/2 и Windows NT этот параметр содержал путь и имя библиотеки DLL, содержащие функции XA Transaction Manager *ax_reg* и *ax_unreg*. Этот формат по-прежнему поддерживается. Если значение параметра не соответствует никакому из перечисленных выше имен мониторов TP, предполагается, что это имя библиотеки, которая содержит функции *ax_reg* и *ax_unreg*. Это правило используется для сред UNIX, OS/2 и Windows NT.

Пользователям TXSeries CICS и Encina: В предыдущих версиях этого продукта в OS/2 и Windows NT требовалось задавать этот параметр как “libEncServer:C” или “libEncServer:E”. Этот вариант поддерживается, однако теперь это не обязательно. Достаточно задать параметр как “CICS” или “ENCINA”.

Пользователям MQSeries: В предыдущих версиях этого продукта в OS/2 и Windows NT требовалось задавать этот параметр как “mqmax”. Этот вариант поддерживается, однако теперь это не обязательно. Достаточно задать параметр как “MQ”.

Пользователям Component Broker: В предыдущих версиях этого продукта в OS/2 и Windows NT требовалось задавать этот параметр как “somtrx1i”. Этот вариант поддерживается, однако теперь это не обязательно. Достаточно задать параметр как “CB”.

Пользователям San Francisco: В предыдущих версиях этого продукта в OS/2 и Windows NT требовалось задавать этот параметр как “ibmsfDB2”. Этот вариант поддерживается, однако теперь это не обязательно. Достаточно задать параметр как “SF”.

Максимальная длина значения этого параметра - 19 символов.

Можно также задать эту информацию в строке DB2 Universal Database XA OPEN. Если несколько мониторов обработки транзакций используют один экземпляр DB2, использование этой возможности обязательно. Дополнительную информацию о применении строки XA OPEN смотрите в руководстве *Administration Guide: Planning*.

Тип узла компьютера (nodetype)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Информационный

Этот параметр содержит информацию о продуктах DB2, установленных на данном компьютере, и, соответственно, информацию о типе конфигурации менеджера баз данных. Ниже показаны возможные значения, возвращаемые этим параметром, и продукты, связанные с каждым типом узла:

- **Сервер баз данных с локальными и удаленными клиентами** – продукт сервера DB2, поддерживающий локальные и удаленные клиенты баз данных и способный обращаться к другим удаленным серверам баз данных.
- **Клиент** – клиент базы данных, способный обращаться к удаленным серверам баз данных.
- **Сервер баз данных с локальными клиентами** – СУБД DB2, поддерживающая локальные клиенты базы данных и способная обращаться к другим, удаленным серверам баз данных.
- **Сервер многораздельных баз данных с локальными и удаленными клиентами** – продукт сервера DB2, поддерживающий локальные и удаленные клиенты баз

данных, способный обращаться к другим удаленным серверам баз данных и поддерживающий параллелизм разделов.
- **Сервер сателлитных баз данных с удаленными клиентами** – СУБД DB2, поддерживающая локальные клиенты базы данных и способная обращаться к другим, удаленным серверам баз данных.

Числовые эквиваленты и константы API для этих значений приводятся в книге *Administrative API Reference*.

Счет оплаты по умолчанию (dft_account_str)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] Пустое [любая допустимая строка]

Для любого требования на соединение от программы учетный идентификатор из префикса, генерируемого DB2 Connect, и суффикса, задаваемого пользователем, посылается от реквестера прикладных программ сервера прикладных программ DRDA. Эта учетная информация дает системным администраторам способ связывать использование ресурсов с каждым обращением пользователя.

Примечание: Этот параметр применяется только для DB2 Connect.

Прикладная программа определяет суффикс, вызывая API `sqlesact()` или читая заданное пользователем значение переменной среды DB2ACCOUNT. Если суффикс не задан ни API, ни переменной среды, DB2 Connect использует в качестве суффикса по умолчанию значение данного параметра. Этот параметр особенно полезен для клиентов ранних версий (до версии 2), которые не могут направлять DB2 Connect учетную строку.

Рекомендация: Используйте в учетной строке следующие символы:
- Латинские буквы (от A до Z)
- Цифры (от 0 до 9)
- Символ подчеркивания (_).

Путь установки Java Development Kit 1.1 (jdk11_path)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] Null [Допустимый путь]

Параметры, связанные с данным

- “Максимальный размер кучи интерпретатора Java (java_heap_sz)” на стр. 405

Этот параметр задает каталог, где установлен Java Development Kit 1.1. По его значению вычисляются CLASSPATH и другие переменные среды, используемые интерпретатором Java.

Поскольку у этого параметра нет значения по умолчанию, вы должны задать его при установке Java Development Kit.

Поддержка систем баз данных объединения (federated)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] No [Yes; No]

Этот параметр разрешает или запрещает поддержку программ, выдающих распределенные требования для данных, управляемых источниками данных (например, семейства DB2 и Oracle).

Управление экземпляром

Следующие параметры относятся к защите и управлению экземпляром менеджера баз данных:

- “Имя группы с правами администратора системы (sysadm_group)”
- “Имя группы с правами управления системой (sysctrl_group)” на стр. 509
- “Имя группы с правами обслуживания системы (sysmaint_group)” на стр. 510
- “Тип аутентификации (authentication)” на стр. 511
- “Разрешение каталогизации без полномочий (catalog_noauth)” на стр. 513
- “Путь баз данных по умолчанию (dftdbpath)” на стр. 513
- “LOGON, требуемый для DB2START/DB2STOP (ss_logon)” на стр. 515
- “Доверять всем клиентам (trust_allclnts)” на стр. 515
- “Аутентификация доверенных клиентов (trust_clntauth)” на стр. 516

Имя группы с правами администратора системы (sysadm_group)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию Пустое

Параметры, связанные с данным

- “Имя группы с правами управления системой (sysctrl_group)” на стр. 509
- “Имя группы с правами обслуживания системы (sysmaint_group)” на стр. 510

Полномочия администратора системы (SYSADM) - это высший уровень прав для менеджера баз данных, позволяющий управлять всеми объектами базы данных. Этот параметр определяет имя группы с полномочиями системного администратора (SYSADM) для экземпляра менеджера баз данных.

Полномочия SYSADM определяются возможностями защиты, применяемыми в конкретной операционной среде.

- В операционных системах Windows 95 и Windows 98 этот параметр должен иметь значение NULL.

Для клиентов Windows 95 и Windows 98, когда используется системная защита, этот параметр должен иметь значение “NULL”, поскольку в Windows 95 и Windows 98 информация о группе не хранится, и нет возможностей определить, является ли пользователь членом заданной группы SYSADM. Если имя группы задано, никакой пользователь не будет считаться ее членом.

- Для операционных систем Windows NT и Windows 2000 значением этого параметра может быть имя любой локальной группы не длиннее 8 символов, и эта группа должна быть определена в базе защиты Windows NT или Windows 2000. Если значение этого параметра - “NULL” полномочия SYSADM будут иметь все члены группы администраторов.
- Для систем на базе UNIX, если значение этого параметра - “NULL”, группа SYSADM по умолчанию - это первичная группа владельца экземпляра.

  Если значение - не “NULL”, именем группы SYSADM может быть любое допустимое имя группы UNIX.
- В OS/2, если для этого параметра задано значение “NULL”, полномочия SYSADM будут иметь пользователи, заданные как администраторы в программе управления профилями пользователей (UPM).

  Если значение этого параметра - имя группы, полномочия SYSADM будут иметь только члены этой группы SYSADM. Указанная группа может быть любой группой управления профилями пользователей.

Если используется защита DCE и *sysadm_group* имеет значение “NULL”, используется имя группы DCE по умолчанию - DB2ADMIN. При этом должен уже существовать правильный принципал DCE, authid которого отображается на DB2ADMIN. Можно задать и другое имя группы.

Чтобы задать для параметра значение по умолчанию (NULL), используйте оператор UPDATE DBM CFG USING SYSADM_GROUP NULL. Ключевое слово “NULL” надо задавать в верхнем регистре. Можно использовать также записную книжку Конфигурирование экземпляра в Центре управления DB2.

Имя группы с правами управления системой (sysctrl_group)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию Пустое

Параметры, связанные с данным

- “Имя группы с правами администратора системы (sysadm_group)” на стр. 508
- “Имя группы с правами обслуживания системы (sysmaint_group)”

Этот параметр определяет имя группы с полномочиями управления системой (SYSCTRL). SYSCTRL дает привилегии, позволяющие воздействовать на системные ресурсы, однако не разрешает прямого доступа к данным.

Внимание: Для клиентов Windows 95 и Windows 98, когда используется системная защита (то есть при авторизации CLIENT, SERVER, или DCS или других допустимых вариантах аутентификации), этот параметр должен иметь значение NULL. Это объясняется тем, что Windows 95 и Windows 98 не хранят информацию о группе, поэтому нет возможностей определить, является ли пользователь членом заданной группы SYSCTRL. Если имя группы задано, никакой пользователь не будет считаться ее членом. Когда используется аутентификация DCE, это не так. В такой ситуации можно указывать имена групп.

Чтобы задать для параметра значение по умолчанию (NULL), используйте оператор UPDATE DBM CFG USING SYSCTRL_GROUP NULL. Ключевое слово “NULL” надо задавать в верхнем регистре. Можно также использовать записную книжку Конфигурирование экземпляра в Центре управления DB2.

Имя группы с правами обслуживания системы (sysmaint_group)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию Пустое

Параметры, связанные с данным

- “Имя группы с правами администратора системы (sysadm_group)” на стр. 508

- “Имя группы с правами управления системой (sysctrl_group)” на стр. 509

Этот параметр определяет имя группы с полномочиями обслуживания системы (SYSMAINT). SYSMAINT дает привилегии для выполнения обслуживающих операций для всех баз данных, связанных с экземпляром, однако не позволяет обращаться непосредственно к данным.

Внимание: Для клиентов Windows 95 и Windows 98, когда используется системная защита (то есть при авторизации CLIENT, SERVER, или DCS или других допустимых вариантах аутентификации), этот параметр должен иметь значение NULL. Это объясняется тем, что Windows 95 и Windows 98 не хранят информацию о группе, поэтому нет возможностей определить, является ли пользователь членом заданной группы SYSMAINT. Если имя группы задано, никакой пользователь не будет считаться ее членом. Когда используется аутентификация DCE, это не так. В такой ситуации можно указывать имена групп.

Чтобы задать для параметра значение по умолчанию (NULL), используйте оператор UPDATE DBM CFG USING SYSMAINT_GROUP NULL. Ключевое слово “NULL” надо задавать в верхнем регистре. Можно также использовать записную книжку Конфигурирование экземпляра в Центре управления DB2.

Тип аутентификации (authentication)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] SERVER [CLIENT; SERVER; SERVER_ENCRYPT; DCS; DCS_ENCRYPT; DCE; DCE_SERVER_ENCRYPT; KERBEROS; KRB_SERVER_ENCRYPT]

Этот параметр определяет, как и когда производится аутентификация пользователя.

Если значение - SERVER, ID пользователя и пароль посылаются с клиента на сервер, и аутентификация проводится на сервере. Значение SERVER_ENCRYPT определяет то же поведение, что и SERVER, но все передаваемые по сети пароли шифруются.

Значение CLIENT задает, что вся аутентификация производится на клиенте, и на сервере проводить аутентификацию не нужно.

Значение DCS задает, что аутентификация производится на системе хоста или AS/400. Значение DCS_ENCRYPT определяет то же поведение, что и DCS, но все передаваемые по сети пароли шифруются. Если вы используете APPC и продукт, обеспечивающий связь, не сообщает пароль клиента серверу DB2, можно указать DCS и получить:
- Аутентификацию типа CLIENT для клиентов DRDA
- Аутентификацию типа SERVER для прочих клиентов

Значение DCE означает, что аутентификация выполняется на сервера DCE при помощи служб защиты DCE. Значение DCE_SERVER_ENCRYPT определяет то же поведение, что и DCE, но все передаваемые по сети пароли шифруются. Значение DCE_SERVER_ENCRYPT используется только на сервере. Оно указывает, что сервер может принимать либо аутентификацию DCE, либо аутентификацию SERVER_ENCRYPT.

Значение KERBEROS означает, что аутентификация выполняется на сервере Kerberos и для аутентификации применяется протокол защиты Kerberos. При типе аутентификации KRB_SERVER_ENCRYPT на сервере и на клиентах которые поддерживают систему защиты Kerberos реальным типом аутентификации в системе будет KERBEROS. Если же клиенты не поддерживают систему защиты Kerberos, реальный тип аутентификации в системе будет эквивалентен SERVER_ENCRYPT.

Примечание: Типы аутентификация Kerberos поддерживаются только на серверах, работающих под Windows 2000.

К типам аутентификации, которые поддерживают шифрование пароля, относятся: SERVER_ENCRYPT, DCS_ENCRYPT, DCE_SERVER_ENCRYPT и KRB_SERVER_ENCRYPT. Эти значения обеспечивают те же самые функции, что и SERVER, DCS, DCE и KERBEROS соответственно в отношении места аутентификации, но все пароли, передаваемые по сети, шифруются в месте отправления и требуют дешифровки в месте приема, как указано типом аутентификации, занесенным в каталог в месте отправления. Значения с шифрованием и без шифрования с соответствующими местами аутентификации можно использовать для выбора различных вариантов использования шифрования для связи между клиентом и шлюзом или между шлюзом и сервером независимо от места аутентификации.

Числовые эквиваленты и константы API для этих значений приводятся в книге *Administrative API Reference*.

Дополнительную информацию о том, когда и при каких условиях использовать DCE или DCS, и о аутентификации для баз данных объединения смотрите в главе “Controlling Database Access” руководства *Administration Guide: Implementation*.

Рекомендация: Обычно следует использовать значение по умолчанию (SERVER). Если ваши входящие требования обрабатываются Kerberos, DB2 Connect или DCE, посмотрите главу “Controlling Database Access” руководства *Administration Guide: Implementation*.

Разрешение каталогизации без полномочий (catalog_noauth)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Клиент
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

Сервер баз данных с локальными и удаленными клиентами; Сервер баз данных с локальными и удаленными клиентами
NO [NO (0) – YES (1)]

Клиент; Сервер баз данных с локальными клиентами; Сервер сателлитных баз данных с удаленными клиентами
YES [NO (0) – YES (1)]

Этот параметр задает, могут ли пользователи, не обладающие полномочиями SYSADM, вносить в каталог и удалять из каталога базы данных и узлы или каталоги DCS и ODBC. Значение этого параметра по умолчанию (0) указывает, что требуются полномочия SYSADM. Если этот параметр имеет значение 1 (YES), полномочия SYSADM не требуются.

Путь баз данных по умолчанию (dftdbpath)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами
- Сервер сателлитных баз данных с удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон]

UNIX Начальный каталог владельца экземпляра [любой существующий путь]

OS/2 и Windows NT
Диск, на котором установлена система DB2 [любой существующий путь]

Этот параметр содержит путь файлов по умолчанию, используемый при создании баз данных этим менеджером баз данных. Если при создании базы данных не задан путь, база данных создается в пути, заданном параметром *dftdbpath*.

В среде многораздельных баз данных путь, в котором создается база данных, не должен быть смонтированным путем NFS (на платформах на основе UNIX) или сетевым диском (в среде Windows NT). Заданный путь должен физически существовать на каждом сервере раздела базы данных. Чтобы избежать неоднозначности, лучше всего задавать путь, смонтированный локально на каждом сервере раздела базы данных. Максимальная длина пути - 205 символов. Система добавляет имя узла в конец пути.

Если базы данных могут достигать большого размера и многие пользователи могут создавать базы данных (в зависимости от среды и назначения системы), часто удобно, чтобы все базы данных создавались и хранились в заданном месте. Также хорошо иметь возможность изолировать базы данных от других прикладных программ и данных (как для обеспечения целостности, так и для резервного копирования и восстановления).

В средах на основе UNIX длина имени *dftdbpath* не может превышать 215 символов; это имя должно быть правильным абсолютным именем пути. Для OS/2 и Windows NT имя *dftdbpath* может быть буквой диска (после которого может идти необязательное двоеточие).

Рекомендация: По возможности помещайте базы данных большого объема на диски, на которых нет других часто используемых данных (таких как файлы операционной системы и журналы баз данных).

LOGON, требуемый для DB2START/DB2STOP (ss_logon)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] YES [NO (0), YES (1)]

Этот параметр применяется только в среде OS/2. Если принять значение по умолчанию для данного параметра, перед выдачей команд DB2START или DB2STOP требуется регистрация с указанием ID пользователя и пароля.

Доверять всем клиентам (trust_allclnts)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] YES [NO, YES, DRDAONLY]

Параметры, связанные с данным

- “Тип аутентификации (authentication)” на стр. 511
- “Аутентификация доверенных клиентов (trust_clntauth)” на стр. 516

Этот параметр активен, только если для параметра *authentication* задано значение CLIENT.

Этот параметр и параметр *trust_clntauth* используются для задания места проверки пользователей в среде базы данных.

Если принять для этого параметра значение по умолчанию “YES”, все клиенты будут рассматриваться как доверенные. Это означает, что сервер считает, что уровень защиты устанавливается на клиенте, и проверка пользователей может проводиться на клиенте.

Этот параметр можно изменить “NO”, если для параметра *authentication* задано значение CLIENT. Если этот параметр имеет значение “NO”, недоверенные клиенты при соединении с сервером должны сообщать и ID пользователя, и пароль. Недоверенные клиенты - это платформы с операционными системами, где нет подсистем защиты для идентификации пользователей.

Задание значения “DRDAONLY” для этого параметра защищает от всех клиентов, кроме клиентов DRDA из DB2 for MVS and OS/390, DB2 for VM and VSE и DB2 for OS/400. Только этим клиентам будет доверяться проведение аутентификации на стороне клиента. Все остальные клиенты должны проходить аутентификацию на сервере, предоставляя ID пользователя и пароль.

Когда параметр *trust_allclnts* имеет значение “DRDAONLY”, параметр *trust_clntauth* задает, где проводится аутентификация клиентов. Если *trust_clntauth* имеет значение “CLIENT”, аутентификация производится на клиенте. Если *trust_clntauth* имеет значение “SERVER”, аутентификация проводится на клиенте, если пароль не задан, и на сервере, если пароль задан.

Дополнительную информацию о доверенных клиентах смотрите в разделе “Выбор метода аутентификации для сервера” в книге *Administration Guide: Implementation*.

Аутентификация доверенных клиентов (trust_clntauth)

Тип конфигурации Менеджер баз данных

Применяется к

- Сервер баз данных с локальными и удаленными клиентами
- Сервер баз данных с локальными клиентами
- Сервер многораздельных баз данных с локальными и удаленными клиентами

Тип параметра Конфигурируемый

По умолчанию [Диапазон] CLIENT [CLIENT, SERVER]

Параметры, связанные с данным

- “Тип аутентификации (authentication)” на стр. 511
- “Доверять всем клиентам (trust_allclnts)” на стр. 515

Этот параметр задает, будет ли доверенный клиент проверяться на сервера или же на клиенте, если клиент передает ID пользователя и пароль для соединения. Этот параметр (как и *trust_allclnts*) активен, только если параметр *authentication* имеет значение CLIENT. Если ID пользователя и пароль не заданы, проверка пользователя предполагается проведенной на клиенте, и дополнительная проверка на сервер не производится.

Если этот параметр имеет значение CLIENT (по умолчанию), доверенный клиент может устанавливать соединение, не передавая ID пользователя и пароль; предполагается, что операционная система уже проверила пользователя. Если этот параметр имеет значение SERVER, ID пользователя и пароль будут проверены на сервере.

Числовое значение для CLIENT - 0. Числовое значение для SERVER - 1.

Дополнительную информацию о доверенных клиентах смотрите в разделе “Выбор метода аутентификации для сервера” в книге *Administration Guide: Implementation*.

Часть 4. Приложения

Приложение A. Переменные реестра и среды DB2

Ниже приводится список переменных реестра и переменных среды DB2, которые могут понадобиться вам при работе с системой. К каждой переменной дается краткое описание; некоторые переменные могут быть неприменимы в вашей среде.

Список всех поддерживаемых переменных реестра можно получить, введя команду:

```
db2set -lr
```

Значение переменной можно изменить для текущего экземпляра или экземпляра по умолчанию командой:

```
db2set имя_переменной_реестра=новое_значение
```

Чтобы изменить переменные среды, нужно выполнить команду `set` и перезагрузить систему.

Значения измененных переменных реестра нужно задавать до выполнения команды DB2START. Дополнительную информацию об изменении и использовании переменных реестра смотрите в руководстве *Administration Guide: Implementation*.

Значения, используемые в описаниях двоичных переменных реестра, имеют эквивалентные. Значения YES, 1 и ON считаются эквивалентными. Аналогичным образом эквивалентны значения NO, 0 и OFF. Эквивалентные значения взаимозаменяемы.

Таблица 21. Основные переменные реестра

Имя переменной **Описание**	**Операционная система**	**Значения**
DB2ACCOUNT	Все	По умолчанию=null
Учетная строка, которая посылается на удаленный хост. Дополнительную информацию смотрите в руководстве *DB2 Connect. Руководство пользователя*.		
DB2BIDI	Все	По умолчанию=NO Значения: YES или NO

Таблица 21. Основные переменные реестра (продолжение)

Имя переменной	Операционная система	Значения
Описание		
Эта переменная включает поддержку языка с двумя направлениями письма, а переменная DB2CODEPAGE задает используемую кодовую страницу. Дополнительную информацию о поддержке языков с двумя направлениями письма смотрите в приложении Поддержка национальных языков руководства *Administration Guide: Planning*.		
DB2_BLOCK_ON_LOG_DISK_FULL	Все	По умолчанию=No Значения: Yes или No
С помощью этой переменной реестра DB2 можно предотвратить генерирование сообщений об ошибках заполнения диска (″disk full″), когда DB2 не может создать новый файл журнала по действующему пути журналов. Вместо этого DB2 будет пытаться создавать файл журнала каждые 5 минут, пока операция не завершится успешно. После каждой попытки DB2 записывает сообщение в файл `db2diag.log`. Единственный способ убедиться в том, что программа зависает из-за переполнения диска - это следить за файлом `db2diag.log`. Пока файл журнала не будет создан успешно, ни одна пользовательская программа, пытающаяся изменить табличные данные, не сможет осуществить принятие транзакций. На запросы только для чтения нельзя воздействовать непосредственно, однако если запросу нужен доступ к данным, которые блокируются требованием изменения или к странице данных которая исправляется в пуле буферов изменяющей программой, запросы только для чтения, скорее всего, также зависнут.		
DB2CODEPAGE	Все	По умолчанию: определяется ID языка, заданного операционной системой.
Задает кодовую страницу данных, передаваемых DB2 для клиентской программы. Пользователь должен изменять DB2CODEPAGE, только если это прямо указано в документации DB2 или предложено службой DB2. Если изменить DB2CODEPAGE на значение, не поддерживаемое операционной системой, результаты могут быть непредсказуемы. В нормальной ситуации задавать значение DB2CODEPAGE не нужно, так как DB2 автоматически получает у операционной системы информацию о кодовой странице.		
DB2COUNTRY	Все	По умолчанию: определяется по ID языка, заданного операционной системой.
Задает код страны, региона или области для клиентской программы, что влияет на форматы даты и времени.		
DB2DBDFT	Все	По умолчанию=null
Задает алиас базы данных, с которой будет происходить неявное соединение. Если программа не соединена с базой данных, но при этом выполняет операторы SQL, будет выполнено неявное соединение с базой данных по умолчанию, заданной переменной среды DB2DBDFT.		
DB2DBMSADDR	32-битные операционные системы Windows	По умолчанию = 0x20000000 для Windows NT, 0x90000000 для Windows 95 Значение: от 0x20000000 до 0xB0000000 с шагом 0x10000

Таблица 21. Основные переменные реестра (продолжение)

Имя переменной	Операционная система	Значения
Описание		
Задает адрес совместно используемой памяти менеджера баз данных по умолчанию в шестнадцатеричном формате. Если *db2start* завершается неудачно из-за пересечения адресов совместно используемой памяти, можно изменить эту переменную реестра, чтобы экземпляр менеджера баз данных разместил свою совместно используемую память по другому адресу.		
DB2_DISABLE_FLUSH_LOG	Все	По умолчанию= OFF Значение: ON или OFF
Задает, хотите ли вы отключить закрытие активного файла журнала в момент завершения оперативного резервного копирования. При завершении оперативного резервного копирования последний активный файл журнала усекается, закрывается и становится доступным для архивирования. В результате у резервной копии будет полный комплект архивных журналов для ее восстановления. Вы можете отключить закрытие последнего активного файла журнала, если для вас важно сохранение пространства последовательных номеров журналов (Log Sequence Number, LSN). Каждый раз при усечении активного файла журнала LSN увеличивается на объем, пропорциональный усеченному пространству. Если вы каждый день делаете много оперативных резервных копий, вы можете отключить закрытие активного файла журнала. Можно также отключить закрытие последнего активного файла журнала, если через небольшое время после завершения резервного копирования вы получаете сообщения о том, что журнал заполнен. При усечении файла журнала зарезервированное пространство активного файла журнала увеличивается на объем, пропорциональный размеру усеченного журнала. Пространство активного журнала освобождается, как только усеченный файл журнала возвращается в систему. Это происходит вскоре после того, как файл журнала перестает быть активным. Вы можете получать сообщения о том, что журнал заполнен, во время этого краткого отрезка времени.		
DB2DISCOVERYTIME	OS/2 и 32-битные операционные системы Windows	По умолчанию=40 секунд, Минимум=20 секунд
Задает время, в течение которого будет проводиться поиск SEARCH систем DB2.		
DB2INCLUDE	Все	По умолчанию=текущий каталог
Задает путь, используемый при обработке оператора SQL INCLUDE текстовый-файл на стадии DB2 PREP. Содержит список каталогов, где будет идти поиск файлов INCLUDE. Смотрите в руководстве *Application Development Guide* описание использования DB2INCLUDE в различных прекомпилируемых языках.		
DB2INSTDEF	OS/2 и 32-битные операционные системы Windows	По умолчанию=DB2
Задает значение, используемое, если не определена переменная DB2INSTANCE.		
DB2INSTOWNER	Windows NT	По умолчанию=null

Таблица 21. Основные переменные реестра (продолжение)

Имя переменной Описание	Операционная система	Значения
Переменная реестра, создаваемая в реестре профиля DB2 при создании экземпляра. Эта переменная содержит имя компьютера - владельца экземпляра.		
DB2_LIC_STAT_SIZE	Все	По умолчанию=null Диапазон: от 0 до 32767
Эта переменная реестра используется для определения максимального размера (в Мбайтах) файла, содержащего статистику лицензий для данной системы. Нулевое значение выключает сбор статистики лицензий. Если переменная не распознана или не определена, по умолчанию размер не ограничивается. Статистика выводится при помощи Центра лицензий.		
DB2NBDISCOVERRCVBUFS	Все	По умолчанию=16 буферов, Минимум=16 буферов
Эта переменная используется для поиска NetBIOS. Она задает количество одновременных результатов поиска, которые можно возвратить клиенту. Если клиент получает больше одновременных результатов, чем задано в этой переменной, лишние результаты уничтожаются слоем NetBIOS. Значение по умолчанию - шестнадцать (16) буферов получения NetBIOS. Если задать меньшее значение, используется значение по умолчанию.		
DB2OPTIONS	Все, кроме Windows 3.1 и Macintosh	По умолчанию=null
Задает опции процессора командной строки.		
DB2SLOGON	Windows 3.x	По умолчанию=null, Значения: YES или NO
Включает защищенную регистрацию в системе для DB2 for Windows 3.x. Если DB2SLOGON=YES, DB2 не записывает ID пользователей и пароли в файл, а содержит их в сегменте памяти. Если переменная DB2SLOGON включена, пользователь должен регистрироваться в системе при каждом запуске Windows 3.x.		
DB2TIMEOUT	Windows 3.x и Macintosh	По умолчанию=(не задано)
Используется для управления сроком ожидания при долгих запросах SQL для клиентов Windows 3.x и клиентов Macintosh. Когда истекает срок ожидания, появляется диалоговое окно с вопросом, нужно ли прервать обработку запроса или же ждать его завершения. Минимальное значение этой переменной - 30 секунд. Если задать значение DB2TIMEOUT от 1 до 30, будет использовано минимальное значение по умолчанию. Если задать значение DB2TIMEOUT равным 0 или отрицательному числу, срок ожидания не ограничивается. По умолчанию эта возможность выключена.		
DB2TRACENAME	Windows 3.x и Macintosh	По умолчанию= DB2WIN.TRC (в Windows 3.x), DB2MAC.TRC (в Macintosh)

Таблица 21. Основные переменные реестра (продолжение)

Имя переменной	Операционная система	Значения
Описание		
В Windows 3.x и Macintosh задает имя файла, в котором хранится информация трассировки. По умолчанию для всех систем файл сохраняется в каталоге текущего экземпляра (например, `\sqllib\db2`). Мы настоятельно рекомендуем указывать для файла трассировки полный путь.		
DB2TRACEON	Windows 3.x и Macintosh	По умолчанию=NO Значения: YES или NO
В Windows 3.x и Macintosh включает трассировку, чтобы записать информацию для IBM в случае возникновения проблем. (Не рекомендуется включать трассировку, если только вы не столкнетесь с проблемой, которую не сможете разрешить.) Информацию об использовании утилиты трассировки с клиентами смотрите в руководстве *Troubleshooting Guide*.		
DB2TRCFLUSH	Windows 3.x и Macintosh	По умолчанию=NO Значения: YES или NO
В Windows 3.x и Macintosh DB2TRACEFLUSH можно использовать в сочетании с DB2TRACEON=`YES`. Если задать DB2TRACEFLUSH=`YES`, каждая запись трассировки будет немедленно записываться в файл трассировки. Это значительно замедлит работу системы DB2, поэтому по умолчанию используется значение DB2TRACEFLUSH=`NO`. Этот параметр полезен в случаях, когда какая-нибудь программа зависает и приходится перезагружать систему. Задание этого ключевого слова гарантирует, что файл трассировки и записи трассировки будут потеряны при перезагрузке.		
DB2TRCSYSERR	Windows 3.x и Macintosh	По умолчанию=1 Значения: 1-32767
Задает количество системных ошибок, после которого клиент выключает трассировку. Значение по умолчанию - трассировать одну системную ошибку, после чего трассировка выключается.		
DB2YIELD	Windows 3.x	По умолчанию=NO Значения: YES или NO
Влияет на поведение клиента Windows 3.x при связи с удаленным сервером. Если значение - `NO`, клиент не будет отдавать процессорное время прочим программам Windows 3.x, и пока программа клиента связывается с удаленным сервером, среда Windows остановлена. Чтобы продолжить остальные задания, необходимо дождаться окончания операции связи. Если значение равно `YES`, система функционирует, как обычно. Рекомендуется проверить, работает ли ваша программа с DB2YIELD=`YES`. Если происходит сбой системы, нужно задать значение DB2YIELD=`NO`. При разработке программы убедитесь, что ваша программа воспринимает и обрабатывает сообщения Windows во время ожидания завершения операции связи.		

Таблица 22. Переменные системной среды

Имя переменной	**Операционная система**	**Значения**
Описание		
DB2CONNECT_IN_APP_PROCESS	Все	По умолчанию=YES Значения: YES или NO
Установка для этой переменной значения NO запускает локальные клиенты DB2 Connect на компьютере с DB2 Connect Enterprise Edition внутри агента. Запуск внутри агента предоставляет возможность наблюдения за локальными клиентами и возможность использования ими поддержки SYSPLEX.		
DB2DOMAINLIST	Только Windows NT Server	По умолчанию=null Значения: Список имен доменов Windows NT через запятую (“,”).
Определяет один или несколько доменов Windows NT. Требования на соединение или подключение будут приниматься только от пользователей, принадлежащих этим доменам. Эту переменную среды нужно использовать только в чистой среде доменов Windows NT с серверами и клиентами DB2 DB2 Universal Database Версии 7.1 (или старше).		
DB2ENVLIST	UNIX	По умолчанию: null
Содержит имена специальных переменных для хранимых процедур или функций, определяемых пользователем. По умолчанию команда **db2start** отфильтровывает все пользовательские переменные среды, кроме тех, которые начинаются на **DB2** или **db2**. Если необходимо передавать хранимым процедурам или функциям, определенным пользователем, какие-либо переменные реестра, можно перечислить имена этих переменных в переменной реестра DB2ENVLIST. Имена переменных разделяются одним или несколькими пробелами. DB2 строит свои собственные PATH и LIBPATH, так что если в переменной DB2ENVLIST заданы PATH и LIBPATH, реальное значение переменной дописывается в конец значения, построенного DB2.		
DB2INSTANCE	Все	По умолчанию=DB2INSTDEF в OS/2 и 32-битных операционных системах Windows.
Эта переменная среды используется для задания экземпляра, активного по умолчанию. В UNIX значение переменной DB2INSTANCE должно задаваться пользователями.		
DB2INSTPROF	OS/2, Windows 3.x и 32-битные операционные системы Windows	По умолчанию: null
Эта переменная среды задает положение каталога экземпляра в OS/2, Windows 3.x и 32-битных операционных системах Windows, если оно отлично от DB2PATH.		
DB2LIBPATH	UNIX	По умолчанию: null

Таблица 22. Переменные системной среды (продолжение)

Имя переменной	**Операционная система**	**Значения**
Описание		
Задает значение LIBPATH в переменной реестра DB2LIBPATH. Значение LIBPATH не может наследоваться процессом-потомком из родительского процесса, если изменился ID пользователя. Поскольку исполняемый файл **db2start** принадлежит пользователю root, DB2 не может унаследовать значения LIBPATH конечных пользователей. Если вы включили имя переменной LIBPATH в переменную реестра DB2ENVLIST, необходимо также указать значение LIBPATH в переменной реестра DB2LIBPATH. Затем исполняемый файл **db2start** считывает значение DB2LIBPATH и дописывает его в конец переменной LIBPATH, построенной DB2.		
DB2NODE	Все	По умолчанию: null Значения: от 1 до 999
Используется для задания логического узла назначения сервера раздела базы данных DB2 Extended Enterprise Edition, к которому вы хотите подключиться или с которым вы хотите соединиться. Если эта переменная не задана, по умолчанию используется логический узел назначения, который определен с портом `0` на данном компьютере.		
DB2_PARALLEL_IO	Все	По умолчанию: null Значения: * (означает все табличные пространства) или список нескольких определенных табличных пространств через запятую
При чтении данных из контейнеров табличных пространств или записи в них данных DB2 может использовать параллельный ввод/вывод, если контейнеров в базе данных несколько. Чтобы принудительно использовать параллельный ввод/вывод для отдельного контейнера, используйте эту переменную реестра. После задания этой переменной реестра выполните команду DB2STOP, а затем DB2START, чтобы изменения вступили в силу.		
DB2PATH	OS/2, Windows 3.x и 32-битные операционные системы Windows	По умолчанию: (зависит от операционной системы)
Эта переменная среды задает, в каком каталоге установлен продукт в OS/2, Windows 3.x и 32-битных операционных системах Windows.		
DB2_STRIPED_CONTAINERS	Все	По умолчанию: null Значения: ON, null

Таблица 22. Переменные системной среды (продолжение)

Имя переменной	Операционная система	Значения
Описание		
При использовании в качестве контейнеров табличных пространств устройств RAID рекомендуется создавать табличное пространство с размером экстентов, равным или кратным размеру полосы RAID. Но из-за одностраничной метки контейнера экстенты не будут совпадать с полосами RAID, и во время требования ввода/вывода может потребоваться обращение к большему количеству дисков, чем было бы оптимально. При использовании контейнеров табличных пространств DMS этой проблемы можно избежать, выделив для метки отдельный (полный) экстент. Это устраняет проблему, но контейнер будет содержать лишний экстент служебной информации. После задания этой переменной реестра выполните команду DB2STOP, а затем DB2START, чтобы изменения вступили в силу.		

Таблица 23. Переменные связи

Имя переменной	Операционная система	Значения
Описание		
DB2CHECKCLIENTINTERVAL	AIX, только на сервере	По умолчанию=0 Значения: Число больше нуля.
Используется для проверки состояния соединений с клиентами APPC. Делает возможным раннее определение остановки клиента вместо ожидания окончания запроса. Если для переменной задано значение ноль, проверка не производится. Если задано значение больше нуля, оно определяет количество внутренних единиц работы DB2. Для руководства приведем следующие значения частоты проверки: низкая частота - 300; средняя частота - 100; высокая частота - 50. Более частая проверка состояния клиента при выполнении требования базы данных увеличивает время выполнения запросов. При большой нагрузке DB2 (то есть при выполнении большого числа внутренних требований), задание маленького значения DB2CHECKCLIENTINTERVAL больше влияет на производительность, чем в ситуации, когда нагрузка маленькая и большую часть времени DB2 находится в состоянии ожидания.		
DB2COMM	Все, только на сервере	По умолчанию=null Значения: APPC, IPXSPX, NETBIOS, NPIPE или TCPIP в любом сочетании
Задает менеджеры связи, запускаемые при запуске менеджера баз данных. Если переменная не установлена, на сервере не запускается никаких менеджеров связи DB2.		
DB2_FORCE_NLS_CACHE	AIX, HP_UX, Solaris	По умолчанию=FALSE Значения: TRUE или FALSE

Таблица 23. Переменные связи (продолжение)

Имя переменной	**Операционная система**	**Значения**
Описание		
Используется для исключения возможности конфликта блокировок в многопоточных программах. Если значение этой переменной реестра - “TRUE”, информация о кодовой странице и коде страны сохраняется, когда поток запрашивает ее в первый раз. С этого момента всем остальным потокам, запрашивающим эту информацию, возвращается информация из кэша. Это исключает конфликт блокировок и в определенных ситуациях увеличивает производительность. Этот параметр нельзя использовать, если между соединениями программа изменяет параметры локализации. Но, скорее всего, он все равно не требуется в этом случае, так как многопоточные программы обычно не изменяют параметров локализации, поскольку это не “потокозащищенное” действие.		
DB2NBADAPTERS	OS/2 и Windows NT	По умолчанию=0 Диапазон: 0-15, Несколько значений нужно разделять запятыми
Используется для задания локальных адаптеров, используемых для сетевой связи DB2 через NetBIOS. Каждый локальный адаптер задается своим логическим номером.		
DB2NBCHECKUPTIME	OS/2 и Windows NT, только на сервере	По умолчанию=1 минута Значения: 1-720
Задает интервал между вызовами процедуры проверки протокола NetBIOS. Интервал проверки задается в минутах. При меньших значениях процедура проверки протокола NetBIOS вызывается чаще и освобождает оставшуюся память и другие системные ресурсы, если происходит неожиданная остановка агента/сеанса.		
DB2NBINTRLISTENS	OS/2 и Windows NT, только на сервере	По умолчанию=1 Значения: 1-10 Несколько значений нужно разделять запятыми
Задает количество отправляемых команд ожидания NetBIOS (NCB), которые будут асинхронно выданы при готовности к прерываниям удаленного клиента. Эта возможность предназначена для сред ″активных прерываний″, чтобы запросы на прерывание от удаленных клиентов могли установить соединение, если серверы заняты обслуживанием остальных удаленных прерываний. Задание маленького значения DB2NBINTRLISTENS будет сохранять сеансы NetBIOS и NCB на сервере. Но в среде с частыми прерываниями клиентов может потребоваться задать большее значение DB2NBINTRLISTENS, чтобы быстро реагировать на прерывание клиентов. **Примечание:** Значения задаются по позициям; они связаны с позициями соответствующих значений DB2NBADAPTERS.		

Таблица 23. Переменные связи (продолжение)

Имя переменной	**Операционная система**	**Значения**
Описание		
DB2NBRECVBUFFSIZE	OS/2 и Windows NT, только на сервере	По умолчанию=4096 байт Диапазон: 4096-65536
Задает размер буферов приема протокола NetBIOS DB2. Эти буфера назначаются NCB приема NetBIOS. Меньшие значения экономят память сервера, но для больших передач данных могут потребоваться большие значения.		
DB2NBBRECVNCBS	OS/2 и Windows NT, только на сервере	По умолчанию=10 Диапазон: 1-99
Задает число команд NetBIOS ″receive_any″ (NCB), которые сервер будет выдавать и поддерживать при работе. Это значение можно настроить в зависимости от количества удаленных клиентов, с которыми соединен сервер. Меньшие значения экономят ресурсы сервера. **Примечание:** Для каждого используемого адаптера можно задать в переменной DB2NBBRECVNCBS собственное уникальное значение. Значения задаются по позициям; они связаны с позициями соответствующих значений DB2NBADAPTERS.		
DB2NBRESOURCES	OS/2 и Windows NT, только на сервере	По умолчанию=null
Задает число ресурсов NetBIOS, выделяемых для использования DB2 в многоконтекстной среде. Эта переменная используется только при работе с многоконтекстными клиентами.		
DB2NBSENDNCBS	OS/2 и Windows NT, только на сервере	По умолчанию=6 Диапазон: 1-720
Число команд отправки NetBIOS (NCB), зарезервированных сервером для использования. Это значение можно настроить в зависимости от количества удаленных клиентов, с которыми соединен сервер. Задание меньшего значения DB2NBSENDNCBS экономит ресурсы сервера. Но вам может потребоваться задать большее значение, чтобы сервер не ожидал отправки на удаленный клиент, когда все остальные команды отправки используются.		
DB2NBSESSIONS	OS/2 и Windows NT, только на сервере	По умолчанию=null Диапазон: 5-254
Задает количество сеансов, которое DB2 требует зарезервировать для использования DB2. Можно задать значение DB2NBSESSIONS, которое будет требовать отдельный сеанс для каждого адаптера, указанного в DB2NBADAPTERS. **Примечание:** Значения задаются по позициям; они связаны с позициями соответствующих значений DB2NBADAPTERS.		
DB2NBXTRANCBS	OS/2 и Windows NT, только на сервере	По умолчанию=по 5 на каждый адаптер Диапазон: 5-254

Таблица 23. Переменные связи (продолжение)

Имя переменной	Операционная система	Значения
Описание		
Задает число команд NetBIOS ″extra″ (NCB), которые сервер должен зарезервировать при вызове команды **db2start**. Можно задать значение DB2NBXTRANCBS, которое будет требовать отдельный сеанс для каждого адаптера, указанного в DB2NBADAPTERS.		
DB2NETREQ	Windows 3.x	По умолчанию=3 Диапазон: 0-25
Задает число требований NetBIOS, которые можно одновременно выполнять на клиентах Windows 3.x. Чем больше заданное значение, тем больше будет использовано памяти до уровня 1 Мбайт. Когда число одновременных требований использования служб NetBIOS достигает заданного, последующие входящие требования служб NetBIOS задерживаются в очереди и активируются, когда завершаются текущие требования. Если задать для DB2NETREQ значение 0, клиент DB2 Windows вызывает NetBIOS в синхронном режиме при помощи опции ожидания NetBIOS. В этом режиме клиент баз данных клиент допускает активность только текущего требования NetBIOS и не обрабатывает последующих требований, пока не завершится текущее требование. Это может повлиять на другие программы. Значение 0 предусмотрено только для совместимости с ранними версиями. Настоятельно рекомендуется не использовать 0.		
DB2RETRY	OS/2 и Windows NT	По умолчанию=0 Диапазон: 0-20000
Количество раз, которое DB2 пытается перезапустить принимающую программу APPC. Если подсистема SNA на сервере/шлюзе не работает, эту переменную профиля можно использовать совместно с DB2RETRYTIME для автоматического перезапуска принимающей программы APPC без разрыва связи с клиентами, использующей другие протоколы. При таком сценарии для восстановления связи с клиентом APPC уже не обязательно останавливать и снова запускать DB2.		
DB2RETRYTIME	OS/2 и Windows NT	По умолчанию=1 минута Диапазон: 0-7200 минут
Количество минут, допускаемое DB2 между последовательными попытками перезапуска принимающей программы APPC. Если подсистема SNA на сервере/шлюзе не работает, эту переменную профиля можно использовать совместно с DB2RETRY для автоматического перезапуска принимающей программы APPC без разрыва связи с клиентами, использующей другие протоколы. При таком сценарии для восстановления связи с клиентом APPC уже не обязательно останавливать и снова запускать DB2.		
DB2SERVICETPINSTANCE	OS/2, Windows NT, AIX и Sun Solaris	По умолчанию=null

Таблица 23. Переменные связи (продолжение)

Имя переменной	**Операционная система**	**Значения**
Описание		
Используется для устранения ошибок, вызванных следующими причинами: • Несколько экземпляров работают на одном компьютере • Экземпляр Версии 6 или Версии 7, работающий на том же компьютере, пытается зарегистрировать TP с теми же именами. При вызове команды **db2start** указанный экземпляр запустит принимающие программы APPC для следующих имен TP: • DB2DRDA • x'07'6DB		
DB2SOSNDBUF	Windows 95 и Windows NT	По умолчанию=32767
Задает величину буферов отправки TCP/IP в операционных системах Windows 95 и Windows NT.		
DB2SYSPLEX_SERVER	OS/2, Windows NT и UNIX	По умолчанию=null
Задает, разрешается ли использование SYSPLEX при соединении с DB2 for OS/390. Если эта переменная не задана (по умолчанию так и есть) или ее значение отлично от нуля, использование SYSPLEX разрешено. Если значение этой переменной реестра равно нулю (0), использование SYSPLEX запрещено. При значении 0 использование SYSPLEX для данного шлюза запрещено независимо от того, что указано в записи каталога баз данных DCS. Дополнительную информацию смотрите в книге *Command Reference*, команда **CATALOG DCS DATABASE**.		
DB2TCPCONNMGRS	Все	По умолчанию=1 на однопроцессорных компьютерах; квадратный корень из числа процессоров, округленный в большую сторону, но не более восьми менеджеров соединений на симметрических многопроцессорных компьютерах. Значения: от 1 до 8

Таблица 23. Переменные связи (продолжение)

Имя переменной	Операционная система	Значения
Описание		
Если эта переменная реестра не задана, создается число менеджеров соединений по умолчанию. Если переменная задана, указанное значение имеет приоритет над значением по умолчанию. Создается указанное количество (максимум 8) менеджеров соединений TCP/IP. Если задано число меньше одного, используется DB2TCPCONNMGRS=1, а в журнал записывается предупреждение о том, что значение выходит за рамки допустимого диапазона. Если задано число больше восьми, используется DB2TCPCONNMGRS=8, а в журнал записывается предупреждение о том, что значение выходит за рамки допустимого диапазона. Значения от одного до восьми используются неизмененными. Если создается несколько менеджеров соединений, пропускная способность соединений должна увеличиться, если одновременно получено несколько соединений с клиентами. Если пользователь работает на компьютере SMP или изменил переменную DB2TCPCONNMGRS, могут быть запущены дополнительные процессы (в UNIX) или потоки (в OS/2 и операционных системах Windows) менеджеров соединений TCP/IP. Дополнительным процессам или потокам требуется дополнительная память. **Примечание:** Если менеджер соединений всего один, понижается производительность удаленных соединений в системах, в которых много пользователей, часто устанавливаются и завершаются соединения или же и то, и другое.		
DB2_VI_ENABLE	Windows NT	По умолчанию=OFF Значения: ON или OFF
Указывает, нужно ли использовать протокол связи архитектуры VI (Virtual Interface Architecture). Если значение этой переменной реестра - “ON”, FCM будет использовать для связи между узлами VI. Если значение этой переменной реестра - “OFF”, FCM будет использовать для связи между узлами TCP/IP. **Примечание:** Значение этой переменной реестра должно совпадать для всех разделов баз данных текущего экземпляра.		
DB2_VI_VIPL	Windows NT	По умолчанию= `vipl.dll`
Задает имя библиотеки Virtual Interface Provider Library (VIPL), используемой DB2. Чтобы библиотека успешно загружалась, заданное в этой переменной имя библиотеки должно быть включено в пользовательскую переменную среды PATH. Все поддерживаемые на данный момент реализации используют одно и то же имя библиотеки.		
DB2_VI_DEVICE	Windows NT	По умолчанию=null Значения: `nic0` или `VINIC`
Задает символическое имя устройства или Virtual Interface Provider Instance, связанного с платой сетевого интерфейса Network Interface Card (NIC). Все независимые производители оборудования (Independent hardware vendors, IHV) производят собственные NIC. На одном компьютере Windows NT можно установить только одну NIC; несколько логических узлов на одном физическом компьютере будут совместно использовать одну и ту же NIC. Символическое имя устройства “VINIC” должно задаваться в верхнем регистре; его можно использовать только для соединений Synfinity. Для всех других поддерживаемых в настоящее время реализаций используйте в качестве символического имени устройства имя “nic0”.		

Таблица 24. Переменные каталога DCE

Имя переменной Описание	Операционная система	Значения
DB2DIRPATHNAME	OS/2, UNIX и 32-битные операционные системы Windows	По умолчанию=null
Задает временную замену значения параметра DIR_PATH_NAME в файле конфигурации менеджера баз данных. Если используется сервер каталогов и назначение оператора CONNECT или команды ATTACH не внесено в каталог в явном виде, оно приписывается к значению переменной DB2DIRPATHNAME (если она задана) для образования полного имени DCE. **Примечание:** Переменная DB2DIRPATHNAME не влияет на глобальное имя экземпляра, которое всегда определяется параметрами конфигурации менеджера баз данных DIR_PATH_NAME и DIR_OBJ_NAME.		
DB2CLIENTADPT	OS/2 и 32-битные операционные системы Windows	По умолчанию=null Диапазон: 0-15
Задает номер адаптера клиента для протокола NETBIOS в OS/2 и 32-битных операционных системах Windows. Значение DB2CLIENTADPT имеет приоритет над значением параметра DFT_CLIENT_ADPT в файле конфигурации менеджера баз данных.		
DB2CLIENTCOMM	OS/2, UNIX и 32-битные операционные системы Windows	По умолчанию=null
Задает временную замену значения параметра DFT_CLIENT_COMM в файле конфигурации менеджера баз данных. Если не заданы ни DFT_CLIENT_COMM, ни DB2CLIENTCOMM, используется первый же протокол, найденный на объекте. Если задано значение одного или обоих этих параметров, будет использован только первый совпадающий протокол. В любом случае, если первое соединение не удается, не делается никаких повторных попыток.		
DB2ROUTE	OS/2, UNIX и 32-битные операционные системы Windows	По умолчанию=null
Задает имя объекта информации маршрутизации, используемое клиентом при соединении с базой данных с другим протоколом базы данных. Значение DB2ROUTE имеет приоритет над значением параметра ROUTE_OBJ_NAME в файле конфигурации менеджера баз данных.		

Таблица 25. Переменные командной строки

Имя переменной	Операционная система	Значения
Описание		
DB2BQTIME	Все	По умолчанию=1 секунда Максимальное значение: 1 секунда
Задает промежуток времени, которое интерфейсный процесс процессора командной строки ждет перед проверкой, активен ли внутренний процесс, и установлением связи с ним.		
DB2BQTRY	Все	По умолчанию=60 попыток Минимальное значение: 0 попыток
Задает число попыток интерфейсного процесса процессора командной строки определить, активен ли внутренний процесс. Переменная действует в сочетании с DB2BQTIME.		
DB2IQTIME	Все	По умолчанию=5 секунд Минимальное значение: 1 секунда
Задает промежуток времени, в течение которого внутренний процесс процессора командной строки ждет во входной очереди передачи команд для интерфейсного процесса.		
DB2RQTIME	Все	По умолчанию=5 секунд Минимальное значение: 1 секунда
Задает промежуток времени, в течение которого внутренний процесс процессора командной строки ждет требования от интерфейсного процесса.		

Таблица 26. Переменные конфигурации MPP

Имя переменной	Операционная система	Значения
DB2ATLD_PORTS	DB2 UDB EEE в AIX, Solaris и Windows NT	По умолчанию= 6000:6063 Значение: число1:число2, где оба числа находятся между 1 и 65535 и число1<=число2
Задает диапазон номеров портов, используемый для внутренней связи TCPIP утилиты AutoLoader. Если переменная не задана, AutoLoader использует внутренний диапазон портов по умолчанию 6000:6063. Если диапазон портов AutoLoader по умолчанию используется другими программами, при помощи этой переменной можно задать альтернативный диапазон.		
DB2ATLD_PWFILE	DB2 UDB EEE в AIX, Solaris и Windows NT	По умолчанию=null Значение: выражение, означающее путь файла

Таблица 26. Переменные конфигурации MPP (продолжение)

Имя переменной	Операционная система	Значения
Задает путь файла, содержащего пароль, используемый при аутентификации AutoLoader. Если переменная не задана, AutoLoader либо использует пароль из своего файла конфигурации, либо интерактивно запрашивает его у вас. Использование этой переменной предназначено для увеличения безопасности паролей и позволяет отделить конфигурационную информацию AutoLoader от информации, связанной с аутентификацией.		
DB2CHGPWD_EEE	DB2 UDB EEE в AIX и Windows NT	По умолчанию=null Значения: YES или NO
Задает, разрешаете ли вы другим пользователям менять пароли в системах EEE на основе AIX или Windows NT. Необходимо, чтобы пароли для всех разделов или узлов управлялись централизованно, при помощи контроллера доменов Windows NT в Windows NT или NIS в AIX. Если пароли не будут управляться централизованно, они могут не совпасть на разных разделах или узлах. В связи с этим пароль может измениться только в разделе базы данных, с которым для этого изменения соединился пользователь. Чтобы изменить эту глобальную переменную реестра, необходимо находиться в корневом каталоге и в экземпляре DAS.		
DB2_FORCE_FCM_BP	AIX	По умолчанию=No Значения: Yes или No
Эта переменная реестра применяется в DB2 UDB EEE for AIX при использовании нескольких логических разделов. При вызове DB2START DB2 выделяет буферы FCM из глобальной памяти базы данных или, если там не хватает места, из отдельного сегмента совместно используемой памяти, который используется всеми демонами FCM (для этого экземпляра) на одном и том же физическом компьютере. Выбор DB2 в большой степени зависит от количества буферов FCM, которое нужно создать (оно, в свою очередь, определяется параметром конфигурации менеджера баз данных FCM_NUM_BUFFERS). Если значение этой переменной реестра - Yes, буферы FCM всегда создаются в отдельном сегменте памяти. Когда буферы FCM создаются в отдельном сегменте памяти, связь между демонами FCM в различных логических разделах одного физического узла происходит через совместно используемую память. В противном случае демоны FCM на одном узле связываются через UNIX Sockets. Достоинством связи через совместно используемую память является более высокая скорость. Недостаток заключается в том, что остается на один сегмент меньше совместно используемой памяти для прочих нужд, главное - для пулов буферов баз данных. Это уменьшает максимальный размер пулов буферов баз данных.		
DB2_NUM_FAILOVER_NODES	Все	По умолчанию: 2 Значения: от 0 до количества логических узлов

Таблица 26. Переменные конфигурации MPP (продолжение)

Имя переменной	Операционная система	Значения
Задает количество узлов, которые можно использовать в качестве узлов восстановления при отказах в среде высокой доступности. При высокой доступности, если узел не работает, его можно перезапустить как второй логический узел на другом хосте. Значение этой переменной определяет, сколько памяти резервируется для ресурсов FCM в расчете на узлы восстановления. Например, на хосте A есть два логических узла: 1 и 2; на хосте B тоже есть два логических узла: 3 и 4. Допустим, что значение DB2_NUM_FAILOVER_NODES равно 2. При запуске DB2START и хост A, и хост B зарезервируют для FCM количество памяти, достаточное для управления четырьмя логическими узлами. Тогда, если один из хостов отказывает, логические узлы отказавшего хоста можно перезапустить на другом хосте.		
DB2PORTRANGE	Windows NT	Значения: nnnn:nnnn
Это значение задает диапазон портов TCP/IP, используемый FCM, чтобы все дополнительные разделы, созданные на другом компьютере, использовали тот же диапазон портов.		
DB2_UPDATE_PART_KEY	Все	По умолчанию=Yes Значения: Yes или No
Для FixPak 3 или старше значение по умолчанию - ″Yes″. Эта переменная реестра задает, разрешено или нет изменение ключа разделения.		

Таблица 27. Переменные компилятора SQL

Имя переменной Описание	Операционная система	Значения
DB2_ANTIJOIN	Все	По умолчанию=NO в среде EEE По умолчанию=YES в среде не EEE Значения: YES или NO
Для сред DB2 Universal Database EEE: Если задать ″Yes″, оптимизатор будет искать возможности преобразования подзапросов “NOT EXISTS” в антиобъединения, которые DB2 может обрабатывать более эффективно. Для других сред (не EEE): Если задать ″No″, оптимизатор ограничит возможности преобразования подзапросов “NOT EXISTS” в антиобъединения.		
DB2_CORRELATED_PREDICATES	Все	По умолчанию=Yes Значения: Yes или No

Таблица 27. Переменные компилятора SQL (продолжение)

Имя переменной	**Операционная система**	**Значения**
Описание		
Значение по умолчанию для этой переменной - ″Yes″. Если по соответствующим столбцам в объединении существуют индексы уникальности, и для этой переменной реестра задано значение ″Yes″, оптимизатор пытается выявить и компенсировать корреляцию предикатов объединения. Если для этой переменной реестра задано значение ″Yes″, оптимизатор использует информацию KEYCARD статистики индексов уникальности, чтобы выявить случаи корреляции, и динамически настраивает совместную избирательность коррелирующих предикатов, таким образом получая более точную оценку размера и стоимости объединения. Выполняется также настройка для корреляции простых предикатов равенства, таких как `WHERE C1=5 AND C2=10` (где существует индекс по C1 и C2). Индекс может не быть индексом уникальности, но все столбцы индекса должны входить в предикат равенства.		
DB2_HASH_JOIN	Все	По умолчанию=NO Значения: YES или NO
Задает в качестве возможного метода объединения при компиляции плана доступа хеш-объединение.		
DB2_LIKE_VARCHAR	Все	По умолчанию: Y,N Значения: Y, N, S или константа с плавающей запятой от 0 до 6,2

Таблица 27. Переменные компилятора SQL (продолжение)

Имя переменной	Операционная система	Значения
Описание		

Управляет собранием и использованием статистики подэлементов. Это статистика о содержимом данных в столбцах, когда структура данных имеет вид последовательностей подполей и подэлементов, разделенных пробелами.
 	
От этой переменной реестра зависит, как оптимизатор поступает с предикатом данной формы:
`COLUMN LIKE '%xxxxxx%'`
 	
где `xxxxxx` - любая цепочка символов.
 	
Следующий синтаксис показывает, как используется эта переменная реестра:
`db2set DB2_LIKE_VARCHAR=[Y|N|S|num1] [,Y|N|S|num2]`
 	
где
• Элемент перед запятой или единственный элемент справа от предиката означает следующее (только для столбцов без позитивной статистики подэлементов):
– S – Оптимизатор оценивает длину каждого элемента в сериях элементов, сочлененных в форме столбца на основе длины строки, заключенной в символы %.
– Y – Значение по умолчанию. Использовать для параметра алгоритма значение по умолчанию 1,9. Использовать с параметром алгоритма алгоритм подэлементов с переменной длиной.
– N – Использовать алгоритм подэлементов с фиксированной длиной.
– num1 – Использовать num1 в качестве параметра алгоритма с алгоритмом подэлементов с переменной длиной.
• Элемент после запятой означает следующее:
– N – Значение по умолчанию. Не собирать или не использовать статистики подэлементов.
– Y – Собирать статистику подэлементов. Использовать алгоритм подэлементов с переменной длиной, использующий собранную статистику совместно со значением по умолчанию 1,9 для параметра алгоритма в случае столбцов с позитивной статистикой подэлементов.
– num2 – Собирать статистику подэлементов. Использовать алгоритм подэлементов с переменной длиной, использующий собранную статистику совместно со значением num2 в качестве параметра алгоритма для столбцов с позитивной статистикой подэлементов.		

DB2_SELECTIVITY	Все	По умолчанию=No Значения: Yes или No
Эта переменная реестра управляет тем, где может использоваться условие SELECTIVITY. Подробности об условии SELECTIVITY смотрите в разделе Language Elements, Search Conditions справочника *SQL Reference*. Если эта переменная реестра имеет значение ″Yes″, условие SELECTIVITY можно указывать для базового предиката, по крайней мере одно из выражений которого содержит переменные хоста.		

Таблица 27. Переменные компилятора SQL (продолжение)

Имя переменной	Операционная система	Значения
Описание		
DB2_NEW_CORR_SQ_FF	Все	По умолчанию=OFF Значения: ON или OFF

Когда значение равно “ON”, влияет на значение избирательности, вычисляемое оптимизатором SQL для некоторых предикатов подзапросов. Эту переменную можно использовать для повышения точности значения избирательности предикатов подзапросов равенства, использующих в списке подзапроса SELECT функцию сводки MIN или MAX. Например:
`SELECT * FROM T WHERE`
`T.COL = (SELECT MIN(T.COL)`
`FROM T WHERE ...)`		

DB2_PRED_FACTORIZE	Все	По умолчанию=NO Значение: YES или NO

Указывает, должен ли оптимизатор искать возможности выделения дополнительных предикатов из дизъюнкций. В некоторых ситуациях дополнительные предикаты могут изменить оценку числа результатов в промежуточном и конечном наборе. Для следующего запроса:
`SELECT n1.empno,`
`n1.lastname`
`FROM employee n1,`
`employee n2`
`WHERE`
`((n1.lastname='SMITH'`
`AND n2.lastname='JONES')`
`OR (n1.lastname='JONES'`
`AND n2.lastname='SMITH'))`
 	
оптимизатор может сгенерировать дополнительные предикаты:
`SELECT n1.empno,`
`n1.lastname`
`FROM employee n1,`
`employee n2`
`WHERE n1.lastname IN`
`('SMITH', 'JONES')`
`AND n2.lastname IN`
`('SMITH', 'JONES')`
`AND`
`((n1.lastname='SMITH'`
`AND n2.lastname='JONES')`
`OR (n1.lastname='JONES'`
`AND n2.lastname='SMITH'))`		

Таблица 28. Переменные производительности

Имя переменной	Операционная система	Значения
Описание		
DB2_AVOID_PREFETCH	Все	По умолчанию=OFF, Значения: ON или OFF
Указывает, нужно ли при восстановлении после сбоя использовать предварительную выборку. Если DB2_AVOID_PREFETCH=ON, предварительная выборка не используется.		
DB2_AWE	Windows 2000	По умолчанию=null Значения: <запись>[,<запись>,...], где <запись>=<ID пула буферов>, <число физических страниц>, <число окон адресации>
Позволяет DB2 UDB в Windows 2000 выделять пулы буферов, где используется до 64 Гбайт памяти. Для поддержки пулов буферов Address Windowing Extensions (AWE) нужно правильно сконфигурировать Windows 2000. Для этого нужно предоставить пользователя право “lock pages in memory” (блокировать страницы в памяти), выделить физические страницы и страницы окон адресации и задать эту переменную реестра. При задании этой переменной нужно знать ID пула буферов, который будет использоваться для поддержки AWE. Этот ID можно посмотреть в столбце BUFFERPOOLID производной таблицы системного каталога SYSCAT.BUFFERPOOLS. **Примечание:** Если поддержка AWE включена, расширенная память не может использоваться ни для каких пулов буферов в базе данных. Кроме того, указанные в этой переменной реестра пулы буферов должны уже существовать в SYSCAT.SYSBUFFERPOOLS.		
DB2_BINSORT	Все	По умолчанию=YES Значения: YES или NO
Включает новый алгоритм сортировки, уменьшающий использование процессорного времени и общее время сортировок. Этот алгоритм распространяет крайне эффективную технику сортировки целых чисел DB2 UDB на все сортируемые типы данных, такие, как BIGINT, CHAR, VARCHAR, FLOAT и DECIMAL, а также на сочетания этих типов. Чтобы включить этот новый алгоритм, используйте команду: `db2set DB2_BINSORT = yes`		
DB2BPVARS	Windows NT	По умолчанию=path

Таблица 28. Переменные производительности (продолжение)

Имя переменной	Операционная система	Значения
Описание		

Задает путь файла, содержащего параметры, используемые при настройке пулов буферов. На данный момент поддерживаются следующие параметры: NT_SCATTER_DMSFILE, NT_SCATTER_DMSDEVICE и NT_SCATTER_SMS.
 	
Для каждого из этих параметров значение по умолчанию равно нулю (или OFF); возможны значения 0 (или OFF) и 1 (или ON). Каждый параметр используется для включения распределенного чтения для соответствующего типа контейнеров. Каждый может быть включен (значение ON), только если переменная реестра DB2NTNOCACHE имеет значение ON. Если значение DB2NTNOCACHE - OFF (или не установлено), в db2diag.log записывается предупреждение, а распределенное чтение остается выключенным. Эти параметры рекомендуются для систем с большим объемом последовательной предварительной выборки из соответствующего типа контейнеров, для которых вы уже решили использовать значение DB2NTNOCACHE, равное ON.
Примечание: Значение DB2NTNOCACHE ON отключает кэширование файлов в Windows NT.
 	
Пример задания пути к этому файлу:
`db2set DB2BPVARS =`
`  f:\BPVARSFILE`
 	
В файле могут содержаться любые из этих параметров в виде:
`параметр=значение`		

DB2CHKPTR	Все	По умолчанию=OFF, Значения: ON или OFF
Указывает, требуется ли проверка указателей во входном тексте.		
DB2_ENABLE_BUFPD	Все	По умолчанию=OFF, Значения: ON или OFF
Задает, использует ли DB2 промежуточную буферизацию для повышения производительности запросов. Не в любой среде такая буферизация может улучшать производительность запросов. Для выявления отдельных улучшений производительности запросов необходимо тестирование.		
DB2_EXTENDED_OPTIMIZATION	Все	По умолчанию=OFF Значения: ON или OFF
Указывает, должен ли оптимизатор запросов использовать оптимизирующие расширения для улучшения производительности запросов. Эти расширения могут улучшать производительность запросов не в любой среде. Для выявления отдельных улучшений производительности запросов необходимо тестирование.		
DB2MAXFSCRSEARCH	Все	По умолчанию=5 Значения: -1, от 1 до 33554

Таблица 28. Переменные производительности (продолжение)

Имя переменной	Операционная система	Значения
Описание		
Задает число записей управления свободным пространством, которые ищутся при добавлении записи в таблицу. Значение по умолчанию - искать пять записей управления свободным пространством. Изменяя это значение, можно улучшить скорость операций вставки за счет повторного использования пространства или наоборот. Используйте большие значения, чтобы улучшить повторное использование пространства. Используйте небольшие значения, чтобы улучшить скорость операций вставки. Если задано значение -1, менеджер баз данных будет искать все записи управления свободным пространством.		
DB2MEMDISCLAIM	AIX	По умолчанию=YES Значения: YES или NO
В AIX память, используемая процессами DB2, может иметь некоторое связанное пространство подкачки. Это пространство подкачки может оставаться зарезервированным, даже если была освобождена связанная с ним память. Это зависит от правил управления выделением настраиваемой виртуальной памяти системы AIX. Переменная реестра DB2MEMDISCLAIM задает, будут ли агенты DB2 явно требовать, чтобы AIX отсоединяла зарезервированное пространство подкачки от освобожденной памяти. Значение DB2MEMDISCLAIM YES предполагает меньшие требования к пространству подкачки и, возможно, меньшую активность диска, связанную с подкачкой. Значение DB2MEMDISCLAIM NO предполагает большие требования к пространству подкачки и, возможно, большую активность диска, связанную с подкачкой. В некоторых ситуациях, например, при избыточном пространстве подкачки и настолько большом объеме реальной памяти, что подкачки никогда не происходит, значение NO дает незначительное улучшение производительности.		
DB2MEMMAXFREE	Все	По умолчанию=8388608 байт Значения: от 0 до 2^{32}-1 байт
Задает максимальный объем неиспользуемой памяти (в байтах), удерживаемой процессами DB2.		
DB2_MMAP_READ	AIX	По умолчанию=ON , Значения: ON или OFF
Работает совместно с DB2_MMAP_WRITE и позволяет DB2 использовать в качестве альтернативного метода ввода/вывода mmap. В большинстве сред, чтобы избежать блокировок операционной системы, когда несколько процессов пишут в разные части одного и того же файла, следует использовать mmap. Но, возможно, ранее вы работали в Parallel Edition V1.2, где значение по умолчанию было OFF, что позволяло AIX кэшировать данные DB2, считанные из файловых систем JFS в память (вне пула буферов). Если вам требуется производительность, сравнимая с DB2 UDB, вы можете либо увеличить размер пула буферов, либо задать для DB2_MMAP_READ и DB2_MMAP_WRITE значения `OFF`.		
DB2_MMAP_WRITE	AIX	По умолчанию=ON Значения: ON или OFF

Таблица 28. Переменные производительности (продолжение)

Имя переменной	Операционная система	Значения
Описание		
Работает совместно с DB2_MMAP_READ и позволяет DB2 использовать в качестве альтернативного метода ввода/вывода mmap. В большинстве сред, чтобы избежать блокировок операционной системы, когда несколько процессов пишут в разные части одного и того же файла, следует использовать mmap. Но, возможно, ранее вы работали в Parallel Edition V1.2, где значение по умолчанию было OFF, что позволяло AIX кэшировать данные DB2, считанные из файловых систем JFS в память (вне пула буферов). Если вам требуется производительность, сравнимая с DB2 UDB, вы можете либо увеличить размер пула буферов, либо задать для DB2_MMAP_READ и DB2_MMAP_WRITE значения `OFF`.		
DB2_NO_PKG_LOCK	Все	По умолчанию=OFF Значения: ON или OFF
Позволяет Глобальному кэшу SQL работать без использования блокировок пакетов, чтобы защищать кэшированные записи пакетов. (Блокировки пакетов - это внутренние системные блокировки.) Чтобы улучшить производительность (получение и освобождение блокировок отнимает время), теперь можно выбрать работу в режиме “без блокировок пакетов”. В этом режиме запрещены некоторые операции баз данных. В эти операции могут входить: операции, делающие пакеты недействительными, операции, делающие пакеты нерабочими, и операции, напрямую изменяющие пакет.		
DB2NTMEMSIZE	Windows NT	По умолчанию=(разное для разных сегментов памяти)
Windows NT требует, чтобы все сегменты совместно используемой памяти резервировались во время инициализации DLL, чтобы гарантировать совпадение адресов для разных процессов. DB2NTMEMSIZE предназначена, чтобы позволить пользователю в случае необходимости изменить параметры DB2 в Windows NT. В большинстве случаев значений по умолчанию должно быть достаточно. Размеры сегментов памяти по умолчанию и изменяющие их опции: 1) Ядро баз данных: размер по умолчанию - 16777216 (16 Мбайт); опция замены - DBMS:<число байт>. 2) Буферы параллельного FCM: размер по умолчанию - 22020096 (21 Мбайт); опция замены - FCM:<число байт>. 3) Графический интерфейс администратора баз данных: размер по умолчанию - 33554432 (32 Мбайт); опция замены - DBAT:<число байт>. 4) Изолированные хранимые процедуры: размер по умолчанию - 16777216 (16 Мбайт); опция замены - APLD:<число байт>. Можно изменить размер нескольких сегментов, указав опции замены через точку с запятой (;). Например, чтобы ограничить ядро баз данных приблизительно 256 Кбайтами, а буферы FCM приблизительно 64 Мбайтами, используйте команду: `db2set DB2NTMEMSIZE=DBMS:256000;FCM:64000000`		
DB2NTNOCACHE	Windows NT	По умолчанию=OFF Значение: ON или OFF
Указывает, должна ли DB2 открывать файлы баз данных с опцией NOCACHE. Если DB2NTNOCACHE=ON, кэширование файловой системы исключается. Если DB2NTNOCACHE=OFF, операционная система кэширует файлы DB2. Это относится ко всем данным, кроме файлов, содержащих длинные поля и большие объекты. Устранение системного кэширования освобождает больше памяти для базы данных, что позволяет увеличить пул буферов или кучу сортировки.		

Таблица 28. Переменные производительности (продолжение)

Имя переменной	Операционная система	Значения
Описание		
DB2NTPRICLASS	Windows NT	По умолчанию=null Значение: R, H, (все прочие значения)
Задает класс приоритета для экземпляра DB2 (программа DB2SYSCS.EXE). Есть три класса приоритета: • NORMAL_PRIORITY_CLASS (класс приоритета по умолчанию) • REALTIME_PRIORITY_CLASS (задается как “R”) • HIGH_PRIORITY_CLASS (задается как “H”) Эта переменная используется в сочетании с приоритетами отдельных потоков (устанавливаемыми посредством DB2PRIORITIES) для определения абсолютного приоритета потоков DB2 по отношению к другим потокам в системе. **Примечание:** Эту переменную следует использовать с осторожностью. Неправильное использование может неблагоприятно повлиять на общую производительность системы. Дополнительную информацию смотрите в описании функции *SetPriorityClass()* в документации по Win32 API.		
DB2NTWORKSET	Windows NT	По умолчанию=1,1
Используется для изменения доступных DB2 минимального и максимального размеров рабочего набора. По умолчанию, если в Windows NT не происходит подкачка страниц, рабочий набор процесса может расти до любого требуемого размера. Но если происходит подкачка страниц, максимальный рабочий набор для процесса - приблизительно 1 Мбайт. DB2NTWORKSET позволяет изменить эту установку по умолчанию. Значение переменной DB2NTWORKSET устанавливается в DB2 следующим образом: DB2NTWORKSET=`min,max`, где min и max задаются в мегабайтах.		
DB2_OVERRIDE_BPF	Все	По умолчанию=не задано Значения: положительное число страниц
Задает размер пула буферов (в страницах), который создается при активации базы данных или при первом соединении. Эта переменная полезна, если при активации базы данных или при первом соединении происходят ошибки в связи с ограничениями памяти. Если база данных не выдержит даже минимального пула буферов в 16 страниц, пользователь может попробовать еще раз, указав меньшее количество страниц в этой переменной среды. Ошибка памяти может быть связана с реальным недостатком памяти (что случается редко) или с попыткой менеджера баз данных выделить большие, неправильно сконфигурированные пулы буферов. Если это значение задано, оно заменит текущий размер пула буферов.		
DB2_PINNED_BP	AIX, HP-UX	По умолчанию=NO Значения: YES или NO

Таблица 28. Переменные производительности (продолжение)

Имя переменной	Операционная система	Значения
Описание		

Эта переменная используется для хранения глобальной памяти базы данных (включая пулы буферов), связанной с базой данных, в главной памяти на нескольких операционных системах AIX. Хранение этой глобальной памяти базы данных в главной памяти системы позволяет улучшить производительность базы данных.
 	
Если бы, например, пул буферов сбрасывался из главной памяти системы на диск, производительность базы данных бы снизилась. Сокращение дискового ввода-вывода, когда пулы буферов размещаются в памяти системы, улучшает производительность базы данных. Если есть другие программы, которым требуется больший объем главной памяти, можно разрешить программам использование глобальной памяти базы данных с подкачкой из главной памяти в зависимости от требований к главной памяти системы.
 	
При работе с HP-UX в 64-битной среде,кроме изменения этой переменной реестра, нужно также предоставить привилегию MLOCK группе экземпляра DB2. Для этого пользователь с правами доступа ROOT должен выполнить следующее:
1. Добавить группу экземпляра DB2 в файл `/etc/privgroup`. Например, если группа экземпляра DB2 принадлежит группе `db2iadm1`, в файл `/etc/privgroup` надо добавить следующую строку:
`db2iadm1 MLOCK`
2. Выполнить следующую команду:
`setprivgrp -f /etc/privgroup`		

DB2PRIORITIES Управляет приоритетами процессов и потоков DB2.	Все	Значения зависят от платформы.
DB2_RR_TO_RS	Все	По умолчанию=NO Значения: YES или NO

Таблица 28. Переменные производительности (продолжение)

Имя переменной	Операционная система	Значения
Описание		
Блокировка следующего ключа гарантирует уровень изоляции Многократное чтение (RR), автоматически блокируя следующий ключ для всех операторов INSERT и DELETE и следующее большее значение ключа для набора результатов для операторов SELECT. Для операторов UPDATE, которые изменяют части ключей в индексе, исходное значение ключа удаляется, а новое значение - вставляется. Блокировка следующего значения при этом производится и для вставки ключа, и для удаления ключа. Блокировка следующего ключа требуется для обеспечения стандартного многократного чтения ANSI и SQL92 и используется в DB2 по умолчанию. Если ваша программа останавливается или зависает, надо проверить информацию снимка для нее. Если есть подозрение, что проблема связана с блокировкой следующего ключа, можно включить переменную реестра DB2_RR_TO_RS, принимая во внимание следующие условия. Переменную DB2_RR_TO_RS можно включить, если никакие программы не полагаются на эффекты многократного чтения(RR) и если при просмотре приемлем пропуск непринятых удалений. Такой пропуск повлияет на уровни изоляции Многократное чтение (RR), Стабильность чтения (RS) и Стабильность на уровне указателя (CS). (Для уровня изоляции Чтение непринятого (UR) блокировка строк не используется.) Если переменная DB2_RR_TO_RS включена, поведение RR при просмотре не может быть гарантировано, поскольку при вставке и удалении ключей индекса блокировка следующего ключа не выполняется. На таблицы каталогов эта опция не влияет. Другой эффект включения переменной DB2_RR_TO_RS - пропуск при просмотре удаленных, но непринятых строк, даже если эти строки могут быть отобраны для просмотра.		
DB2_SORT_AFTER_TQ	Все	По умолчанию=NO Значения: YES или NO
Указывает, как оптимизатор должен работать с направленными очередями таблиц в многораздельной базе данных, если получатель требует, чтобы данные были отсортированы, а число получающих узлов равно числу отправляющих узлов. Когда DB2_SORT_AFTER_TQ=NO, оптимизатор старается производить сортировку в месте отправки и слияние строк в месте приема. Когда DB2_SORT_AFTER_TQ=YES, оптимизатор старается передавать строки несортированными, не сливать их в месте приема и сортировать их на месте приема только после получения всех строк.		
DB2_STPROC_LOOKUP_FIRST	Все	По умолчанию=OFF Значения: ON или OFF

Таблица 28. Переменные производительности (продолжение)

Имя переменной	Операционная система	Значения
Описание		
Эта переменная ранее называлась DB2_DARI_LOOKUP_ALL; она указывает, должен ли сервер UDB производить просмотр в каталоге для всех DARI и хранимых процедур до поиска в подкаталоге *function* подкаталога *sqllib*; и в подкаталоге *unfenced* подкаталога *function* подкаталога *sqllib*. **Примечание:** Задание для этой переменной значения “ON” ухудшит производительность для хранимых процедур PARAMETER TYPE DB2DARI, находящихся в вышеуказанных каталогах, так как, возможно, в конфигурации EEE просмотр каталога будет выполняться на другом узле до поиска в каталогах function. При вызове хранимой процедуры DB2 по умолчанию сначала ищет совместно используемую библиотеку с тем же именем, что и эта хранимая процедура, в подкаталоге *function* подкаталога *sqllib* и в подкаталоге *unfenced* подкаталога *function* подкаталога *sqllib* и только потом ищет имя совместно используемой библиотеки для хранимых процедур в системном каталоге. Только хранимые процедуры с PARAMETER TYPE DB2DARI могут иметь те же имена, что и их совместно используемые библиотеки, поэтому поведение DB2 по умолчанию выгодно только для хранимых процедур DB2DARI. Если используются хранимые процедуры, внесенные в каталог с другим PARAMETER TYPE, затрачиваемое DB2 время на поиск в указанных выше каталогах ухудшает производительность этих хранимых процедур. Чтобы улучшить производительность хранимых процедур, внесенных в каталог с другим PARAMETER TYPE (не DB2DARI), задайте для переменной реестра DB2_STPROC_LOOKUP_FIRST значение ON. При таком значении этой переменной реестра DB2 будет искать имя совместно используемой библиотеки для хранимых процедур в системном каталоге перед поиском в указанных выше каталогах.		

Таблица 29. Переменные связей данных

Имя переменной	Операционная система	Значения
Описание		
DLFM_BACKUP_DIR_NAME	AIX, Windows NT	По умолчанию: null Значения: TSM или любой существующий путь
Задает используемое устройство резервного копирования. Если изменить значение этой переменной реестра с TSM на какой-нибудь путь во время выполнения, архивные файлы не перемещаются. В новое место помещаются только новые резервные копии. Ранее архивированные файлы не перемещаются.		
DLFM_BACKUP_LOCAL_MP	AIX, Windows NT	По умолчанию: null Значения: любой правильный путь локальной точки монтирования в системе DFS
Задает полный путь к точке монтирования в системе DFS. Если задан путь, он используется вместо пути, заданного в DLFM_BACKUP_DIR_NAME.		

Таблица 29. Переменные связей данных (продолжение)

Имя переменной **Описание**	**Операционная система**	**Значения**
DLFM_BACKUP_TARGET Задает тип используемой резервной копии.	AIX, Windows NT	По умолчанию: null Значения: LOCAL, TSM, XBSA
DLFM_BACKUP_TARGET_LIBRARY Задает полный путь к DLL или совместно используемой библиотеке. Эта библиотека загружается посредством библиотеки *libdfmxbsa.a*.	AIX, Windows NT	По умолчанию: null Значения: любой правильный путь к DLL или совместно используемой библиотеке
DLFM_ENABLE_STPROC Указывает, используется ли для связывания групп файлов хранимая процедура.	AIX, Windows NT	По умолчанию: NO Значения: YES или NO
DLFM_FS_ENVIRONMENT Задает среду, в которой работают серверы связей данных. NATIVE означает, что сервер связей данных находится однокомпьютерной среде и может получить контроль над файлами на собственном компьютере. DFS означает, что сервер связей данных находится в среде распределенной файловой системы (DFS) и может получить контроль над файлами во всей файловой системе. Смешивать наборы файлов DFS с обычными файловыми системами не разрешается.	AIX, Windows NT	По умолчанию: NATIVE Значения: NATIVE или DFS
DLFM_GC_MODE Управляет сбором файлов мусора на сервере связей данных. Если задать SLEEP, сбор мусора не будет происходить. Если задать PASSIVE, сбор мусора запускается, только когда больше не выполняется никаких транзакций. Если задать ACTIVE, сбор мусора происходит, даже если выполняются другие транзакции.	AIX, Windows NT	По умолчанию: PASSIVE Значения: SLEEP, PASSIVE или ACTIVE
DLFM_INSTALL_PATH Задает путь, где установлены исполняемые файлы связей данных.	AIX, Windows NT	По умолчанию В AIX: /usr/lpp/ db2_06_00 /adm В NT: DB2PATH /bin Диапазон: любой правильный путь

Таблица 29. Переменные связей данных (продолжение)

Имя переменной Описание	Операционная система	Значения
DLFM_LOG_LEVEL	AIX, Windows NT	По умолчанию: LOG_INFO Значения: LOG_CRIT, LOG_DEBUG, LOG_ERR, LOG_INFO, LOG_NOTICE, LOG_WARNING
Задает уровень записи диагностической информации.		
DLFM_PORT	Все, кроме Windows 3.n	По умолчанию: 50100 Значения: любой правильный номер порта
Задает номер порта, используемый при связи с серверами связей данных, на которых работает менеджер связей данных DB2. Эта переменная реестра используется, только когда таблица содержит столбец “DATALINKS”.		
DLFM_TSM_MGMTCLASS	AIX, Windows NT, Solaris	По умолчанию: класс управления TSM по умолчанию Значения: любой допустимый класс управления TSM
Задает класс управления TSM, который должен использоваться для архивирования и получения связанных файлов. Если значение этой переменной не задано, используется класс управления TSM по умолчанию.		

Таблица 30. Разные переменные

Имя переменной Описание	Операционная система	Значения
DB2ADMINSERVER	OS/2, Windows 95, Windows NT и UNIX	По умолчанию=null
Указывает, какой экземпляр DB2 сконфигурирован в качестве сервера администратора DB2.		
DB2CLIINIPATH	Все	По умолчанию=null
Переопределяет путь по умолчанию файла конфигурации DB2 CLI/ODBC (db2cli.ini) и задает иное положение клиента. Заданное значение должно представлять правильный путь на системе клиента.		
DB2DEFPREP	Все	По умолчанию=NO Значения: ALL, YES или NO
Воспроизводит во время выполнения действие опции прекомпиляции DEFERRED_PREPARE для программ, прекомпилированных до появления этой опции. Например, если программа для DB2 v2.1.1 или более ранней версии выполняется в среде DB2 v2.1.2 или более поздней версии, для включения ’отложенной подготовки’ можно использовать DB2DEFPREP.		

Таблица 30. Разные переменные (продолжение)

Имя переменной **Описание**	**Операционная система**	**Значения**
DB2_DJ_COMM	Все	По умолчанию=null Возможные значения: libdrda.a, libsqlnet.a, libnet8.a, libdrda.dll, libsqlnet.dll, libnet8.dll и так далее.
Задает библиотеки оболочки, загружаемые при запуске менеджера баз данных. Задание этой переменной уменьшает расходы на загрузку часто используемых оболочек во время выполнения. Для других операционных систем поддерживаются другие значения (расширение .dll используется для операционной системы Windows NT; расширение .a используется для операционной системы AIX). Имена библиотек различаются в зависимости от протокола и от операционной системы. Эта переменная доступна, только если значение параметра менеджера баз данных *federated* - YES.		
DB2DMNBCKCTLR	Windows NT	По умолчанию=null Значения: ? или имя домена
Если вы знаете имя домена, для которого этот сервер DB2 является резервным контроллером домена, задайте DB2DMNBCKCTLR=ИМЯ_ДОМЕНА. ИМЯ_ДОМЕНА необходимо вводить в верхнем регистре. Чтобы DB2 определила, для какого домена локальный компьютер является резервным контроллером домена, задайте DB2DMNBCKCTLR=?. Если переменная профиля DB2DMNBCKCTLR не задана или пуста, DB2 выполняет аутентификацию на первичном контроллере домена. **Примечание:** DB2 по умолчанию не использует имеющийся резервный контроллер домена, так как резервный контроллер домена может выйти из синхронизации с первичным контроллером домена, и появится проблема с защитой. Выход из синхронизации может произойти, когда база данных безопасности первичного контроллера домена изменяется, но изменения не отправляются на резервный контроллер домена. Это может произойти из-за задержек сети или если не работает служба просмотра компьютеров.		
DB2_ENABLE_LDAP	Все	По умолчанию=NO Значения: YES или NO
Указывает, нужно ли использовать протокол LDAP (Lightweight Directory Access Protocol - протокол упрощенного доступа к каталогам). LDAP - это метод доступа к службам каталогов.		
DB2_FALLBACK	Windows NT	По умолчанию=OFF Значения: ON или OFF
Эта переменная позволяет принудительно разорвать все соединения с базами данных при обработке возврата к прежней конфигурации. Она используется вместе с поддержкой восстановления при отказах в среде Windows NT с Microsoft Cluster Server (MSCS). Если DB2_FALLBACK не задана или имеет значение OFF, и если при возврате к прежней конфигурации имеется соединение с базой данных, ресурсы DB2 выключить невозможно. Это означает, что возврат к к прежней конфигурации не удастся.		
DB2_FORCE_TRUNCATION	Все	По умолчанию=NO Значения: YES или NO

Таблица 30. Разные переменные (продолжение)

Имя переменной	**Операционная система**	**Значения**
Описание		
Используется при восстановлении после перезапуска. Если задать значение “NO”, восстановление после перезапуска будет отменено в случае, если оно будет чересчур рано (то есть до того, как прочтены все активные журналы) останавливаться из-за плохой страницы. Обычно это вызвано поврежденной страницей в одном из журналов. Пользователь может задать для этой переменной значение “YES”, чтобы сообщить восстановлению после перезапуска, что оно должно продолжать работу, как если бы оно дошло до конца журналов. Если эта переменная имеет значение “YES”, журналы, не считанные во время восстановления, будут затерты при активизации базы данных. По умолчанию используется “NO”, и при обнаружении плохой страницы работа **не** продолжается. Используйте эту переменную только по указанию обслуживающего персонала IBM.		
DB2_GRP_LOOKUP	Windows NT	По умолчанию=null Значения: LOCAL, DOMAIN
Эта переменная сообщает DB2, где нужно выполнять регистрацию учетных записей пользователей и поиск членов групп. Если задать значение LOCAL, DB2 всегда будет просматривать группы и регистрировать учетные записи пользователей на сервере DB2. Если задать значение DOMAIN, DB2 всегда будет просматривать группы и регистрировать учетные записи пользователей на сервере Windows NT, к которому принадлежит данная учетная запись пользователя.		
DB2_INDEX_2BYTEVARLEN	Все	По умолчанию=NO Значения: YES или NO
Эта переменная реестра позволяет задавать в качестве части ключа индекса столбцы длиной более 255 байт. У индексов, созданных до задания для этой переменной реестра значения YES, для ключей по-прежнему останется предельное ограничение 255. Индексы, созданные после задания для данной переменной реестра значения ″Yes″, будут вести себя как двухбайтные индексы, если для этой переменной реестра вновь задать значение ″No″. Изменения для этой переменной реестра влияют на несколько операторов SQL, в том числе CREATE TABLE, CREATE INDEX и ALTER TABLE. Дополнительную информацию об этих операторах смотрите в изменениях для справочника *SQL Reference*.		
DB2LDAP_BASEDN	Все	По умолчанию=null Значения: Любое правильное базовое имя домена.
Задает базовое имя домена для каталога LDAP.		
DB2LDAPCACHE	Все	По умолчанию=YES Значения: YES или NO

Таблица 30. Разные переменные (продолжение)

Имя переменной	Операционная система	Значения
Описание		

Указывает, что нужно включить кэш LDAP. Этот кэш используется для внесения в каталог на локальном компьютере каталоги баз данных, узлов и DCS.
 	
Чтобы кэш содержал наиболее свежую информацию, выполните следующие команды:
`REFRESH LDAP DB DIR`
`REFRESH LDAP NODE DIR`
 	
Эти команды обновляют каталог баз данных и каталог узлов и удаляют из них неправильные записи.		

DB2LDAP_CLIENT_PROVIDER	Только Windows 95/98/NT/2000	По умолчанию=null (в случае доступности используется Microsoft; иначе используется IBM.) Значения: IBM или Microsoft
При работе в среде Windows DB2 поддерживает для доступа к каталогу LDAP либо клиенты LDAP Microsoft, либо клиенты LDAP IBM. Эта переменная реестра используется для выбора, какой клиент LDAP должна использовать DB2, в явном виде. **Примечание:** Чтобы вывести текущее значение этой переменной реестра, используйте команду `db2set`: `db2set DB2LDAP_CLIENT_PROVIDER`		
DB2LDAPHOST	Все	По умолчанию=null Значения: Любое правильное имя хоста.
Задает имя хоста, на котором находится каталог LDAP.		
DB2LDAP_SEARCH_SCOPE	Все	По умолчанию= DOMAIN Значения: LOCAL, DOMAIN, GLOBAL
Задает область поиска для информации, находящейся в разделах или доменах в LDAP. “LOCAL” выключает поиск в каталоге LDAP. “DOMAIN” ищет в LDAP только для текущего раздела каталога. “GLOBAL” ищет в LDAP во всех разделах каталогов, пока не найдет требуемый объект.		
DB2LOADREC	Все	По умолчанию=null
Используется для замены положения загрузочной копии при повторе транзакций. Если пользователь изменил физическое положение загрузочной копии, до повтора транзакций необходимо изменить DB2LOADREC.		
DB2LOCK_TO_RB	Все	По умолчанию=null Значения: Statement
Задает, будет ли истечение срока блокировки вызывать откат всей транзакции или же только текущего оператора. Если DB2LOCK_TO_RB имеет значение `STATEMENT`, истечение срока вызовет откат только текущего оператора. Любое другое значение приведен к откату всей транзакции.		

Таблица 30. Разные переменные (продолжение)

Имя переменной	Операционная система	Значения
Описание		
DB2_NEWLOGPATH2	UNIX	По умолчанию=0 Значения: 0 или 1
Этот параметр позволяет задать, будет ли использоваться второй путь для реализации двойной регистрации в журналах. Используемый путь получается добавлением “2” к текущему значению параметра конфигурации *logpath*.		
DB2NOEXITLIST	Все	По умолчанию=OFF Значения: ON или OFF
Если эта переменная задана, она указывает DB2 не устанавливать в программах обработчик списка выходов и не выполнять COMMIT. Обычно DB2 устанавливает в программах обработчик списка выхода процессов, и в случае нормального завершения программы он выполняет операцию COMMIT. Для программ, динамически загружающих библиотеку DB2 и выгружающих ее до окончания программы, вызов обработчика списка выхода не удается, так как в программе больше не загружена процедура обработчика. Если ваша программа работает таким образом, нужно задать переменную DB2NOEXITLIST и убедиться, что ваша программа вызывает все необходимые COMMIT в явном виде.		
DB2REMOTEPREG	Windows 95 и Windows NT	По умолчанию=null Значение: Любое правильное имя компьютера Windows 95 или Windows NT
Задает имя удаленного компьютера, содержащего список реестра Win32 профилей экземпляров DB2 и экземпляров DB2. Значение DB2REMOTEPREG нужно задать только один раз после установки DB2, и изменять его нельзя. Используйте эту переменную с особой осторожностью.		
DB2ROUTINE_DEBUG	AIX и Windows NT	По умолчанию=OFF Значения: ON, OFF
Указывает, нужно ли включать возможность отладки хранимых процедур Java. Если вы не занимаетесь отладкой хранимых процедур Java, оставьте значение по умолчанию - OFF. Включение отладки влияет на производительность. Дополнительную информацию об отладке хранимых процедур Java смотрите в книге *Application Development Guide*.		
DB2SORCVBUF	Windows 95 и Windows NT	По умолчанию=32767
Задает величину буферов приема TCP/IP в операционных системах Windows 95 и Windows NT.		
DB2SORT	Все, только на сервере	По умолчанию=null
Задает положение библиотеки, загружаемой при выполнении утилитой LOAD. Эта библиотека содержит точку входа для функций, используемых при сортировке и индексировании данных. Переменная DB2SORT предназначена для использования внешних программных продуктов сортировки с утилитой LOAD при генерировании индексов таблиц. Этот путь должен задаваться относительно сервера баз данных.		

Таблица 30. Разные переменные (продолжение)

Имя переменной **Описание**	**Операционная система**	**Значения**
DB2SYSTEM	Windows NT, Windows 95, OS/2 и UNIX	По умолчанию=null
Задает имя, которое будет использоваться пользователями и администраторами баз данных для идентификации системы сервера DB2. По возможности это имя должно быть уникальным в вашей сети. Оно выводится на уровне систем в дереве объектов Центра управления и помогает администраторам идентифицировать системы серверов, которыми можно управлять из Центра управления. При использовании функции Ассистента конфигурирования клиента 'Поиск в сети' поиск DB2 возвращает это имя, и оно же выводится на уровне системы в результирующем дереве объектов. Это имя помогает пользователям идентифицировать систему, содержащую базу данных, доступ к которой им нужно получить. Значение DB2SYSTEM задается во время установки так: • В Windows NT или Windows 95 программа установки задает его равным имени компьютера, указанному для системы Windows • В OS/2 пользователю предлагается ввести имя DB2SYSTEM во время установки. • В системах UNIX оно совпадает с именем хоста TCP/IP системы UNIX.		
DB2UPMPR	OS/2	По умолчанию=ON Значения: ON или OFF
Задает, будет ли выводиться экран регистрации UPM, если пользователь ввел в OS/2 неверный ID пользователя или пароль.		
DB2_VENDOR_INI	AIX, HP-UX, Sun Solaris и Windows NT	По умолчанию=null Значения: Любой правильный путь и имя файла
Задает файл, содержащий все установки среды, специфичные для конкретного производителя. Значение читается при запуске менеджера баз данных.		
DB2_XBSA_LIBRARY	AIX, HP-UX, Sun Solaris и Windows NT	По умолчанию=null Значения: Любой правильный путь и имя файла

Таблица 30. Разные переменные (продолжение)

Имя переменной	Операционная система	Значения
Описание		

Указывает на предоставляемую производителем библиотеку XBSA. В AIX эта установка должна включать в себя совместно используемый объект, если он не носит имя `shr.o`. В HP-UX, Sun Solaris и Windows NT имени совместно используемого объекта не требуется. Например, чтобы использовать NetWorker Business Suite Module для DB2 фирмы Legato, надо задать значение для этой переменной реестра так:
`db2set DB2_XSBA_LIBRARY="/usr/lib/libxdb2.a(bsashr10.o)"`
Интерфейс XBSA можно вызвать из команд BACKUP DATABASE или RESTORE DATABASE. Например:
`db2 backup db sample use XBSA`
`db2 restore db sample use XBSA`		

Приложение B. Таблицы объяснения и их определения

Когда активна возможность объяснения, в таблицы объяснения записывается информация о планах доступа. В этом разделе описаны следующие таблицы объяснения и их определения:

Перед запуском объяснения должны быть созданы таблицы объяснения. Для их создания используйте пример входного сценария процессора командной строки из файла EXPLAIN.DDL, расположенного в подкаталоге 'misc' каталога 'sqllib'. Соединитесь с базой данных, в которой надо создать таблицы объяснения. Затем введите команду `db2 -tf EXPLAIN.DDL`, и эти таблицы будут созданы. Дополнительную информацию смотрите в разделе “Определения таблиц для таблиц объяснения” на стр. 580.

Когда возможность объяснения заполняет таблицы объяснения, не активируются никакие триггеры или реляционные или проверочные ограничения. Например, если для таблицы EXPLAIN_INSTANCE определен триггер вставки и сохраняется информация объяснения о соответствующем операторе, этот триггер не будет активирован.

Более подробную информацию о возможности объяснения смотрите в разделе “Глава 7. Возможность объяснения SQL” на стр. 225.

Пояснение к таблицам объяснения:

Заголовок	**Объяснение**
Имя столбца	Имя столбца
Тип данных	Тип данных столбца
Допусти-мость пустых значений	Да: Пустые значения разрешены Нет: Пустые значения не разрешены

Ключ	PK: Столбец является частью первичного ключа FK: Столбец является частью внешнего ключа
Описание	Описание столбца

Таблица EXPLAIN_ARGUMENT

Таблица EXPLAIN_ARGUMENT содержит индивидуальные характеристики каждого отдельного оператора, если они есть.

Таблица 31. Таблица EXPLAIN_ARGUMENT

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
EXPLAIN_REQUESTER	VARCHAR (128)	Нет	FK	ID авторизации пользователя, потребовавшего объяснение.
EXPLAIN_TIME	TIMESTAMP	Нет	FK	Время запуска требования объяснения.
SOURCE_NAME	VARCHAR (128)	Нет	FK	Имя выполняемого пакета при объяснении динамического оператора или имя исходного файла при объяснении статического оператора SQL.
SOURCE_SCHEMA	VARCHAR (128)	Нет	FK	Схема или спецификатор источника требования объяснения.
EXPLAIN_LEVEL	CHAR(1)	Нет	FK	Уровень информации объяснения, к которой относится эта строка.
STMTNO	INTEGER	Нет	FK	Номер оператора в пакете, к которому относится эта информация объяснения.
SECTNO	INTEGER	Нет	FK	Номер раздела в пакете, к которому относится эта информация объяснения.
OPERATOR_ID	INTEGER	Нет	Нет	Уникальный ID для этой операции в запросе.
ARGUMENT_TYPE	CHAR(8)	Нет	Нет	Тип аргумента для этой операции.
ARGUMENT_VALUE	VARCHAR (1024)	Да	Нет	Значение аргумента для этой операции. NULL, если значение записано в LONG_ARGUMENT_VALUE.
LONG_ARGUMENT_VALUE	CLOB(1M)	Да	Нет	Значение аргумента для этой операции, если оно не помещается в ARGUMENT_VALUE. NULL, если значение записано в ARGUMENT_VALUE.

Таблица 32. Значения столбцов ARGUMENT_TYPE и ARGUMENT_VALUE

Значение ARGUMENT_TYPE	Возможные значения ARGUMENT_VALUE	Описание
AGGMODE	COMPLETE PARTIAL INTERMEDIATE FINAL	Индикаторы частичного суммирования.

Таблица 32. Значения столбцов ARGUMENT_TYPE и ARGUMENT_VALUE (продолжение)

Значение ARGUMENT_TYPE	Возможные значения ARGUMENT_VALUE	Описание
BITFLTR	TRUE FALSE	Хеш-объединение будет использовать битовый фильтр для улучшения производительности.
CSETEMP	TRUE FALSE	Флаг временной таблицы для общего подвыражения.
DIRECT	TRUE	Индикатор прямой выборки.
DUPLWARN	TRUE FALSE	Флаг предупреждения о повторениях.
EARLYOUT	TRUE FALSE	Индикатор раннего выхода.
ENVVAR	Каждая строка такого типа содержит: • Имя переменной среды • Значение переменной среды	Переменная среды, влияющая на оптимизатор
FETCHMAX	IGNORE INTEGER	Переопределяет значение аргумента MAXPAGES операции FETCH.
GROUPBYC	TRUE FALSE	Есть ли столбцы группировки.
GROUPBYN	Целое	Число столбцов сравнения.
GROUPBYR	Каждая строка такого типа содержит: • Порядковый номер столбца в условии Group by (за которым идет двоеточие и пробел) • Имя столбца	Требование группировки.
INNERCOL	Каждая строка такого типа содержит: • Порядковый номер столбца (за которым идет двоеточие и пробел) • Имя столбца • Значение упорядочивания **(A)** По возрастанию **(D)** По убыванию	Столбцы внутреннего упорядочивания.
ISCANMAX	IGNORE INTEGER	Переопределяет значение аргумента MAXPAGES операции ISCAN.
JN_INPUT	INNER OUTER	Указывает, подается ли на операцию внутреннее или внешнее объединение.
LISTENER	TRUE FALSE	Индикатор очереди таблиц программы приема.
MAXPAGES	ALL NONE INTEGER	Максимальное число страниц, ожидаемых для предварительной выборки.
MAXRIDS	NONE INTEGER	Идентификаторы максимальной строки для включения в каждое требование предварительной выборки списка.

Таблица 32. Значения столбцов ARGUMENT_TYPE и ARGUMENT_VALUE (продолжение)

Значение ARGUMENT_TYPE	Возможные значения ARGUMENT_VALUE	Описание
NUMROWS	INTEGER	Ожидаемое число сортируемых строк.
ONEFETCH	TRUE FALSE	Индикатор одной выборки.
OUTERCOL	Каждая строка такого типа содержит: • Порядковый номер столбца (за которым идет двоеточие и пробел) • Имя столбца • Значение упорядочивания **(A)** По возрастанию **(D)** По убыванию	Столбцы внешнего порядка.
OUTERJN	LEFT RIGHT	Индикатор внешнего объединения.
PARTCOLS	Имя столбца	Столбцы разделения для операции.
PREFETCH	LIST NONE SEQUENTIAL	Тип приемлемой предварительной выборки.
RMTQTEXT	Текст запроса	Текст удаленного запроса
ROWLOCK	EXCLUSIVE NONE REUSE SHARE SHORT (INSTANT) SHARE UPDATE	Намерение блокировки строки.
ROWWIDTH	INTEGER	Длина сортируемой строки.
SCANDIR	FORWARD REVERSE	Направление просмотра.
SCANGRAN	INTEGER	Внутриразделный параллелизм, уровень просмотра для внутриразделного параллелизма, выраженный в SCANUNIT.
SCANTYPE	LOCAL PARALLEL	Внутриразделный параллелизм, просмотр индекса или таблицы.
SCANUNIT	ROW PAGE	Внутриразделный параллелизм, единица уровня просмотра.
SERVER	Удаленный сервер	Удаленный сервер
SHARED	TRUE	Внутриразделный параллелизм, совместный индикатор TEMP.
SLOWMAT	TRUE FALSE	Флаг медленной материализации.
SNGLPROD	TRUE FALSE	Внутриразделный параллелизм, сортировка или временная, порожденная одним агентом.

Таблица 32. Значения столбцов ARGUMENT_TYPE и ARGUMENT_VALUE (продолжение)

Значение ARGUMENT_TYPE	Возможные значения ARGUMENT_VALUE	Описание
SORTKEY	Каждая строка такого типа содержит: • Порядковый номер столбца в ключе (за которым идет двоеточие и пробел) • Имя столбца • Значение упорядочивания **(A)** По возрастанию **(D)** По убыванию	Столбцы ключа сортировки.
SORTTYPE	PARTITIONED SHARED ROUND ROBIN REPLICATED	Внутриразделный параллелизм, тип сортировки.
TABLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE REUSE SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	Намерение блокировки таблицы.
TQDEGREE	INTEGER	внутриразделный параллелизм, число подагентов, обращающихся к очереди таблиц.
TQMERGE	TRUE FALSE	Индикатор объединенной (сортированной) очереди таблиц.
TQREAD	READ AHEAD STEPPING SUBQUERY STEPPING	Свойство чтения очереди таблиц.
TQSEND	BROADCAST DIRECTED SCATTER SUBQUERY DIRECTED	Свойство отправки очереди таблиц.
TQTYPE	LOCAL	Внутриразделный параллелизм, очередь таблиц.
TRUNCSRT	TRUE	Усеченная сортировка (число порожденных строк ограничено).
UNIQUE	TRUE FALSE	Индикатор уникальности.
UNIQKEY	Каждая строка такого типа содержит: • Порядковый номер столбца в ключе (за которым идет двоеточие и пробел) • Имя столбца	Столбцы ключа уникальности.
VOLATILE	TRUE	Таблица сильно переменного объема

Таблица EXPLAIN_INSTANCE

Таблица EXPLAIN_INSTANCE - это главная управляющая таблица для всей информации объяснения. Каждая строка данных в таблицах объяснения связана с одной из уникальных строк в этой таблице. Таблица EXPLAIN_INSTANCE содержит основную информацию об исходных объясняемых операторах SQL, а также информацию о среде, для которой дано объяснение.

Определение этой таблицы смотрите в разделе “Определение таблицы EXPLAIN_INSTANCE” на стр. 582.

Таблица 33. Таблица EXPLAIN_INSTANCE

Имя столбца	Тип данных	Допусти-мость пустых значений	Ключ	Описание
EXPLAIN_REQUESTER	VARCHAR (128)	Нет	PK	ID авторизации пользователя, потребовавшего объяснение.
EXPLAIN_TIME	TIMESTAMP	Нет	PK	Время запуска требования объяснения.
SOURCE_NAME	VARCHAR (128)	Нет	PK	Имя выполняемого пакета при объяснении динамического оператора или имя исходного файла при объяснении статического оператора SQL.
SOURCE_SCHEMA	VARCHAR (128)	Нет	PK	Схема или спецификатор источника требования объяснения.
EXPLAIN_OPTION	CHAR(1)	Нет	Нет	Указывает, какая информация объяснения была затребована для этого запроса. Возможные значения: **P** PLAN SELECTION
SNAPSHOT_TAKEN	CHAR(1)	Нет	Нет	Указывает, был ли сделан снимок объяснения для этого запроса. Возможные значения: **Y** Да, снимок (снимки) объяснения был сделан и сохранен в таблице EXPLAIN_STATEMENT. Обычная информация объяснения также была получена. **N** Нет, снимок объяснения не был сделан. Была получена обычная информация объяснения. **O** Был сделан только снимок объяснения. Обычная информация объяснения не была получена.

Таблица 33. Таблица EXPLAIN_INSTANCE (продолжение)

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
DB2_VERSION	CHAR(7)	Нет	Нет	Номер выпуска продукта DB2 Universal Database, обрабатывающего этот запрос объяснения. Формат: vv.rr.m, где: **vv** Номер версии **rr** Номер выпуска **m** Номер служебного выпуска
SQL_TYPE	CHAR(1)	Нет	Нет	Указывает тип оператора SQL для этого экземпляра объяснения - статический или динамический. Возможные значения: **S** Статический SQL **D** Динамический SQL
QUERYOPT	INTEGER	Нет	Нет	Указывает класс оптимизации запросов, используемый компилятором SQL во время запуска объяснения. Это значение указывает уровень оптимизации запросов, использованный компилятором SQL для объясняемых операторов SQL.
BLOCK	CHAR(1)	Нет	Нет	Указывает тип блокировки указателей, использованный при компиляции этих операторов SQL. Дополнительную информацию смотрите в столбце BLOCK в SYSCAT.PACKAGES. Возможные значения: **N** Без блокировок **U** Блокировки однозначных указателей **B** Блокировки всех указателей
ISOLATION	CHAR(2)	Нет	Нет	Указывает тип изоляции, использованный при компиляции этих операторов SQL. Дополнительную информацию смотрите в столбце ISOLATION в SYSCAT.PACKAGES. Возможные значения: **RR** Многократное чтение **RS** Стабильность чтения **CS** Стабильность на уровне указателя **UR** Чтение непринятого
BUFFPAGE	INTEGER	Нет	Нет	Содержит значение, которое имел параметр конфигурации базы данных BUFFPAGE во время запуска объяснения.
AVG_APPLS	INTEGER	Нет	Нет	Содержит значение, которое имел параметр конфигурации AVG_APPLS во время запуска объяснения.
SORTHEAP	INTEGER	Нет	Нет	Содержит значение, которое имел параметр конфигурации базы данных SORTHEAP во время запуска объяснения.

Таблица 33. Таблица EXPLAIN_INSTANCE (продолжение)

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
LOCKLIST	INTEGER	Нет	Нет	Содержит значение, которое имел параметр конфигурации базы данных LOCKLIST во время запуска объяснения.
MAXLOCKS	SMALLINT	Нет	Нет	Содержит значение, которое имел параметр конфигурации базы данных MAXLOCKS во время запуска объяснения.
LOCKS_AVAIL	INTEGER	Нет	Нет	Содержит число блокировок, доступных по предположению оптимизатора каждому пользователю. (Вычисляется на основе значений LOCKLIST и MAXLOCKS.)
CPU_SPEED	DOUBLE	Нет	Нет	Содержит значение, которое имел параметр конфигурации менеджера баз данных CPUSPEED во время запуска объяснения.
REMARKS	VARCHAR(254)	Да	Нет	Пользовательский комментарий.
DBHEAP	INTEGER	Нет	Нет	Содержит значение, которое имел параметр конфигурации базы данных DBHEAP во время запуска объяснения.
COMM_SPEED	DOUBLE	Нет	Нет	Содержит значение, которое имел параметр конфигурации базы данных COMM_BANDWIDTH во время запуска объяснения.
PARALLELISM	CHAR(2)	Нет	Нет	Возможные значения: • N = Без параллелизма • P = Внутрираздельный параллелизм • IP = Межраздельный параллелизм • BP = Внутрираздельный параллелизм и межраздельный параллелизм
DATAJOINER	CHAR(1)	Нет	Нет	Возможные значения: • N = План не относится к системам объединения • Y = План относится к системам объединения

Таблица EXPLAIN_OBJECT

Таблица EXPLAIN_OBJECT задает, какие объекты данных требуются плану доступа, сгенерированному для данного оператора SQL.

Таблица 34. Таблица EXPLAIN_OBJECT

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
EXPLAIN_REQUESTER	VARCHAR (128)	Нет	FK	ID авторизации пользователя, потребовавшего объяснение.
EXPLAIN_TIME	TIMESTAMP	Нет	FK	Время запуска требования объяснения.
SOURCE_NAME	VARCHAR (128)	Нет	FK	Имя выполняемого пакета при объяснении динамического оператора или имя исходного файла при объяснении статического оператора SQL.
SOURCE_SCHEMA	VARCHAR (128)	Нет	FK	Схема или спецификатор источника требования объяснения.
EXPLAIN_LEVEL	CHAR(1)	Нет	FK	Уровень информации объяснения, к которой относится эта строка.
STMTNO	INTEGER	Нет	FK	Номер оператора в пакете, к которому относится эта информация объяснения.
SECTNO	INTEGER	Нет	FK	Номер раздела в пакете, к которому относится эта информация объяснения.
OBJECT_SCHEMA	VARCHAR (128)	Нет	Нет	Схема, к которой принадлежит этот объект.
OBJECT_NAME	VARCHAR (128)	Нет	Нет	Имя объекта.
OBJECT_TYPE	CHAR(2)	Нет	Нет	Описательная метка типа объекта.
CREATE_TIME	TIMESTAMP	Да	Нет	Время создания объекта; для табличной функции - null.
STATISTICS_TIME	TIMESTAMP	Да	Нет	Время последнего обновления статистики для этого объекта; null, если статистика для объекта не создавалась.
COLUMN_COUNT	SMALLINT	Нет	Нет	Число столбцов в объекте.
ROW_COUNT	INTEGER	Нет	Нет	Примерное число строк в объекте.
WIDTH	INTEGER	Нет	Нет	Средняя ширина объекта в байтах. Для индекса - -1.
PAGES	INTEGER	Нет	Нет	Оценка числа страниц, занимаемых объектом в пуле буферов. Для табличной функции - -1.
DISTINCT	CHAR(1)	Нет	Нет	Указывает, различны ли все строки объекта (то есть есть ли повторения) Возможные значения: **Y** Да **N** Нет
TABLESPACE_NAME	VARCHAR (128)	Да	Нет	Имя табличного пространства, где хранится объект; null, если табличное пространство не используется.

Таблица 34. Таблица EXPLAIN_OBJECT (продолжение)

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
OVERHEAD	DOUBLE	Нет	Нет	Оценка полных расходов в миллисекундах на одну произвольную операцию ввода-вывода для указанного табличного пространства. Включает расходы на действия контроллера, поиск на диске и время задержки. -1, если табличное пространство не используется.
TRANSFER_RATE	DOUBLE	Нет	Нет	Оценка времени на чтение страницы данных в миллисекундах из заданного табличного пространства. -1, если табличное пространство не используется.
PREFETCHSIZE	INTEGER	Нет	Нет	Число страниц данных, читаемых при предварительной выборке. Для табличной функции - -1.
EXTENTSIZE	INTEGER	Нет	Нет	Размер экстента, в страницах данных. Это число страниц, записываемых в один контейнер табличного пространства перед переходом к другому контейнеру. Для табличной функции - -1.
CLUSTER	DOUBLE	Нет	Нет	Степень кластеризации данных с индексом. Если >= 1, это параметр CLUSTERRATIO. Если >= 0 и < 1, это параметр CLUSTERFACTOR. -1 для таблиц, табличных функций или если статистика не собрана.
NLEAF	INTEGER	Нет	Нет	Число конечных страниц, занимаемых значениями этого объекта (индекса). -1 для таблиц, табличных функций или если статистика не собрана.
NLEVELS	INTEGER	Нет	Нет	Число уровней индекса в дереве этого объекта (индекса). -1 для таблиц, табличных функций или если статистика не собрана.
FULLKEYCARD	BIGINT	Нет	Нет	Число различных значений полного ключа, содержащихся в этом объекте (индексе). -1 для таблиц, табличных функций или если статистика не собрана.
OVERFLOW	INTEGER	Нет	Нет	Полное число записей переполнения в таблице. -1 для таблиц, табличных функций или если статистика не собрана.
FIRSTKEYCARD	BIGINT	Нет	Нет	Число различных значений полного ключа. –1 для таблиц, табличных функций или если статистика не собрана.
FIRST2KEYCARD	BIGINT	Нет	Нет	Число различных значений ключа для первых {2,3,4} столбцов индекса. –1 для таблиц, табличных функций или если статистика не собрана.
FIRST3KEYCARD	BIGINT	Нет	Нет	
FIRST4KEYCARD	BIGINT	Нет	Нет	

Таблица 34. Таблица EXPLAIN_OBJECT (продолжение)

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
SEQUENTIAL_PAGES	INTEGER	Нет	Нет	Число конечных страниц, расположенных на диске в порядке ключей индекса, без пропусков между ними или с небольшими пропусками. –1 для таблиц, табличных функций или если статистика не собрана.
DENSITY	INTEGER	Нет	Нет	Отношение SEQUENTIAL_PAGES к числу страниц в диапазоне страниц, занимаемом индексом, выраженное в процентах (целое от 0 до 100). –1 для таблиц, табличных функций или если статистика не собрана.

Таблица 35. Возможные значения OBJECT_TYPE

Значение	Описание
IX	Индекс
TA	Таблица
TF	Табличная функция

Таблица EXPLAIN_OPERATOR

Таблица EXPLAIN_OPERATOR содержит все операции, нужные компилятору SQL для оператора SQL.

Таблица 36. Таблица EXPLAIN_OPERATOR

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
EXPLAIN_REQUESTER	VARCHAR (128)	Нет	FK	ID авторизации пользователя, потребовавшего объяснение.
EXPLAIN_TIME	TIMESTAMP	Нет	FK	Время запуска требования объяснения.
SOURCE_NAME	VARCHAR (128)	Нет	FK	Имя выполняемого пакета при объяснении динамического оператора или имя исходного файла при объяснении статического оператора SQL.
SOURCE_SCHEMA	VARCHAR (128)	Нет	FK	Схема или спецификатор источника требования объяснения.
EXPLAIN_LEVEL	CHAR(1)	Нет	FK	Уровень информации объяснения, к которой относится эта строка.
STMTNO	INTEGER	Нет	FK	Номер оператора в пакете, к которому относится эта информация объяснения.
SECTNO	INTEGER	Нет	FK	Номер раздела в пакете, к которому относится эта информация объяснения.

Таблица 36. Таблица EXPLAIN_OPERATOR (продолжение)

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
OPERATOR_ID	INTEGER	Нет	Нет	Уникальный ID для этой операции в запросе.
OPERATOR_TYPE	CHAR (6)	Нет	Нет	Описательная метка типа операции.
TOTAL_COST	DOUBLE	Нет	Нет	Оценка суммарной общей стоимости (в условных единицах времени) выполнения выбранного плана доступа вплоть до этой операции включительно.
IO_COST	DOUBLE	Нет	Нет	Оценка суммарной стоимости ввода/вывода (в страницах ввода/вывода данных) выполнения выбранного плана доступа вплоть до этой операции включительно.
CPU_COST	DOUBLE	Нет	Нет	Оценка суммарной стоимости работы процессора (в командах) для выполнения выбранного плана доступа вплоть до этой операции включительно.
FIRST_ROW_COST	DOUBLE	Нет	Нет	Оценка суммарной стоимости (в условных единицах времени) выборки первой строки для плана доступа вплоть до этой операции включительно. Это значение включает все необходимые расходы на инициализацию.
RE_TOTAL_COST	DOUBLE	Нет	Нет	Оценка суммарной стоимости (в условных единицах времени) выборки следующей строки для плана доступа вплоть до этой операции включительно.
RE_IO_COST	DOUBLE	Нет	Нет	Оценка суммарной стоимости ввода/вывода (в страницах ввода/вывода данных) выборки следующей строки для плана доступа вплоть до этой операции включительно.
RE_CPU_COST	DOUBLE	Нет	Нет	Оценка суммарной стоимости работы процессора (в условных единицах времени) для выборки следующей строки для плана доступа вплоть до этой операции включительно.
COMM_COST	DOUBLE	Нет	Нет	Оценка суммарной стоимости связи (в кадрах TCP/IP) для выполнения выбранного плана доступа вплоть до этой операции включительно.
FIRST_COMM_COST	DOUBLE	Нет	Нет	Оценка суммарной стоимости связи (в кадрах TCP/IP) для выборки первой строки для плана доступа вплоть до этой операции включительно. Это значение включает все необходимые расходы на инициализацию.
BUFFERS	DOUBLE	Нет	Нет	Оценка требований к буферу для этой операции и ее ввода.
REMOTE_TOTAL_COST	DOUBLE	Нет	Нет	Оценка суммарной общей стоимости (в условных единицах времени) выполнения операций на удаленных базах данных.

Таблица 36. Таблица EXPLAIN_OPERATOR (продолжение)

Имя столбца	Тип данных	Допусти-мость пустых значений	Ключ	Описание
REMOTE_COMM_COST	DOUBLE	Нет	Нет	Оценка суммарной стоимости связи для выполнения выбранного плана удаленного доступа вплоть до этой операции включительно.

Таблица 37. Значения OPERATOR_TYPE

Значение	Описание
DELETE	Удаление
FETCH	Выборка
FILTER	Фильтрация строк
GENROW	Генерация строки
GRPBY	Группировка
HSJOIN	Хеш-объединение
INSERT	Вставка
IXAND	Динамическая битовая операция AND над индексами
IXSCAN	Просмотр индекса
MSJOIN	Объединение просмотром со слиянием
NLJOIN	Объединение со вложенным циклом
RETURN	Результат
RIDSCN	Просмотр идентификатора строки (RID)
RQUERY	Удаленный запрос
SORT	Сортировка
TBSCAN	Просмотр таблицы
TEMP	Построение временной таблицы
TQ	Очередь таблиц
UNION	Объединение
UNIQUE	Исключение повторений
UPDATE	Изменение

Таблица EXPLAIN_PREDICATE

Таблица EXPLAIN_PREDICATE указывает, какие предикаты применяются данной операцией.

Таблица 38. Таблица EXPLAIN_PREDICATE

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
EXPLAIN_REQUESTER	VARCHAR (128)	Нет	FK	ID авторизации пользователя, потребовавшего объяснение.
EXPLAIN_TIME	TIMESTAMP	Нет	FK	Время запуска требования объяснения.
SOURCE_NAME	VARCHAR (128)	Нет	FK	Имя выполняемого пакета при объяснении динамического оператора или имя исходного файла при объяснении статического оператора SQL.
SOURCE_SCHEMA	VARCHAR (128)	Нет	FK	Схема или спецификатор источника требования объяснения.
EXPLAIN_LEVEL	CHAR(1)	Нет	FK	Уровень информации объяснения, к которой относится эта строка.
STMTNO	INTEGER	Нет	FK	Номер оператора в пакете, к которому относится эта информация объяснения.
SECTNO	INTEGER	Нет	FK	Номер раздела в пакете, к которому относится эта информация объяснения.
OPERATOR_ID	INTEGER	Нет	Нет	Уникальный ID для этой операции в запросе.
PREDICATE_ID	INTEGER	Нет	Нет	Уникальный ID для данного предиката указанной операции.
HOW_APPLIED	CHAR (5)	Нет	Нет	Как предикат используется указанной операцией.
WHEN_EVALUATED	CHAR(3)	Нет	Нет	Указывает, когда вычисляется подзапрос, используемый этим предикатом. Возможные значения: **пробел** Этот предикат не содержит подзапрос. **EAA** Подзапрос, используемый в этом предикате, вычисляется прикладной программой (EAA). Это значит, что он вычисляется заново для каждой строки, обрабатываемой указанной операцией, когда применяется предикат. **EAO** Подзапрос, используемый в этом предикате, вычисляется при открытии (EAO). Это значит, что он вычисляется только один раз для указанной операции, и его результаты используются в прикладной программе предиката для каждой строки. **MUL** В этом предикате есть несколько типов подзапросов.
RELOP_TYPE	CHAR(2)	Нет	Нет	Тип реляционного оператора, используемый в этом предикате.

Таблица 38. Таблица EXPLAIN_PREDICATE (продолжение)

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
SUBQUERY	CHAR(1)	Нет	Нет	Требуется ли данным предикатом поток данных из подзапроса. Могут потребоваться несколько потоков подзапросов. Возможные значения: **N** Потоки подзапросов не требуются. **Y** Требуются один или несколько потоков подзапросов
FILTER_FACTOR	DOUBLE	Нет	Нет	Оценка доли строк, удовлетворяющие данному предикату.
PREDICATE_TEXT	CLOB(1M)	Да	Нет	Текст предиката, воссозданный из внутреннего представления оператора SQL. Null, если текст недоступен.

Таблица 39. Возможные значения HOW_APPLIED

Значение	Описание
JOIN	Используется для объединения таблиц
RESID	Оценивается как остаточный предикат
SARG	Оценивается как предикат с аргументом поиска для страницы индекса или данных
START	Используется как начальное условие
STOP	Используется, как конечное условие

Таблица 40. Возможные значения RELOP_TYPE

Значение	Описание
пробелы	Не применимо
EQ	Равно
GE	Больше или равно
GT	Больше
IN	В списке
LE	Меньше или равно
LK	Подобно
LT	Меньше
NE	Не равно
NL	Является пустым
NN	Не является пустым

Таблица EXPLAIN_STATEMENT

Таблица EXPLAIN_STATEMENT содержит текст оператора SQL для различных уровней информации объяснения. В ней хранится как исходный текст оператора SQL, введенный пользователем, так и версия, использованная оптимизатором при выборе плана доступа для этого оператора SQL. Эта более поздняя версия может мало походить на исходный оператор, так как он может быть переписан и/или улучшен при помощи добавочных предикатов, определенных компилятором SQL.

Определение этой таблицы смотрите в разделе “Определение таблицы EXPLAIN_STATEMENT” на стр. 586.

Таблица 41. Таблица EXPLAIN_STATEMENT

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
EXPLAIN_REQUESTER	VARCHAR (128)	Нет	PK, FK	ID авторизации пользователя, потребовавшего объяснение.
EXPLAIN_TIME	TIMESTAMP	Нет	PK, FK	Время запуска требования объяснения.
SOURCE_NAME	VARCHAR (128)	Нет	PK, FK	Имя выполняемого пакета при объяснении динамического оператора или имя исходного файла при объяснении статического оператора SQL.
SOURCE_SCHEMA	VARCHAR (128)	Нет	PK, FK	Схема или спецификатор источника требования объяснения.
EXPLAIN_LEVEL	CHAR(1)	Нет	PK	Уровень информации объяснения, к которой относится эта строка. Допустимые значения: **O** Исходный текст (как он был введен пользователем) **P** PLAN SELECTION
STMTNO	INTEGER	Нет	PK	Номер оператора в пакете, к которому относится эта информация объяснения. Для динамических операторов SQL имеет значение 1. Для статических операторов SQL это значение совпадает со значением в производной таблице каталога SYSCAT.STATEMENTS.
SECTNO	INTEGER	Нет	PK	Номер раздела в пакете, содержащем этот оператор SQL. Для динамических операторов SQL этот номер раздела указывает раздел для этого оператора во время выполнения. Для статических операторов SQL это значение совпадает со значением в производной таблице каталога SYSCAT.STATEMENTS.

Таблица 41. Таблица EXPLAIN_STATEMENT (продолжение)

Имя столбца	Тип данных	Допусти-мость пустых значений	Ключ	Описание
QUERYNO	INTEGER	Нет	Нет	Числовой идентификатор для объясняемого оператора SQL. Для динамических операторов SQL (за исключением оператора SQL EXPLAIN), выданных через CLP или CLI, по умолчанию используется последовательно возрастающие значения. В остальных случаях по умолчанию используется значение STMTNO для статических операторов SQL и 1 для динамических операторов SQL.
QUERYTAG	CHAR(20)	Нет	Нет	Тэг идентификатора для каждого объясняемого оператора SQL. Для динамических операторов SQL, выданных через CLP (за исключением оператора SQL EXPLAIN), по умолчанию используется значение 'CLP'. Для динамических операторов SQL, выданных через CLI (за исключением оператора SQL EXPLAIN), по умолчанию используется значение 'CLI'. В остальных случаях значение по умолчанию - это пробелы.
STATEMENT_TYPE	CHAR(2)	Нет	Нет	Описывает тип объясняемого запроса. Возможные значения: **S** Select (выбор) **D** Delete (удаление) **DC** Delete (удаление) в текущем положении указателя **I** Insert (вставка) **U** Update (изменение) **UC** Update (изменение) в текущем положении указателя
UPDATABLE	CHAR(1)	Нет	Нет	Указывает, может ли этот оператор быть изменен. Это относится, в частности, к операторам SELECT, для которых можно определить, могут ли изменяться эти операторы. Возможные значения: **' '** Не применим (пробел) **N** Нет **Y** Да
DELETABLE	CHAR(1)	Нет	Нет	Указывает, может ли этот оператор быть удален. Это относится, в частности, к операторам SELECT, для которых можно определить, могут ли удаляться эти операторы. Возможные значения: **' '** Не применим (пробел) **N** Нет **Y** Да

Таблица 41. Таблица EXPLAIN_STATEMENT (продолжение)

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
TOTAL_COST	DOUBLE	Нет	Нет	Оценка общих затрат (в условных единицах времени) на выполнение выбранного плана доступа для этого оператора; имеет значение 0 (ноль), если EXPLAIN_LEVEL равен *O* (исходный оператор), так как в это время еще не выбран план доступа.
STATEMENT_TEXT	CLOB(1M)	Нет	Нет	Текст или часть текста объясняемого оператора SQL. Для объяснения уровня выбора плана выводится текст, восстановленный из внутреннего представления оператора; его синтаксис подобен синтаксису SQL, но не обязательно точно ему следует.
SNAPSHOT	BLOB(10M)	Да	Нет	Снимок внутреннего представления этого оператора SQL на указанном уровне объяснения (EXPLAIN_LEVEL). Этот столбец предназначен для использования с инструментом наглядного объяснения (Visual Explain) DB2. Этому столбцу присваивается пустое значение, если EXPLAIN_LEVEL равен *0* (исходный оператор), так как в то время, когда сохраняется эта конкретная версия оператора, еще не выбран план доступа.
QUERY_DEGREE	INTEGER	Нет	Нет	Указывает степень внутриразделного параллелизма во время вызова объяснения. Для уровня исходного оператора это назначенная степень внутриразделного параллелизма. Для уровня выбора плана это степень внутриразделного параллелизма, которую будет использовать этот план.

Таблица EXPLAIN_STREAM

Таблица EXPLAIN_STREAM представляет входные и выходные потоки данных между отдельными операциями и объектами данных. Сами объекты данных описаны в таблице EXPLAIN_OBJECT. Операции, работающие с потоком данных, описаны в таблице EXPLAIN_OPERATOR.

Таблица 42. Таблица EXPLAIN_STREAM

Название столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
EXPLAIN_REQUESTER	VARCHAR (128)	Нет	FK	ID авторизации пользователя, потребовавшего объяснение.
EXPLAIN_TIME	TIMESTAMP	Нет	FK	Время запуска требования объяснения.

Таблица 42. Таблица EXPLAIN_STREAM (продолжение)

Название столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
SOURCE_NAME	VARCHAR (128)	Нет	FK	Имя выполняемого пакета при объяснении динамического оператора или имя исходного файла при объяснении статического оператора SQL.
SOURCE_SCHEMA	VARCHAR (128)	Нет	FK	Схема или спецификатор источника требования объяснения.
EXPLAIN_LEVEL	CHAR(1)	Нет	FK	Уровень информации объяснения, к которой относится эта строка.
STMTNO	INTEGER	Нет	FK	Номер оператора в пакете, к которому относится эта информация объяснения.
SECTNO	INTEGER	Нет	FK	Номер раздела в пакете, к которому относится эта информация объяснения.
STREAM_ID	INTEGER	Нет	Нет	Уникальный ID для этого потока данных указанной операции.
SOURCE_TYPE	CHAR(1)	Нет	Нет	Указывает источник этого потока данных: **O** Операция **D** Объект данных
SOURCE_ID	SMALLINT	Нет	Нет	Уникальный ID операции в запросе, которая является источником потока данных. -1, если SOURCE_TYPE - 'D'.
TARGET_TYPE	CHAR(1)	Нет	Нет	Указывает назначение этого потока данных: **O** Операция **D** Объект данных
TARGET_ID	SMALLINT	Нет	Нет	Уникальный ID операции в запросе, которая является назначением потока данных. -1, если TARGET_TYPE - 'D'.
OBJECT_SCHEMA	VARCHAR (128)	Да	Нет	Схема, к которой принадлежит объект данных - источник или назначение потока. Null, если и SOURCE_TYPE, и TARGET_TYPE - 'O'.
OBJECT_NAME	VARCHAR (128)	Да	Нет	Имя объекта данных - источника или назначения потока. Null, если и SOURCE_TYPE, и TARGET_TYPE - 'O'.
STREAM_COUNT	DOUBLE	Нет	Нет	Оценка мощности потока данных.
COLUMN_COUNT	SMALLINT	Нет	Нет	Число столбцов в потоке данных.
PREDICATE_ID	INTEGER	Нет	Нет	Если поток - часть подзапроса для предиката, ID этого предиката, иначе -1.

Таблица 42. Таблица EXPLAIN_STREAM (продолжение)

Название столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
COLUMN_NAMES	CLOB(1M)	Да	Нет	Имена и информация упорядочивания столбцов, входящих в поток. Имена имеют следующий формат: `NAME1(A)+NAME2(D)+NAME3+NAME4` где *(A)* указывает столбец с восходящим порядком, *(D)* - столбец с нисходящим порядком, а отсутствие информации о порядке указывает, что столбец неупорядочен или его порядок не используется.
PMID	SMALLINT	No	Нет	ID карты разделения.
SINGLE_NODE	CHAR (5)	Да	Нет	Указывает, является ли поток данных одноразделным или многоразделным. **MULT** На нескольких разделах **COOR** На узле координатора **HASH** Направляет хеш-функцией **RID** Направляется ID строки **FUNC** Направляется функцией (PARTITION() или NODENUMBER()) **CORR** Направляется значением корреляции **Числовой** Направляется на отдельный предопределенный узел
PARTITION_COLUMNS	CLOB(64K)	Да	Нет	Список столбцов, по которым разделяется этот поток данных.

Таблица ADVISE_INDEX

В таблицу ADVISE_INDEX записываются рекомендуемые индексы.

Таблица 43. Таблица ADVISE_INDEX

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
EXPLAIN_REQUESTER	VARCHAR (128)	Нет	Нет	ID авторизации пользователя, потребовавшего объяснение.
EXPLAIN_TIME	TIMESTAMP	Нет	Нет	Время запуска требования объяснения.

Таблица 43. Таблица ADVISE_INDEX (продолжение)

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
SOURCE_NAME	VARCHAR (128)	Нет	Нет	Имя выполняемого пакета при объяснении динамического оператора или имя исходного файла при объяснении статического оператора SQL.
SOURCE_SCHEMA	VARCHAR (128)	Нет	Нет	Схема или спецификатор источника требования объяснения.
EXPLAIN_LEVEL	CHAR(1)	Нет	Нет	Уровень информации объяснения, к которой относится эта строка.
STMTNO	INTEGER	Нет	Нет	Номер оператора в пакете, к которому относится эта информация объяснения.
SECTNO	INTEGER	Нет	Нет	Номер раздела в пакете, к которому относится эта информация объяснения.
QUERYNO	INTEGER	Нет	Нет	Числовой идентификатор для объясняемого оператора SQL. Для динамических операторов SQL (за исключением оператора SQL EXPLAIN), выданных через CLP или CLI, по умолчанию используется последовательно возрастающие значения. В остальных случаях по умолчанию используется значение STMTNO для статических операторов SQL и 1 для динамических операторов SQL.
QUERYTAG	CHAR(20)	Нет	Нет	Тэг идентификатора для каждого объясняемого оператора SQL. Для динамических операторов SQL, выданных через CLP (за исключением оператора SQL EXPLAIN), по умолчанию используется значение 'CLP'. Для динамических операторов SQL, выданных через CLI (за исключением оператора SQL EXPLAIN), по умолчанию используется значение 'CLI'. В остальных случаях значение по умолчанию - это пробелы.
NAME	VARCHAR (128)	Нет	Нет	Имя индекса.
CREATOR	VARCHAR (128)	Нет	Нет	Спецификатор имени индекса.
TBNAME	VARCHAR (128)	Нет	Нет	Имя таблицы или псевдоним, для которого определен индекс.
TBCREATOR	VARCHAR (128)	Нет	Нет	Спецификатор имени таблицы.
COLNAMES	CLOB(64K)	Нет	Нет	Список имен столбцов.
UNIQUERULE	CHAR(1)	Нет	Нет	Правило уникальности: D = Повторения допустимы P = Первичный индекс U = Допустимы только уникальные значения
COLCOUNT	SMALLINT	Нет	Нет	Число столбцов в ключе плюс число включаемых столбцов, если они есть.

Таблица 43. Таблица ADVISE_INDEX (продолжение)

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
IID	SMALLINT	Нет	Нет	Внутренний ID индекса.
NLEAF	INTEGER	Нет	Нет	Число конечных страниц; –1, если статистика не собиралась.
NLEVELS	SMALLINT	Нет	Нет	Число уровней индекса; –1, если статистика не собиралась.
FULLKEYCARD	BIGINT	Нет	Нет	Число различных полных значений ключа; –1, если статистика не собиралась.
FIRSTKEYCARD	BIGINT	Нет	Нет	Число различных первых значений ключа; –1, если статистика не собиралась.
CLUSTERRATIO	SMALLINT	Нет	Нет	Степень кластеризации данных для индекса; –1, если статистика не собиралась или же если собиралась подробная статистика для индекса (в последнем случае вместо нее используется CLUSTERFACTOR)
CLUSTERFACTOR	DOUBLE	Нет	Нет	Более точное значение степени кластеризации, или –1, если подробная статистика индекса не собиралась или же если индекс определен для псевдонима.
USERDEFINED	SMALLINT	Нет	Нет	Определяется пользователем.
SYSTEM_REQUIRED	SMALLINT	Нет	Нет	1, если индекс необходим для первичного ключа или ключа уникальности ИЛИ это индекс по столбцу идентификатор объекта (OID) типизированной таблицы. 2, если индекс необходим для первичного ключа или ключа уникальности И это индекс по столбцу идентификатор объекта (OID) типизированной таблицы. 0 в остальных случаях.
CREATE_TIME	TIMESTAMP	Нет	Нет	Время создания индекса.
STATS_TIME	TIMESTAMP	Да	Нет	Момент последнего изменения записанной статистики для этого индекса. Null, если статистика недоступна.
PAGE_FETCH_PAIRS	VARCHAR(254)	Нет	Нет	Список пар целых числе в символьном виде. Каждая пара представляет число страниц в возможном буфере и число выборок страниц для просмотра таблицы при использовании индекса с этим возможным буфером. (Строка нулевой длины, если данные недоступны.)
REMARKS	VARCHAR(254)	Да	Нет	Пользовательский комментарий или null.
DEFINER	VARCHAR (128)	Нет	Нет	Пользователь - создатель индекса.
CONVERTED	CHAR(1)	Нет	Нет	Зарезервирован для будущего использования.

Таблица 43. Таблица ADVISE_INDEX (продолжение)

Имя столбца	Тип данных	Допусти-мость пустых значений	Ключ	Описание
SEQUENTIAL_PAGES	INTEGER	Нет	Нет	Число конечных страниц, расположенных на диске в порядке ключей индекса, без пропусков между ними или с небольшими пропусками. (–1, если статистика недоступна.)
DENSITY	INTEGER	Нет	Нет	Отношение SEQUENTIAL_PAGES к числу страниц в диапазоне страниц, занимаемом индексом, выраженное в процентах (целое от 0 до 100, –1, если статистика недоступна.)
FIRST2KEYCARD	BIGINT	Нет	Нет	Число ключей, различных по первым двум столбцам индекса (–1, если статистики нет или она неприменима)
FIRST3KEYCARD	BIGINT	Нет	Нет	Число ключей, различных по первым трем столбцам индекса (–1, если статистики нет или она неприменима)
FIRST4KEYCARD	BIGINT	Нет	Нет	Число ключей, различных по первым четырем столбцам индекса (–1, если статистики нет или она неприменима)
PCTFREE	SMALLINT	Нет	Нет	Процент свободного места на конечных страницах индекса, оставляемого при начальном построении индекса. Это пространство доступна для вставок после построения индекса.
UNIQUE_COLCOUNT	SMALLINT	Нет	Нет	Число столбцов, требуемых для ключа уникальности. Всегда <=COLCOUNT. < COLCOUNT, только если есть включаемые столбцы. –1 если индекс не является индексом уникальности (допускает повторения).
MINPCTUSED	SMALLINT	Нет	Нет	Если не нуль, разрешена реорганизация индекса без отключения, а значение используется как порог минимального используемого пространства перед слиянием страниц.
REVERSE_SCANS	CHAR(1)	Нет	Нет	Y = Индекс поддерживает обратный просмотр N = Индекс не поддерживает обратный просмотр
USE_INDEX	CHAR(1)	Да	Нет	Y = индекс рекомендован или оценен N = индекс не рекомендован
CREATION_TEXT	CLOB(1M)	Нет	Нет	Оператор SQL для создания индекса.
PACKED_DESC	BLOB(20M)	Да	Нет	Внутреннее описание таблицы.

Таблица ADVISE_WORKLOAD

Таблица ADVISE_WORKLOAD содержит операторы, составляющие рабочую нагрузку. Подробности о рабочей нагрузке смотрите в разделе *Administration Guide: Performance*.

Таблица 44. Таблица ADVISE_WORKLOAD

Имя столбца	Тип данных	Допустимость пустых значений	Ключ	Описание
WORKLOAD_NAME	CHAR(128)	Нет	Нет	Имя собрания операторов SQL (рабочей нагрузки), к которой принадлежат операторы.
STATEMENT_NO	INTEGER	Нет	Нет	Номер оператора в рабочей нагрузке, к которому относится эта информация объяснения.
STATEMENT_TEXT	CLOB(1M)	Нет	Нет	Содержание оператора SQL.
STATEMENT_TAG	VARCHAR(256)	Нет	Нет	Тэг идентификатора для каждого объясняемого оператора SQL.
FREQUENCY	INTEGER	Нет	Нет	Число вхождений этого оператора в рабочую нагрузку.
IMPORTANCE	DOUBLE	Нет	Нет	Важность оператора.
COST_BEFORE	DOUBLE	Да	Нет	Стоимость (в условных единицах времени) запроса, если рекомендованные индексы не созданы.
COST_AFTER	DOUBLE	Да	Нет	Стоимость (в условных единицах времени) запроса, если рекомендованные индексы созданы.

Определения таблиц для таблиц объяснения

Перед запуском объяснения должны быть созданы таблицы объяснения. Следующие определения создают необходимые таблицы объяснения:

- “Определение таблицы EXPLAIN_ARGUMENT” на стр. 581
- “Определение таблицы EXPLAIN_INSTANCE” на стр. 582
- “Определение таблицы EXPLAIN_OBJECT” на стр. 583
- “Определение таблицы EXPLAIN_OPERATOR” на стр. 584
- “Определение таблицы EXPLAIN_PREDICATE” на стр. 585
- “Определение таблицы EXPLAIN_STATEMENT” на стр. 586
- “Определение таблицы EXPLAIN_STREAM” на стр. 587
- “Определение таблицы ADVISE_INDEX” на стр. 588
- “Определение таблицы ADVISE_WORKLOAD” на стр. 590

Другой способ создания этих таблиц - использовать пример входного сценария процессора командной строки из файла EXPLAIN.DDL, расположенного в подкаталоге 'misc' каталога 'sqllib'. Соединитесь с базой данных, в которой надо создать таблицы объяснения. Затем введите команду `db2 -tf EXPLAIN.DDL`, и эти таблицы будут созданы.

Определение таблицы EXPLAIN_ARGUMENT

```
CREATE TABLE EXPLAIN_ARGUMENT ( EXPLAIN_REQUESTER   VARCHAR(128)  NOT NULL,
                                EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                SOURCE_NAME       VARCHAR(128) NOT NULL,
                                SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                  EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                  STMTNO            INTEGER      NOT NULL,
                                  SECTNO            INTEGER      NOT NULL,
                                   OPERATOR_ID       INTEGER      NOT NULL,
                                  ARGUMENT_TYPE      CHAR(8)       NOT NULL,
                                  ARGUMENT_VALUE     VARCHAR(1024) NOT NULL,
                                  LONG_ARGUMENT_VALUE CLOB(1M)      NOT LOGGED,
                                    FOREIGN KEY (EXPLAIN_REQUESTER,
                                                 EXPLAIN_TIME,
                                                 SOURCE_NAME,
                                                 SOURCE_SCHEMA,
                                                 EXPLAIN_LEVEL,
                                                 STMTNO,
                                                 SECTNO)
                                    REFERENCES EXPLAIN_STATEMENT
                                    ON DELETE CASCADE )
```

Определение таблицы EXPLAIN_INSTANCE

```
CREATE TABLE EXPLAIN_INSTANCE ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                SOURCE_NAME       VARCHAR(128) NOT NULL,
                                SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                  EXPLAIN_OPTION    CHAR(1)      NOT NULL,
                                  SNAPSHOT_TAKEN    CHAR(1)      NOT NULL,
                                  DB2_VERSION       CHAR(7)      NOT NULL,
                                  SQL_TYPE          CHAR(1)      NOT NULL,
                                  QUERYOPT          INTEGER      NOT NULL,
                                  BLOCK             CHAR(1)      NOT NULL,
                                  ISOLATION         CHAR(2)      NOT NULL,
                                  BUFFPAGE          INTEGER      NOT NULL,
                                  AVG_APPLS         INTEGER      NOT NULL,
                                  SORTHEAP          INTEGER      NOT NULL,
                                  LOCKLIST          INTEGER      NOT NULL,
                                  MAXLOCKS          SMALLINT     NOT NULL,
                                  LOCKS_AVAIL       INTEGER      NOT NULL,
                                  CPU_SPEED         DOUBLE       NOT NULL,
                                  REMARKS           VARCHAR(254),
                                  DBHEAP            INTEGER      NOT NULL,
                                  COMM_SPEED        DOUBLE       NOT NULL,
                                  PARALLELISM       CHAR(2)      NOT NULL,
                                  DATAJOINER        CHAR(1)      NOT NULL,
                                       PRIMARY KEY (EXPLAIN_REQUESTER,
                                                    EXPLAIN_TIME,
                                                    SOURCE_NAME,
                                                      SOURCE_SCHEMA))
```

Определение таблицы EXPLAIN_OBJECT

```
CREATE TABLE EXPLAIN_OBJECT ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                              EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                              SOURCE_NAME       VARCHAR(128) NOT NULL,
                              SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                STMTNO            INTEGER      NOT NULL,
                                SECTNO            INTEGER      NOT NULL,
                              OBJECT_SCHEMA     VARCHAR(128) NOT NULL,
                              OBJECT_NAME       VARCHAR(128) NOT NULL,
                              OBJECT_TYPE       CHAR(2)      NOT NULL,
                              CREATE_TIME       TIMESTAMP,
                              STATISTICS_TIME   TIMESTAMP,
                              COLUMN_COUNT      SMALLINT     NOT NULL,
                              ROW_COUNT         INTEGER      NOT NULL,
                              WIDTH             INTEGER      NOT NULL,
                              PAGES             INTEGER      NOT NULL,
                              DISTINCT          CHAR(1)      NOT NULL,
                              TABLESPACE_NAME   VARCHAR(128),
                              OVERHEAD          DOUBLE       NOT NULL,
                              TRANSFER_RATE     DOUBLE       NOT NULL,
                              PREFETCHSIZE      INTEGER      NOT NULL,
                              EXTENTSIZE        INTEGER      NOT NULL,
                              CLUSTER           DOUBLE       NOT NULL,
                              NLEAF             INTEGER      NOT NULL,
                              NLEVELS           INTEGER      NOT NULL,
                              FULLKEYCARD       BIGINT       NOT NULL,
                              OVERFLOW          INTEGER      NOT NULL,
                              FIRSTKEYCARD      BIGINT       NOT NULL,
                              FIRST2KEYCARD     BIGINT       NOT NULL,
                              FIRST3KEYCARD     BIGINT       NOT NULL,
                              FIRST4KEYCARD     BIGINT       NOT NULL,
                              SEQUENTIAL_PAGES  INTEGER      NOT NULL,
                              DENSITY           INTEGER      NOT NULL,
                                  FOREIGN KEY (EXPLAIN_REQUESTER,
                                               EXPLAIN_TIME,
                                               SOURCE_NAME,
                                               SOURCE_SCHEMA,
                                               EXPLAIN_LEVEL,
                                               STMTNO,
                                               SECTNO)
                                  REFERENCES EXPLAIN_STATEMENT
                                  ON DELETE CASCADE )
```

Определение таблицы EXPLAIN_OPERATOR

```
CREATE TABLE EXPLAIN_OPERATOR ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                SOURCE_NAME       VARCHAR(128) NOT NULL,
                                SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                  EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                  STMTNO            INTEGER      NOT NULL,
                                  SECTNO            INTEGER      NOT NULL,
                                   OPERATOR_ID       INTEGER      NOT NULL,
                                  OPERATOR_TYPE     CHAR(6)      NOT NULL,
                                  TOTAL_COST        DOUBLE       NOT NULL,
                                  IO_COST           DOUBLE       NOT NULL,
                                  CPU_COST          DOUBLE       NOT NULL,
                                  FIRST_ROW_COST    DOUBLE       NOT NULL,
                                  RE_TOTAL_COST     DOUBLE       NOT NULL,
                                  RE_IO_COST        DOUBLE       NOT NULL,
                                  RE_CPU_COST       DOUBLE       NOT NULL,
                                  COMM_COST         DOUBLE       NOT NULL,
                                  FIRST_COMM_COST   DOUBLE       NOT NULL,
                                  REMOTE_TOTAL_COST DOUBLE       NOT NULL,
                                  REMOTE_COMM_COST  DOUBLE       NOT NULL,
                                    FOREIGN KEY (EXPLAIN_REQUESTER,
                                                 EXPLAIN_TIME,
                                                 SOURCE_NAME,
                                                 SOURCE_SCHEMA,
                                                 EXPLAIN_LEVEL,
                                                 STMTNO,
                                                 SECTNO)
                                    REFERENCES EXPLAIN_STATEMENT
                                    ON DELETE CASCADE )
```

Определение таблицы EXPLAIN_PREDICATE

```
CREATE TABLE EXPLAIN_PREDICATE ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                 EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                 SOURCE_NAME       VARCHAR(128) NOT NULL,
                                 SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                   EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                   STMTNO            INTEGER      NOT NULL,
                                   SECTNO            INTEGER      NOT NULL,
                                    OPERATOR_ID       INTEGER      NOT NULL,
                                 PREDICATE_ID      INTEGER      NOT NULL,
                                    HOW_APPLIED       CHAR(5)      NOT NULL,
                                    WHEN_EVALUATED    CHAR(3)      NOT NULL,
                                    RELOP_TYPE        CHAR(2)      NOT NULL,
                                    SUBQUERY          CHAR(1)      NOT NULL,
                                    FILTER_FACTOR     DOUBLE       NOT NULL,
                                    PREDICATE_TEXT    CLOB(1M)     NOT LOGGED,
                                     FOREIGN KEY (EXPLAIN_REQUESTER,
                                                  EXPLAIN_TIME,
                                                  SOURCE_NAME,
                                                  SOURCE_SCHEMA,
                                                  EXPLAIN_LEVEL,
                                                  STMTNO,
                                                  SECTNO)
                                     REFERENCES EXPLAIN_STATEMENT
                                     ON DELETE CASCADE )
```

Определение таблицы EXPLAIN_STATEMENT

```
CREATE TABLE EXPLAIN_STATEMENT ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                 EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                 SOURCE_NAME       VARCHAR(128) NOT NULL,
                                 SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                   EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                   STMTNO            INTEGER      NOT NULL,
                                   SECTNO            INTEGER      NOT NULL,
                                    QUERYNO           INTEGER      NOT NULL,
                                    QUERYTAG          CHAR(20)     NOT NULL,
                                    STATEMENT_TYPE    CHAR(2)      NOT NULL,
                                    UPDATABLE         CHAR(1)      NOT NULL,
                                    DELETABLE         CHAR(1)      NOT NULL
                                   TOTAL_COST        DOUBLE       NOT NULL,
                                    STATEMENT_TEXT    CLOB(1M)     NOT NULL
                                                                   NOT LOGGED,
                                    SNAPSHOT          BLOB(10M)    NOT LOGGED,
                                    QUERY_DEGREE      INTEGER      NOT NULL,
                                        PRIMARY KEY (EXPLAIN_REQUESTER,
                                                     EXPLAIN_TIME,
                                                     SOURCE_NAME,
                                                     SOURCE_SCHEMA,
                                                     EXPLAIN_LEVEL,
                                                     STMTNO,
                                                        SECTNO),
                                     FOREIGN KEY (EXPLAIN_REQUESTER,
                                                     EXPLAIN_TIME,
                                                     SOURCE_NAME,
                                                        SOURCE_SCHEMA)
                                        REFERENCES EXPLAIN_INSTANCE
                                     ON DELETE CASCADE )
```

Определение таблицы EXPLAIN_STREAM

```
CREATE TABLE EXPLAIN_STREAM ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                              EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                              SOURCE_NAME       VARCHAR(128) NOT NULL,
                              SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                STMTNO            INTEGER      NOT NULL,
                                SECTNO            INTEGER      NOT NULL,
                              STREAM_ID         INTEGER      NOT NULL,
                              SOURCE_TYPE       CHAR(1)      NOT NULL,
                              SOURCE_ID         SMALLINT     NOT NULL,
                              TARGET_TYPE       CHAR(1)      NOT NULL,
                              TARGET_ID         SMALLINT     NOT NULL,
                              OBJECT_SCHEMA     VARCHAR(128),
                              OBJECT_NAME       VARCHAR(128),
                              STREAM_COUNT      DOUBLE       NOT NULL,
                              COLUMN_COUNT      SMALLINT     NOT NULL,
                              PREDICATE_ID      INTEGER      NOT NULL,
                              COLUMN_NAMES      CLOB(1M)     NOT LOGGED,
                              PMID              SMALLINT     NOT NULL,
                              SINGLE_NODE       CHAR(5),
                              PARTITION_COLUMNS CLOB(64K)    NOT LOGGED,
                                  FOREIGN KEY (EXPLAIN_REQUESTER,
                                               EXPLAIN_TIME,
                                               SOURCE_NAME,
                                               SOURCE_SCHEMA,
                                               EXPLAIN_LEVEL,
                                               STMTNO,
                                               SECTNO)
                                  REFERENCES EXPLAIN_STATEMENT
                                  ON DELETE CASCADE )
```

Определение таблицы ADVISE_INDEX

```
CREATE TABLE ADVISE_INDEX (EXPLAIN_REQUESTER VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           EXPLAIN_TIME      TIMESTAMP    NOT NULL
                              WITH DEFAULT CURRENT TIMESTAMP,
                           SOURCE_NAME       VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           SOURCE_SCHEMA     VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           EXPLAIN_LEVEL     CHAR(1)      NOT NULL
                              WITH DEFAULT '',
                           STMTNO            INTEGER      NOT NULL
                              WITH DEFAULT 0,
                           SECTNO            INTEGER      NOT NULL
                              WITH DEFAULT 0,
                           QUERYNO           INTEGER      NOT NULL
                              WITH DEFAULT 0,
                           QUERYTAG          CHAR(20)     NOT NULL
                              WITH DEFAULT '',
                           NAME              VARCHAR(128) NOT NULL,
                           CREATOR           VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           TBNAME            VARCHAR(128) NOT NULL,
                           TBCREATOR         VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           COLNAMES          CLOB(64K)    NOT NULL,
                           UNIQUERULE        CHAR(1)      NOT NULL
                              WITH DEFAULT '',
                           COLCOUNT          SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           IID               SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           NLEAF             INTEGER      NOT NULL
                              WITH DEFAULT 0,
                           NLEVELS           SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           FIRSTKEYCARD      BIGINT       NOT NULL
                              WITH DEFAULT 0,
                           FULLKEYCARD       BIGINT       NOT NULL
                              WITH DEFAULT 0,
                           CLUSTERRATIO      SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           CLUSTERFACTOR     DOUBLE       NOT NULL
                              WITH DEFAULT 0,
                           USERDEFINED       SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           SYSTEM_REQUIRED   SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           CREATE_TIME       TIMESTAMP    NOT NULL
                              WITH DEFAULT CURRENT TIMESTAMP,
                           STATS_TIME        TIMESTAMP
                              WITH DEFAULT CURRENT TIMESTAMP,
                           PAGE_FETCH_PAIRS  VARCHAR(254) NOT NULL
                              WITH DEFAULT '',
                           REMARKS           VARCHAR(254)
```

```
                    WITH DEFAULT '',
DEFINER             VARCHAR(128) NOT NULL
                    WITH DEFAULT '',
CONVERTED           CHAR(1) NOT NULL
                    WITH DEFAULT '',
SEQUENTIAL_PAGES    INTEGER      NOT NULL
                    WITH DEFAULT 0,
DENSITY             INTEGER      NOT NULL
                    WITH DEFAULT 0,
FIRST2KEYCARD       BIGINT       NOT NULL
                    WITH DEFAULT 0,
FIRST3KEYCARD       BIGINT       NOT NULL
                    WITH DEFAULT 0,
FIRST4KEYCARD       BIGINT       NOT NULL
                    WITH DEFAULT 0,
PCTFREE             SMALLINT     NOT NULL
                    WITH DEFAULT -1,
UNIQUE_COLCOUNT     SMALLINT     NOT NULL
                    WITH DEFAULT -1,
MINPCTUSED          SMALLINT     NOT NULL
                    WITH DEFAULT 0,
REVERSE_SCANS       CHAR(1)      NOT NULL
                    WITH DEFAULT 'N',
USE_INDEX           CHAR(1),
CREATION_TEXT       CLOB(1M)     NOT NULL
                    NOT LOGGED WITH DEFAULT '',
PACKED_DESC         BLOB(1M)     NOT LOGGED)
```

Определение таблицы ADVISE_WORKLOAD

```
CREATE TABLE ADVISE_WORKLOAD (WORKLOAD_NAME  CHAR(128)    NOT NULL
                                WITH DEFAULT 'WK0',
                              STATEMENT_NO   INTEGER      NOT NULL
                                WITH DEFAULT 1,
                              STATEMENT_TEXT CLOB(1M)     NOT NULL NOT LOGGED,
                              STATEMENT_TAG  VARCHAR(256) NOT NULL
                                 WITH DEFAULT '',
                              FREQUENCY      INTEGER      NOT NULL
                                WITH DEFAULT 1,
                              IMPORTANCE     DOUBLE       NOT NULL
                                WITH DEFAULT 1,
                              COST_BEFORE    DOUBLE,
                              COST_AFTER     DOUBLE)
```

Приложение C. Инструменты объяснения SQL

Инструмент **db2expln** описывает план доступа, который выбирается для статических операторов SQL в пакетах, хранимых в таблицах системного каталога. Этот инструмент можно использовать для получения быстрого объяснения выбранного плана доступа для пакетов, данные объяснения для которых не были захвачены во время связывания.

Инструмент **dynexpln** описывает план доступа, выбранный для динамических операторов. Этот инструмент создает для операторов статический пакет, а для их описания использует **db2expln**.

Эти инструменты объяснения можно использовать для просмотра плана доступа, выбранного для конкретного оператора SQL. Такой план доступа можно также просмотреть с помощью встроенной возможность объяснения (“Глава 7. Возможность объяснения SQL” на стр. 225) в сочетании с наглядным объяснением. Возможность объяснения позволяет объяснять как динамические, так и статические операторы SQL. Эта возможность отличается от инструментов объяснения только тем, что при использовании наглядного объяснения информация представляется в графической форме. Во всем прочем уровень детальности в обоих методах одинаковый.

Чтобы полностью использовать вывод инструментов `db2expln` и `dynexpln`, следует просмотреть:

- Информацию о различных операторах SQL, поддерживаемых этими инструментами, и относящихся к ним понятиях (например, предикатах в операторе SELECT).
- Информацию о назначении пакета (плана доступа). (Эту информацию смотрите в разделе “Концепции доступа к данным и оптимизация” на стр. 169.)
- Информацию о назначении и содержании таблиц системного каталога. (Смотрите эту информацию в справочнике *SQL Reference*.)
- Информацию о других понятиях можно найти в разделе “Часть 2. Настройка производительности прикладных программ” на стр. 41.

Информация об инструментах `db2expln` и `dynexpln` содержится в следующих темах:

- Запуск db2expln и dynexpln
- Синтаксис и параметры db2expln
- Замечания по использованию db2expln
- Синтаксис и параметры dynexpln

- Замечания по использованию dynexpln
- Описание вывода db2expln и dynexpln
- Примеры вывода db2expln и dynexpln.

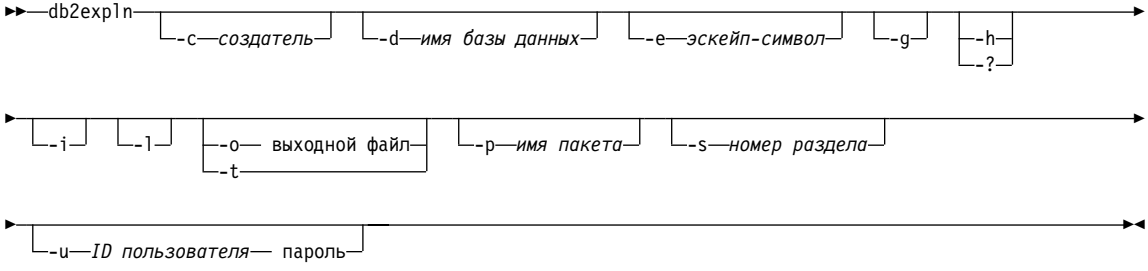
Запуск db2expln и dynexpln

Инструменты объяснения (db2expln и dynexpln) находятся в подкаталоге misc каталога вашего экземпляра sqllib. Если db2expln и dynexpln нет в данном каталоге, они должны быть в каталоге, который указан в переменной среды PATH.

Программа db2expln соединяется с базой данных и при первом обращении к этой базе данных связывает себя с ней, используя файл db2expln.bnd. Файл db2expln.bnd находится в подкаталоге bnd каталога sqllib.

Чтобы запустить db2expln, вам требуется привилегия SELECT для производных таблиц системного каталога и полномочия EXECUTE для пакета db2expln. Для выполнения dynexpln необходимо обладать полномочиями BINDADD для базы данных, используемая для соединения с базой данных схема должна существовать или вы должны иметь полномочия EXPLICIT_SCHEMA для этой базы данных, а также вы должны обладать всеми привилегиями, необходимыми для объяснения этих операторов SQL. (Заметим, что все нужные уровни авторизации можно получить вместе с полномочиями SYSADM или DBADM автоматически.)

Синтаксис и параметры db2expln



Где:

-c создатель

ID пользователя создателя пакета.

Если не задать эту опцию, появится соответствующая подсказка.

При задании имени создателя можно использовать символы подстановки - знак процента (%) и подчеркивание (_), так же, как и в предикате LIKE.

-d имя базы данных
Имя базы данных, содержащей пакеты, которые следует объяснить.

Если не задать эту опцию, появится соответствующая подсказка.

-e эскейп-символ
Используется для задания символа, который будет интерпретироваться как эскейп-символ, а не как символ поиска по шаблону.

Например, команда `db2expln` для объяснения пакета TESTID.CALC% будет выглядеть так: `db2expln -c TESTID -p CALC%.` Однако эта команда будет объяснять и все другие планы, имена которых начинаются с `CALC`. Чтобы объяснить только пакет TESTID.CALC%, нужно воспользоваться эскейп-символом. Изменив эту команду на `db2expln -c TESTID -e ! -p CALC!%`, вы зададите, что символ `!` будет использоваться в качестве эскейп-символа, а `!%` будет интерпретироваться как символ `%`.

-g Показать диаграммы плана оптимизатора. Проверяется каждый раздел, и конструируется исходная диаграмма плана оптимизатора (как и в наглядном объяснении). Учтите, что сгенерированная диаграмма может не соответствовать исходному плану.

-h или -?
Получить справочную информацию о входных параметрах. При задании этой опции все другие опции игнорируются.

-i Выводить в объясненном плане ID операторов. При использовании ID операторов вывод из `db2expln` можно сопоставлять с выводом возможности объяснения.

-l Если задать эту опцию, в имени пакета можно будет использовать символы как нижнего, так верхнего регистра. Если опция **-l** не задана, имя пакета будет преобразовано в верхний регистр

-o выходной файл
Имя файла, куда `db2expln` будет записывать результаты.

Если задать опцию **-o** без имени файла, будет предложено ввести его. Имя файла по умолчанию - `db2expln.out`.

-t Вывод направляется на терминал.

Если не задать опции **-o** или **-t**, вам предложат ввести имя файла; по умолчанию вывод направляется на терминал.

-p имя пакета
Имя объясняемого пакета.

Если не задать эту опцию, появится соответствующая подсказка.

При задании имени пакета можно использовать символы подстановки - знак процента (%) и подчеркивание (_), так же, как и в предикате LIKE.

-s номер раздела
Номер объясняемого раздела в пакете. Чтобы объяснить все разделы в пакете, можно задать ноль (0). Если аргументами для создателя пакета (**-c**) или для имени пакета (**-p**) подразумевается объяснение нескольких пакетов, а следовательно и нескольких разделов, значение для раздела, если оно задано, будет заменено нулем (0).

Если не задать эту опцию, появится соответствующая подсказка.

Номера разделов можно найти при помощи запроса к системному каталогу SYSCAT.STATEMENTS. (Описание таблиц системного каталога смотрите в справочнике *SQL Reference*.)

-u ID пользователя пароль
Использовать при соединении с базой данных заданные ID пользователя и пароль.

Как ID пользователя, так и пароль должны быть допустимыми в соответствии с соглашениями об именовании DB2 и распознаваться базой данных.

Некоторые из вышеприведенных флагов опций могут иметь специальный смысл в вашей операционной системе и в связи с этим неправильно интерпретироваться в командной строке `db2expln` . Однако почти всегда существует возможность ввести эти символы, предварив их эскейп-символом. Дополнительную информацию смотрите в руководстве пользователя своей операционной системы.

Справочные сообщения и сообщения об исходном состоянии, генерируемые инструментом `db2expln`, записываются в стандартный вывод. Все подсказки и прочие сообщения о состоянии, генерируемые этим инструментом объяснения, записываются в стандартный вывод ошибок. Текст объяснения записывается в стандартный вывод или файл в зависимости от выбранной опции вывода.

С помощью опций **-p** и **-c** в одном вызове объяснения можно объяснить несколько планов, если в строчных константах для пакетов и создателей использовать шаблоны LIKE. Один символ можно представлять знаком подчеркивания (_), а несколько символов (в частности, 0) - знаком процента (%).

Например, для объяснения все разделов всех пакетов в базе данных SAMPLE с записью результата в файл **my.exp** введите:

```
db2expln -d SAMPLE -p % -c % -s 0 -o my.exp
```

Замечания по использованию db2expln

Ниже приводятся общие сообщения, выводимые db2expln:

- `No packages found for database <база данных>, package pattern: <создатель>.<пакет>.` (Для базы данных <база данных> пакеты не найдены, шаблон пакета: <создатель>.<пакет>.)

  Это сообщение появится в выводе, если в базе данных не будут найдены пакеты, соответствующие заданному шаблону.

- `Bind messages can be found in db2expln.msg` (В db2expln.msg не удается найти сообщения связывания)

  Это сообщение появится в выводе при неудачном завершении связывании db2expln.bnd. Дополнительную информацию о возникновении этой ошибки можно будет найти в файле db2expln.msg текущего каталога.

- `Section number overridden to 0 for potential multiple packages.` (Номер раздела задан равным 0 в связи возможной обработкой нескольких пакетов.)

  Это сообщение появится в выводе, если db2expln может найти несколько пакетов. Такая ситуация возможна, если в аргументах пакета или создателя используется один из символов подстановки.

- `No static sections qualify from package.` (Статические разделы из пакета не определяются.)

  Это сообщение появляется в выводе, если заданный пакет содержит только динамические операторы SQL, то есть в нем нет статических разделов.

- `Database <база данных>, package <создатель>.<пакет> is not valid. Rebind and then rerun db2expln.` (База данных <база данных>, пакет <создатель>.<пакет> недопустимы. Свяжите их повторно, а затем перезапустите db2expln.)

  Это сообщение появляется в выводе, если текущий заданный пакет недопустим. Как было указано, повторите для плана команду BIND или REBIND, чтобы снова создать в базе данных допустимый пакет, а затем перезапустите db2expln.

- `Section not processed: Produced by unsupported release.` (Раздел не обработан: Сгенерирован неподдерживаемой версией.)

  Это сообщение появится в выводе также, если обрабатываемый в текущий момент пакет был сгенерирован версией DB2, отличной от версии, откуда взят выполняемый файл db2expln. В этом случае воспользуйтесь копией db2expln версии DB2, которая сгенерировала этот раздел.

Исключенные операторы SQL: Следующие операторы не будут объясняться:

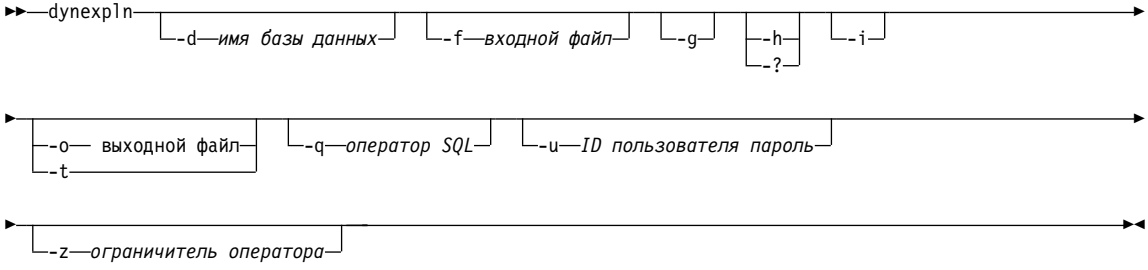
- BEGIN/END DECLARE SECTION
- BEGIN/END COMPOUND
- INCLUDE

- WHENEVER
- COMMIT и ROLLBACK
- CONNECT
- OPEN указатель
- FETCH
- CLOSE указатель
- PREPARE
- EXECUTE
- EXECUTE IMMEDIATE
- DESCRIBE
- Динамический DECLARE CURSOR
- Управляющие операторы SQL

Любые подоператоры в составном операторе SQL могут иметь свои собственные разделы, объяснение которых допускается **db2expln**.

Синтаксис и параметры dynexpln

Где:



-d имя базы данных
Имя базы данных, содержащей пакеты, которые следует объяснить.

Если не задать эту опцию, появится соответствующая подсказка.

-f входной файл
Имя файла, содержащего операторы SQL для объяснения.

Если опция ограничителя оператора (**-z**) не используется, в каждой строке входного файла должен стоять только один оператор SQL. В этот файл можно ввести операторы SQL. Оператор SQL начинается с символа **--** и продолжается до конца строки.

-g Показать диаграммы плана оптимизатора. Проверяется каждый раздел, и конструируется исходная диаграмма плана оптимизатора (как и в

наглядном объяснении). Учтите, что сгенерированная диаграмма может не соответствовать исходному плану.

-h или -?
Получить справочную информацию о входных параметрах. При задании этой опции все другие опции игнорируются.

-i Выводить в объясненном плане ID операторов. При использовании ID операторов вывод из `db2expln` можно сопоставлять с выводом возможности объяснения.

-o выходной файл
Имя файла, куда `db2expln` будет записывать результаты.

-t Вывод направляется на терминал.

Если заданы и вывод (**-o**), и опция **-t**, вывод направляется на терминал.

Если не задать опции выходного файла **-o** или **-t**, будет предложено ввести имя файла; по умолчанию вывод направляется на терминал.

-q оператор SQL
Оператор SQL, который будет объясняться.

Если не задать эту опцию и необязательный параметр входного файла (**-f**), появится подсказка указать оператор SQL, который следует объяснить.

Если заданы и эта опция, и необязательный параметр входного файла (**-f**), `dynexpln` сначала описывает операторы, заданные опцией оператора SQL (**-s**), а затем операторы во входном файле (**-f**).

-u ID пользователя пароль
Использовать при соединении с базой данных заданные ID пользователя и пароль.

Как ID пользователя, так и пароль должны быть допустимыми в соответствии с соглашениями об именовании DB2 и распознаваться базой данных.

-z ограничитель оператора
Этот символ используется для указания, что достигнут конец оператора SQL.

По умолчанию ограничитель оператора отсутствует. Если эта опция не используется, каждая строка файла будет считаться отдельным оператором SQL. При использовании этой опции `dynexpln` будет использовать заданный символ завершения для разделения операторов.

Некоторые из вышеприведенных флагов опций могут иметь специальный смысл для вашей операционной системы и из-за этого неправильно интерпретироваться в командной строке `dynexpln`. Однако почти всегда

существует возможность ввести эти символы, предварив их эскейп-символом. Дополнительную информацию смотрите в руководстве пользователя своей операционной системы.

Если используется опция ограничителя оператора (**-z**), можно ввести несколько операторов, воспользовавшись опцией оператора SQL (**-s**). При этом операторы следует разделить символами завершения.

Справочные сообщения и сообщения об исходном состоянии, генерируемые инструментом dynexpln, записываются в стандартный вывод. Все подсказки и прочие сообщения о состоянии, генерируемые этим инструментом объяснения, записываются в стандартный вывод ошибок. Текст объяснения записывается в стандартный вывод или файл в зависимости от выбранной опции вывода.

Например, чтобы соединиться с базой данных SAMPLE, объяснить все операторы в файле TRYIT и записать результаты в файл my.exp, введите:

```
dynexpln -d SAMPLE -f TRYIT -o my.exp
```

Замечания по использованию dynexpln

Чтобы объяснить динамические операторы, dynexpln создает для этих операторов статическую программу, а затем вызывает db2expln. Чтобы создать статические операторы, dynexpln генерирует с этими операторами простейшую программу на языке C, а затем вызывает прекомпилятор DB2 для создания пакета. (Генерируемая программа на языке C неполная, и ее нельзя скомпилировать; она содержит только информацию, достаточную, чтобы прекомпилятор мог построить пакет.)

Ниже приводятся общие сообщения, выводимые dynexpln:

- Все сообщения об ошибках из db2expln.

  Поскольку dynexpln вызывает db2expln, большая часть сообщений об ошибках db2expln может появиться и в dynexpln.

- `Error connecting to the database.` (Ошибка соединения с базой данных.)

  Это сообщение появляется в выводе, если происходит ошибка при соединении с базой данных. Кроме того, выводится сообщение об ошибке CLI, указывающее, почему не может быть установлено соединение. Устраните причину ошибки и вновь запустите dynexpln.

- `The file "<имя файла>" must be removed before dynexpln will run.` (Перед запуском dynexpln нужно удалить Файл ″<имя файла>″.)

  Это сообщение появляется, если указанный файл существует во время запуска dynexpln. Удалите этот файл или измените имя создаваемого файла, изменив значение переменной среды DYNEXPLN_PACKAGE, и снова запустите dynexpln.

- `The package "<создатель>.<пакет>" must be dropped before dynexpln will run.` (Перед запуском dynexpl нужно отбросить пакет ″<создатель>.<пакет>″.)

  Это сообщение появляется, если данный пакет существует во время запуска `dynexpln`. Отбросьте этот пакет или измените имя создаваемого пакета, изменив значение переменной среды **DYNEXPLN_PACKAGE**, и снова запустите `dynexpln`.

- `Error writing file "<имя файла>".` (Ошибка записи файла ″<имя файла>″.)

  Это сообщение появляется, если невозможно записать данный файл. Убедитесь, что `dynexpln` может записывать файлы в текущем каталоге и вновь запустите программу.

- `Error reading input file "<"<имя файла>">".` (Ошибка чтения входного файла ″<имя файла>″.)

  Это сообщение появляется, если невозможно прочесть файл, заданный в опции **-f**. Убедитесь, что файл существует и что `dynexpln` может его прочесть. Вновь запустите `dynexpln`.

Переменные среды: Вместе с `dynexpln` могут использоваться две переменные среды:

- **DYNEXPLN_OPTIONS** - опции прекомпилятора SQL, которые используются при построении пакета для операторов. Используется тот же синтаксис, что и при вводе команды PREP в командной строке.

  Например: `DYNEXPLN_OPTIONS="OPTLEVEL 5 BLOCKING ALL"`

- **DYNEXPLN_PACKAGE** - имя пакета, который создается в базе данных. В этот пакет помещаются описываемые операторы. Если эта переменная не определена, пакету дается значение по умолчанию **DYNEXPLN**. (В имени этой переменной среды используются только первые восемь символов.)

  Это имя применяется также для генерации имен промежуточных файлов, которые использует `dynexpln`.

Описание вывода db2expln и dynexpln

В выводе db2expln и dynexpln информация объяснения для каждого пакета разбивается на две части:

- Информация о пакете, например дата связывания и соответствующие опции связывания
- Информация о разделе, например номер раздела, за которым следует объясняемый оператор SQL. Под информацией раздела находится вывод объяснения выбранного плана доступа для показанного оператора SQL.

Шаги плана доступа или раздела показаны в порядке их выполнения менеджером баз данных. Каждый из основных шагов представлен заголовком, выровненным по левому краю, а ниже с отступом приводится информация о

данном шаге. Для вывода объяснения плана доступа в левом поле вывода используется линейка отступов. Эта линейка задает также ″область действия″ для операции; в пределах одной операции сначала обрабатываются операции более низкого уровня (с большим отступом вправо), а потом происходит возврат к предыдущему уровню.

Важно помнить, что выбранный план доступа основан на расширенной версии исходного оператора SQL (как показано в выводе). Исходный оператор, например, может привести к активации любого числа триггеров и ограничений. Кроме того, формат этого оператора SQL может быть заменен компонентом переписывания запросов компилятора SQL на эквивалентный, но более эффективный формат. Все это включается в информацию, передаваемую оптимизатору, когда он определяет наиболее эффективный план для данного оператора. Таким образом, план доступа, показанный в выводе объяснения, может существенно отличаться от плана доступа, который можно ожидать для исходного оператора SQL. Встроенная утилита объяснения (смотрите в разделе “Глава 7. Возможность объяснения SQL” на стр. 225) показывает фактический оператор SQL, используемый для оптимизации, в форме SQL-подобного оператора, который создается посредством обратной трансляции внутреннего представления запроса.

При сравнении вывода от `db2expln` или `dynexpln` с выводом утилиты Объяснения может оказаться чрезвычайно полезной опция ID оператора (**-i**). Каждый раз, когда `db2expln` или `dynexpln` начинает обработку нового оператора из возможности объяснения, номер ID оператора будет выводиться слева от объясненного плана. ID операторов можно использовать для сопоставления шагов в различных представлениях плана доступа. Учтите, что не всегда имеется полное соответствие между операциями в выводе объяснения и операциями, показываемыми инструментами `db2expln` и `dynexpln`.

Текст объяснения, который могут генерировать инструменты `db2expln` и `dynexpln`, описывается в следующих темах:

- Обращение к таблице
- Временные таблицы
- Объединения
- Потоки данных
- Операторы Insert, Update и Delete
- Подготовка идентификатора строки (RID)
- Суммирование
- Параллельная обработка
- Обработка оператора объединения
- Различные операторы.

Обращение к таблице

Оператор обращения к таблице сообщает имя и тип таблицы, к которой выполняется обращение. У этого оператора есть два формата:

1. Обычные таблицы трех типов:
   - Имя таблицы, к которой производится обращение:

     ```
     Access Table Name = схема.имя  ID = ts,n
     ```

     где:
     - *схема.имя* - полное имя таблицы, к которой выполняется обращение
     - *ID* - соответствующие данной таблице ID табличного пространства (TABLESPACEID) и ID таблицы (TABLEID) из каталога SYSCAT.TABLES
   - Имя иерархической таблицы, к которой производится обращение:

     ```
     Access Hierarchy Table Name = схема.имя  ID = ts,n
     ```

     где:
     - *схема.имя* - полное имя таблицы, к которой выполняется обращение
     - *ID* - соответствующие данной таблице ID табличного пространства (TABLESPACEID) и ID таблицы (TABLEID) из каталога SYSCAT.TABLES
   - Имя сводной таблицы, к которой производится обращение:

     ```
     Access Summary Table Name = схема.имя  ID = ts,n
     ```

     где:
     - *схема.имя* - полное имя таблицы, к которой выполняется обращение
     - *ID* - соответствующие данной таблице ID табличного пространства (TABLESPACEID) и ID таблицы (TABLEID) из каталога SYSCAT.TABLES
2. Временные таблицы двух типов:
   - ID временной таблицы, к которой производится обращение:

     ```
     Access Temp Table  ID = tn
     ```

     где:
     - *ID* - это соответствующий ID, назначенный инструментом `db2expln`
   - ID объявленной глобальной временной таблицы, к которой производится обращение:

     ```
     Access Global Temp Table  ID = ts,tn
     ```

     где:
     - *ID* - это ID табличного пространства (TABLESPACEID) из каталога SYSCAT.TABLES, соответствующий таблице `ts`;

Более подробное описание доступа обеспечивается дополнительными операторами, которые следуют за оператором обращения к таблице. Эти операторы расположены под оператором обращения к таблице с отступом. Возможны следующие операторы:

- Число столбцов
- Параллельный просмотр
- Направление просмотра
- Способ обращения к строкам
- Намерения блокировок
- Предикаты
- Различные операторы таблиц.

Число столбцов

Следующий оператор указывает число столбцов, используемых из каждой строки таблицы:

```
#Columns = n
```

Параллельный просмотр

Следующий оператор указывает, что менеджер баз данных будет использовать несколько подагентов для чтения из таблицы в параллельном режиме:

```
Parallel Scan
```

Если он отсутствует, таблица будет читаться только одним агентом (или подагентом).

Направление просмотра

Следующий оператор указывает, что менеджер баз данных будет читать строки в обратном порядке:

```
Scan Direction = Reverse
```

Если он отсутствует, просмотр выполняется в прямом порядке - это направление по умолчанию.

Способ обращения к строкам

Для описания способа обращения к нужным строкам таблицы будет выведен один из следующих операторов:

- Оператор `Relation Scan` указывает, что при поиске нужных строк таблица будет просматриваться последовательно.
  - Следующий оператор указывает, что предварительное чтение данных применяться не будет:

    ```
        Relation Scan
      | Prefetch: None
    ```

  - Следующий оператор указывает, что оптимизатор предварительно определил число страниц, которые будут читаться заранее:

```
  Relation Scan
  |  Prefetch: n Pages
```

– Следующий оператор указывает, что должно выполняться предварительное чтение данных:

```
Relation Scan
|  Prefetch: Eligible
```

– Следующий оператор указывает, что идентификация нужных строк и обращение к ним будет проводиться по индексу:

```
Index Scan:  Name = схема.имя  ID = xx
|  Index Columns:
```

где:

- *схема.имя* - полное имя просматриваемого индекса
- *ID* - это соответствующий столбец IID в производной таблице каталога SYSCAT.INDEXES.

После этого оператора следуют строки столбцов (одна строка для каждого столбца в индексе). Каждый столбец в индексе выводится в одном из следующих форматов:

```
n: имя_столбца (Ascending)
n: имя_столбца (Descending)
n: имя_столбца (Include Column)
```

Тип просмотра индекса уточняется с помощью следующих операторов:

- Предикаты ограничения диапазона для индекса выводятся в виде:

```
#Key Columns = n
|  Start Key: xxxxx
|  Stop Key: xxxxx
```

Где `xxxxx` - одно из следующих значений:

- `Start of Index`
- `End of Index`
- `Inclusive Value:` или `Exclusive Value:`

  Включаемое значение ключа (Inclusive Value) будет включено в просмотр индекса. Исключаемое значение ключа (Exclusive Value) не будет включено в просмотр. Значение для каждой части ключа будет представлено одной из следующих строк:

```
n: 'строка'
n: nnn
n: гггг-мм-дд
n: чч:мм:сс
n: гггг-мм-дд чч:мм:сс.нннннн
n: NULL
n: ?
```

Если значение представлено литеральной строкой, выводятся только первые 20 символов. Если в строке больше 20 символов, в конце будет стоять многоточие ... Некоторые ключи нельзя определить до выполнения данного раздела. Это будет показано знаком вопроса ? вместо значения.

- `Index-Only Access`

  Если все необходимые столбцы можно получить из индексного ключа, будет выведен этот оператор, и обращение к табличным данным выполняться не будет.

- Следующий оператор указывает, что предварительное чтение страниц индекса выполняться не будет:

  ```
  Index Prefetch: None
  ```

- Следующий оператор указывает, что должно выполняться предварительное чтение страниц индекса:

  ```
  Index Prefetch: Eligible
  ```

- Следующий оператор указывает, что не будет выполняться предварительное чтение страниц данных:

  ```
  Data Prefetch: None
  ```

- Следующий оператор указывает, что должно выполняться предварительное чтение страниц данных:

  ```
  Data Prefetch: Eligible
  ```

- Если есть предикаты, которые могут быть переданы Менеджеру индекса, чтобы помочь ему при поиске, для указания числа таких предикатов используется следующий оператор:

  ```
  Sargable Index Predicate(s)
  |  #Predicates = n
  ```

– Оператор `Fetch Direct` указывает, что обращение к нужным строкам будет выполняться с использованием ID строк (RID), подготовленных до этого в плане доступа.

Намерения блокировок

Следующий оператор показывают тип блокировок на уровне таблицы и на уровне строки, получаемых при каждом обращении к таблице:

```
Lock Intents
|  Table: xxxx
|  Row  : xxxx
```

Допустимые значения для блокировок таблицы (Table):

- Exclusive (Монопольная)
- Intent Exclusive (Намерение монопольной)
- Intent None (Без назначения)
- Intent Share (Намерение совместной)

- Share (Совместная)
- Share Intent Exclusive (Совместная с намерением монопольной)
- Super Exclusive (Сверхмонопольная)
- Update (Изменение)

Допустимые значения для блокировок строки (Row):

- Exclusive (Монопольная)
- Next Key Exclusive (Следующий ключ монопольный, не появляется в выводе `db2expln`)
- None (Нет)
- Share (Совместная)
- Next Key Share (Следующий ключ совместный)
- Update (Изменение)
- Next Key Weak Exclusive (Следующий ключ слабый монопольный)
- Weak Exclusive (Слабая монопольная)

Объяснение этих типов блокировок смотрите в разделе “Атрибуты блокировок” на стр. 53.

Предикаты

Информацию об используемых в плане доступа предикатах дают два оператора:

1. Следующий оператор указывает число предикатов, которые будут оцениваться при возврате данных:

   ```
   Residual Predicate(s)
   |  #Predicates = n
   ```

2. Следующий оператор указывает число предикатов, которые будут оцениваться при обращении к данным: В это число предикатов не входят стековые операции, например суммирование и сортировка.

   ```
   Sargable Predicate(s)
   |  #Predicates = n
   ```

Число предикатов, выводимое этими операторами, может не соответствовать числу предикатов в операторе SQL, так как предикаты могут быть:

- Применены несколько раз в одном запросе
- Преобразованы и расширены неявными предикатами в процессе оптимизации запроса
- Преобразованы и сжаты в меньшее количество предикатов в процессе оптимизации запроса.

Различные операторы таблиц

- Следующий оператор указывает, что обращение будет выполняться только к одной строке:

```
Single Record
```

- Следующий оператор появляется, если при обращении к таблице используется уровень изоляции, отличающийся от уровня изоляции пакета:

  ```
  Isolation Level: xxxx
  ```

  Отличающийся уровень изоляции может использоваться по многим причинам, например:

  – Пакет был связан с уровнем изоляции Многократное чтение и воздействует на ограничения реляционной целостности; уровень изоляции доступа к родительской таблице для проверки ограничений реляционной целостности понижен до стабильности на уровне указателя для предотвращения удержания ненужных блокировок данной таблицы.
  – Пакет был связан с уровнем изоляции Чтение непринятого и выполняет оператор DELETE или UPDATE; уровень изоляции для фактического удаления повышен до стабильности на уровне указателя.

- Следующий оператор указывает, что при достаточном объеме памяти кучи сортировки некоторые или все строки, которые читаются из временной таблицы, будут кэшироваться вне пула буферов:

  ```
  Keep Rows In Private Memory
  ```

- Если для таблицы задан атрибут резкого изменения объема, это указывается оператором:

  ```
  Volatile Cardinality
  ```

Временные таблицы

Промежуточные или временные рабочие таблицы используются планом доступа для хранения в них данных при выполнении. Эти таблицы существуют, только пока выполняется план доступа. Как правило, временные таблицы используются, когда в плане доступа требуется предварительная оценка подзапросов, или когда для промежуточных результатов не хватает доступной памяти.

Если необходимо создание промежуточной таблицы, может появиться один из двух возможных операторов. Эти операторы указывают, что следует создать временную таблицу и вставить в нее строки. ID - это идентификатор, назначаемый `db2expln` для удобства обращения к данной временной таблице. Этот ID снабжен префиксом 't', который указывает, что таблица временная.

- Следующий оператор указывает, что будет создана обычная временная таблица:

  ```
  Insert Into Temp Table  ID = tn
  ```

- Следующий оператор указывает, что обычная временная таблица будет создана несколькими подагентами параллельно:

  ```
  Insert Into Shared Temp Table  ID = tn
  ```

- Следующий оператор указывает, что будет создана сортированная временная таблица:

  ```
  Insert Into Sorted Temp Table  ID = tn
  ```

- Следующий оператор указывает, что сортированная временная таблица будет создана несколькими подагентами параллельно:

  ```
  Insert Into Sorted Shared Temp Table  ID = tn
  ```

- Следующий оператор указывает, что будет создана объявленная глобальная временная таблица:

  ```
  Insert Into Global Temp Table  ID = ts,tn
  ```

- Следующий оператор указывает, что объявленная глобальная временная таблица будет создана несколькими подагентами параллельно:

  ```
  Insert Into Shared Global Temp Table  ID = ts,tn
  ```

- Следующий оператор указывает, что будет создана сортированная объявленная глобальная временная таблица:

  ```
  Insert Into Sorted Global Temp Table  ID = ts,tn
  ```

- Следующий оператор указывает, что отсортированная объявленная глобальная временная таблица будет создана несколькими подагентами параллельно:

  ```
  Insert Into Sorted Shared Global Temp Table  ID = ts,tn
  ```

Каждый из вышеприведенных операторов сопровождается записью:

```
#Columns = n
```

которая указывает, сколько столбцов в каждой строке вставляются в данную временную таблицу.

Сортированные временные таблицы

Сортированные временные таблицы могут быть получены в результате следующих операций:

- ORDER BY
- DISTINCT
- GROUP BY
- Объединение слиянием
- Подзапрос '= ANY'
- Подзапрос '<> ALL'
- INTERSECT или EXCEPT
- UNION (без ключевого слова ALL)

За исходным оператором создания сортированной временной таблицы может следовать ряд дополнительных операторов:

- Следующий оператор указывает число столбцов ключей, используемых в сортировке:

  ```
  #Sort Key Columns = n
  ```

  Для каждого столбца в ключе сортировки, будет выведена одна из следующих строк:

  ```
  Key n: имя_столбца (Ascending)
  Key n: имя_столбца (Descending)
  Key n: (Ascending)
  Key n: (Descending)
  ```

- Следующие операторы дают оценку числа строк и оценку размера строки, что позволяет выделить оптимальную кучу сортировки во время выполнения.

  ```
  Sortheap Allocation Parameters:
  |  #Rows     = n
  |  Row Width = n
  ```

- Если необходимы только первые строки отсортированного результата, выводится:

  ```
  Sort Limited To Estimated Row Count
  ```

- Для сортировки в симметричной мультипроцессорной среде (SMP) тип выполняемой сортировки выводится одним из следующих операторов:

  ```
  Use Partitioned Sort
  Use Shared Sort
  Use Replicated Sort
  Use Round-Robin Sort
  ```

  Описание различных методов сортировки смотрите в разделе “Стратегии параллельной сортировки” на стр. 205.

- Следующий оператор указывает, будет оставлен результат сортировки в куче сортировки или нет:

  ```
      Piped
  ```

  и

  ```
  Not Piped
  ```

  Если указывается конвейерная сортировка (Piped), менеджер баз данных будет хранить вывод сортировки в памяти, не помещая результат сортировки в другую временную таблицу. (Сравнительное описание конвейерных и неконвейерных сортировок смотрите в разделе “Влияние сортировки на оптимизатор” на стр. 201.)

- Следующий оператор указывает, что при сортировке будут удаляться повторяющиеся значения:

  ```
  Duplicate Elimination
  ```

- Если в сортировке выполняется суммирование, это будет указано одним из следующих операторов:

```
   Partial Aggregation
Intermediate Aggregation
Buffered Partial Aggregation
Buffered Intermediate Aggregation
```

Завершение временной таблицы

После обращения к таблице, которое влечет стековую операцию по созданию временной таблицы (то есть при доступе к таблице производится создание временной таблицы), будет следовать оператор завершения (completion), который обрабатывает конец файла - завершение создания временной таблицы, которое обеспечивает последующий доступ к ее строкам. Будет выведена одна из следующих строк:

```
Temp Table Completion  ID = tn
Shared Temp Table Completion  ID = tn
Sorted Temp Table Completion  ID = tn
Sorted Shared Temp Table Completion  ID = tn
```

Табличные функции

Табличные функции - это пользовательские функции, возвращающие оператору данные в форме таблицы. Дополнительную информацию о табличных функциях смотрите в справочнике *SQL Reference*. Табличные функции отмечаются оператором:

```
Access User Defined Table Function
|   Name = схема.имяфункции
|   Language = xxxx
|   Fenced   Deterministic   NULL Call   Disallow Parallel
```

Вместе с атрибутами табличной функции приводит язык, на котором она написана (C, OLE или Java).

Объединения

Существует три типа объединений (их описание смотрите в разделе “Концепции объединения” на стр. 182):

- Хеш-объединение (Hash join)
- Объединение слиянием (Merge join)
- Объединение со вложенным циклом (Nested loop join).

В момент запуска раздела, где выполняется объединение, выводится один из следующих операторов:

```
Hash Join
```

или

```
Merge Join
```

или

```
Nested Loop Join
```

Может выполняться левое внешнее объединение. На левое внешнее объединение указывает одним из следующих операторов:

```
Left Outer Hash Join
```

или

```
Left Outer Merge Join
```

или

```
Left Outer Nested Loop Join
```

Для объединения слиянием и объединения со вложенным циклом внешней таблицей объединения будет таблица, на которую есть ссылка в предыдущем операторе доступа, показанном в выводе. Внутренней таблицей объединения будет таблица, на которую есть ссылка в операторе доступа, находящемся в области действия данного оператора объединения. Для хеш-объединений операторы доступа, напротив, будут указывать внешнюю таблицу в области действия объединения и внутреннюю таблицу перед данным объединением.

Для хеш-объединения или объединения слияния могут быть выведены следующие дополнительные операторы:

- В некоторых случаях для объединения просто нужно определить, совпадает или нет любая строка во внутренней таблице с текущей строкой внешней таблицы. Это указывается следующим оператором:

  ```
  Early Out: Single Match Per Outer Row
  ```

- После выполнения объединения возможно применение предикатов. Число применяемых предикатов указывается так:

  ```
  Residual Predicate(s)
     |  #Predicates = n
  ```

Для хеш-объединения могут быть выведены следующие дополнительные операторы:

- Хеш-таблица строится из внутренней таблицы. Если построение хеш-таблицы было вынесено в предикат обращения к внутренней таблице, это указывается в обращении к внутренней таблице следующим оператором:

  ```
  Process Hash Table For Join
  ```

- При обращении к внешней таблице для улучшения производительности объединения может быть построена проверочная таблица. Построение проверочной таблицы указывается в обращении к внешней таблице следующим оператором:

  ```
  Process Probe Table For Hash Join
  ```

- Оценка памяти в байтах, необходимой для построения хеш-таблицы, дается следующим параметром:

  ```
  Estimated Build Size: n
  ```

- Оценка памяти в байтах, необходимой для проверочной таблицы, дается следующим параметром:

  ```
  Estimated Probe Size: n
  ```

Для объединения с вложенным циклом непосредственно после оператора объединения может быть выведен следующий дополнительный оператор:

```
Piped Inner
```

Этот оператор указывает, что внутренняя таблица объединения является результатом другой серии операций. Такая таблица называется также *составной внутренней* таблицей.

Если в объединении участвует больше двух таблиц, шаги объяснения должны читаться сверху вниз. Допустим, например, что вывод объяснения имеет следующий вид:

```
Access ..... W
Join
|  Access ..... X
Join
|  Access ..... Y
Join
|  Access ..... Z
```

Эти шаги:

1. Берут нужную строку из W.
2. Объединяют строку из W со следующей строкой из X и называют результат P1 (первый результат частичного объединения).
3. Объединяют P1 со следующей строкой из Y и создают P2.
4. Объединяют P2 со следующей строкой из Z и получают окончательный результат, выраженный одной строкой.
5. Если в Z есть еще строки, возврат к шагу 4.
6. Если в Y есть еще строки, возврат к шагу 3.
7. Если в X есть еще строки, возврат к шагу 2.
8. Если в W есть еще строки, возврат к шагу 1.

Потоки данных

В пределах плана доступа часто требуется управлять созданием данных и перемещением данных из одной серии операций к другой. Понятие потока данных позволяет управлять группой операций в плане доступа как модулем. Начало потока данных указывается следующим оператором:

```
Data Stream n
```

где n - уникальный идентификатор, назначаемый db2expln для удобства ссылки на поток данных. На окончание потока данных указывает оператор:

```
End of Data Stream n
```

Все операции между этими операторами рассматриваются как часть одного и того же потока данных.

Поток данных обладает рядом характеристик, и за начальным оператором потока данных может следовать один или несколько операторов описания этих характеристики:

- Если операция потока данных определяется значением, сгенерированным раньше в плане доступа, поток данных помечается как:

  ```
  Correlated
  ```

- Как и в случае сортированной временной таблицы, следующие операторы указывают, будет или нет поток данных храниться в памяти:

  ```
  Piped
  ```

  и

  ```
  Not Piped
  ```

  Как и в случае временных таблиц, если во время выполнения не хватает памяти, конвейерный (piped) поток данных может быть записан на диск. План доступа обеспечивает обе эти возможности.

- Следующий оператор указывает, что для данного потока данных требуется только одна запись:

  ```
  Single Record
  ```

При обращении к потоку данных вывод будет содержать следующий оператор:

```
Access Data Stream n
```

Операторы Insert, Update и Delete

Смысл этих операторов SQL понятен из их текста объяснения. Вот возможные варианты текста этих операторов SQL:

- `Insert: Table Name = схема.имя ID = ts,n`
- `Update: Table Name = схема.имя ID = ts,n`
- `Delete: Table Name = схема.имя ID = ts,n`
- `Insert: Hierarchy Table Name = схема.имя ID = ts,n`
- `Update: Hierarchy Table Name = схема.имя ID = ts,n`
- `Delete: Hierarchy Table Name = схема.имя ID = ts,n`
- `Insert: Summary Table Name = схема.имя ID = ts,n`
- `Update: Summary Table Name = схема.имя ID = ts,n`
- `Delete: Summary Table Name = схема.имя ID = ts,n`
- `Insert: Global Temporary Table ID = ts, tn`
- `Update: Global Temporary Table ID = ts, tn`

- `Delete: Global Temporary Table ID = ts, tn`

Подготовка идентификатора строки (RID)

Для некоторых планов доступа эффективность можно повысить, отсортировав идентификаторы нужных строк (RID) с удалением дубликатов (операция OR над индексами); этот же прием используется для идентификации RID, встречающихся во всех индексах, к которым выполняется обращение (операция AND над индексами) перед фактическим обращением к таблице. Есть три основных случая использования подготовки RID, что указывается операторами объяснения:

- Следующий оператор указывает, что для подготовки списка нужных RID используется “операция OR над индексами” :

  ```
  Index ORing RID Preparation
  ```

  Под *операцией OR над индексами* понимается выполнение нескольких обращений к индексам и объединение полученных различных RID, появляющихся хотя бы в одном из этих индексов. Оптимизатор рассматривает возможность логической операции OR над индексами, если предикаты соединяются ключевыми словами OR или если имеется предикат IN. Обращения могут выполняться как к одному, так и к разным индексам.
- Другое применение подготовки RID - это подготовка входных данных, используемых во время предварительного чтения списка, что указывается следующим оператором:

  ```
  List Prefetch RID Preparation
  ```
- Под *операцией AND над индексами* понимается выполнение нескольких обращений к индексам и составление результата из тех RID, которые появляются во всех этих индексах. Обработка логической операции AND для индексов начинается с оператора:

  ```
  Index ANDing
  ```

  Если оптимизатор оценил размер набора результатов, эта оценка будет показана в следующем операторе:

  ```
  Optimizer Estimate of Set Size: n
  ```

  В обработке RID используются операции фильтрации AND над индексами и битовый фильтр для определения RID, которые появляются в каждом обращении к индексу. Обработка RID с использованием операции AND над индексами указывается следующими операторами:

  ```
  Index ANDing Bitmap Build
  Index ANDing Bitmap Probe
  Index ANDing Bitmap Build and Probe
  ```

  Если оптимизатор оценил размер набора результатов для битового образа, эта оценка будет показана в следующем операторе:

  ```
  Optimizer Estimate of Set Size: n
  ```

Если для любого типа подготовки RID может быть выполнено предварительное чтение списка, на это указывает следующий оператор:

```
Prefetch: Enabled
```

Суммирование

Суммирование выполняется для строк, отвечающих заданным критериям (если они есть) из предикатов оператора SQL. Если должна быть выполнена функции суммирования какого либо рода, появляется один из следующих операторов:

```
Aggregation
Predicate Aggregation
Partial Aggregation
Partial Predicate Aggregation
Intermediate Aggregation
Intermediate Predicate Aggregation
Final Aggregation
Final Predicate Aggregation
```

Оператор Predicate aggregation сообщает, что операция суммирования была помещена в стек для обработки в качестве предиката при фактическом обращении к данным.

Под любым из вышеприведенных операторов суммирования будет следовать указание типа выполняемой функции суммирования:

- `Group By`
- `Column Function(s)`
- `Single Record`.

Конкретную функцию столбца (Column Function) можно посмотреть в исходном операторе SQL. Одиночная запись (Single Record) выбирается из индекса для операции MIN или MAX.

Если используется Predicate aggregation, за оператором обращения к таблице, для которой выполняется суммирование, будет следовать строка о завершении суммирования, соответствующая всей необходимой обработке до конца каждой группы или до конца файла. Выводится одна из следующих строк:

```
Aggregation Completion
Partial Aggregation Completion
Intermediate Aggregation Completion
Final Aggregation Completion
```

Параллельная обработка

Для параллельного выполнения операторов SQL (с использованием внутриразделього или межраздельного параллелизма) требуются некоторые специальные операции. Операции для параллельных планов описаны ниже.

- При запуске внутриразделього параллельного плана части плана будут выполняться одновременно, с использованием нескольких подагентов. О создании подагентов сообщает оператор:

```
Process Using n Subagents
```

- При запуске межраздельного параллельного плана раздел плана разбивается на несколько подразделов. Для запуска каждый подраздел посылается на один или несколько узлов. Важным подразделом является *подраздел координатора*. Подраздел координатора - это первый подраздел в каждом плане. Он первым принимает управление и отвечает за распределение других подразделов и возвращение результатов в вызывающую программу.

  О распределении подраздела сообщает оператор:

  ```
  Distribute Subsection #n
  ```

  Узел, получающие подраздел для выполнения, можно определить одним из восьми способов:

  – Следующая запись указывает, что подраздел будет послан на узел в группе узлов на основании значения столбцов.

    ```
    Directed by Hash
    |   #Columns = n
    |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
    ```

  – Следующая запись указывает, что подраздел будет послан на узел, определенный заранее. (Это часто происходит, когда в операторе используется функция NODENUMBER().)

    ```
    Directed by Node Number
    ```

  – Следующая запись указывает, что подраздел будет послан на узел в соответствии с ранее определенным номером подраздела в данной группе узлов. (Это часто происходит, когда в операторе используется функция PARTITION().)

    ```
    Directed by Partition Number
    |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
    ```

  – Следующая запись указывает, что подраздел будет послан на узел, который предоставил текущую строку для указателя программы.

    ```
    Directed by Position
    ```

  – Следующая запись указывает, что данный подраздел получит только один узел, определенный при компиляции оператора.

    ```
    Directed to Single Node
    |  Node Number = n
    ```

  – Следующая запись указывает, что подраздел будет выполняться на узле координатора.

    ```
    Directed to Coordinator Node
    ```

  – Следующая запись указывает, что подраздел будет послан на все перечисленные узлы.

    ```
    Broadcast to Node List
    |  Nodes = n1, n2, n3, ...
    ```

  – Следующая запись указывает, что данный подраздел получит только один узел, определенный при выполнении оператора.

```
Directed to Any Node
```

- Для перемещения данных между подразделами в среде многораздельной базы данных или между подагентами в симметричной мультипроцессорной среде (SMP) используются очереди таблиц. Очереди таблиц описываются следующим образом:
  - Следующие операторы указывают, что данные помещаются в очередь таблиц:

    ```
    Insert Into Synchronous Table Queue  ID = qn
    Insert Into Asynchronous Table Queue  ID = qn
    Insert Into Synchronous Local Table Queue  ID = qn
    Insert Into Asynchronous Local Table Queue  ID = qn
    ```

  - Для очередей таблиц раздела базы данных назначение для строк, вставленных в очередь таблиц, описывается одной из следующих записей:

    ```
    Broadcast to Coordinator Node
    ```

    Все строки посылаются на узел координатора.

    ```
    Broadcast to All Nodes of Subsection n
    ```

    Все строки посылаются в каждый раздел базы данных, где запускается данный подраздел.

    ```
    Hash to Specific Node
    ```

    Каждая строка посылается в раздел базы данных на основании значений в данной строке.

    ```
    Send to Specific Node
    ```

    Каждая строка посылается в раздел базы данных, определенный во время выполнения данного оператора.

    ```
    Send to Random Node
    ```

    Каждая строка посылается в случайный раздел базы данных.
  - В некоторых ситуациях очередь таблиц раздела базы данных будет на какое-то время переполняться, и некоторые строки будут направляться во временную таблицу. Это описывается следующим оператором:

    ```
    Rows Can Overflow to Temporary Table
    ```

  - После обращения к таблице, которая содержит стековую операцию по вставке строк в очередь таблиц, будет следовать оператор завершения, обрабатывающий строки, которые не могут быть отосланы немедленно. Выводится одна из следующих строк:

    ```
    Insert Into Synchronous Table Queue Completion  ID = qn
    Insert Into Asynchronous Table Queue Completion  ID = qn
    Insert Into Synchronous Local Table Queue Completion  ID = qn
    Insert Into Asynchronous Local Table Queue Completion  ID = qn
    ```

  - Получение данных из очереди таблиц отмечается следующим оператором:

```
Access Table Queue  ID = qn
Access Local Table Queue  ID = qn
```

За этими сообщениями всегда следует указание числа получаемых столбцов.

```
#Columns = n
```

– Если очередь таблиц сортирует строки на месте приема, обращение к этой очереди таблиц также будет сопровождаться одним из следующих сообщений:

```
Output Sorted
Output Sorted and Unique
```

За этими сообщениями следует указание числа ключей, используемых для сортировки.

```
#Key Columns = n
```

Для каждого столбца в ключе сортировки выводится одна из следующих записей:

```
Key n: (Ascending)
Key n: (Descending)
```

– Если предикаты будут применены к строкам на месте приема очереди таблиц, выводится следующее сообщение:

```
Residual Predicate(s)
|  #Predicates = n
```

- Некоторые подразделы в среде многораздельных баз данных могут содержать явные циклы, на что указывает оператор:

```
Jump Back to Start of Subsection
```

Обработка оператора объединения

Для выполнения оператора SQL в базе данных объединения требуется выполнять части этого оператора на других источниках данных.

Следующая запись показывает, что будет выполняться обращение к источнику данных:

```
Distributed Subquery #n
|   #Columns = n
```

К данным, полученным в результате распределенного подзапроса, можно применить предикаты. Число применяемых предикатов указывается так:

```
Residual Predicate(s)
   |  #Predicates = n
```

Подробные записи для каждого распределенного подзапроса приводятся отдельно. Опции распределенных подзапросов описаны ниже:

- Источник данных для распределенного подзапроса указывается одной из следующих записей:

  ```
  Server: имя_сервера (тип, версия)
  Server: имя_сервера (тип)
  Server: имя_сервера
  ```

- Оператор SQL для распределенного подзапроса выводится следующим образом:

  ```
  Subquery SQL Statement:
оператор
  ```

- Псевдонимы, ссылки на которые есть в подзапросе, выводятся так:

  ```
  Nickname Referenced:
  схема.псевдоним Base = схемабазы.таблицабазы
  ```

- Если от сервера объединения на источник данных перед выполнением подзапроса передаются значения, их число указывается так:

  ```
  #Input Columns: n
  ```

- Если значения передаются из источника данных на сервер объединения после выполнения подзапроса, их число будет показано так:

  ```
  #Output Columns: n
  ```

Различные операторы

- Разделы для операторов языка определения данных (DDL) будут показаны в выводе так:

  ```
  DDL Statement
  ```

  Дополнительный вывод объяснения для операторов DDL отсутствует.

- Разделы операторов SET для изменяемых специальных регистров, например **CURRENT EXPLAIN SNAPSHOT**, будут указаны так:

  ```
  SET Statement
  ```

  Дополнительный вывод объяснения для операторов SET отсутствует.

- Если оператор SQL содержит условие DISTINCT, в выводе может появиться следующий текст:

  ```
  Distinct Filter  #Columns = n
  ```

  где `n` - число столбцов в таблице, участвующих в получении различных строк. Для получении значений различных строк эти строки должны быть упорядочены, чтобы можно было отфильтровать повторения. Этот оператор не появится, если менеджер баз данных не должен явно отфильтровывать повторения, например, в следующих случаях:

  – Существует индекс уникальности, и в операции DISTINCT принимают участие все столбцы этого индекса
  – Повторения можно устранить во время сортировки.

- Если следующая операция зависит от конкретного идентификатора записи, выводится оператор:

  ```
  Positioned Operation
  ```

  Этот оператор появляется для любого оператора SQL, где используется условие WHERE CURRENT OF.

- Если какие-то предикаты нужно применить к результату, и их нельзя применить как часть другой операции, выводится следующий оператор:

  ```
  Residual Predicate Application
  |  #Predicates = n
  ```

- Следующий оператор появится, если в операторе SQL присутствует операция UNION:

  ```
  UNION
  ```

- Если в плане доступа есть операция, единственной целью которой является генерация значений строк для их использования в последующих операциях, появится следующий оператор:

  ```
  Table Constructor
  |  n-Row(s)
  ```

  Конструкторы таблиц могут использоваться для преобразования значений из набора в ряд строк, которые затем передаются последующим операциям. Если от конструктора таблиц требуется следующая строка, выводится оператор:

  ```
  Access Table Constructor
  ```

- Если существует операция, которая выполняется только при определенных условиях, выводится следующий оператор:

  ```
  Conditional Evaluation
  |  Condition #n:
  |  |  #Predicates = n
  |  Action #n:
  ```

  Оценка условий используется, например, для реализации оператора SQL CASE или таких внутренних механизмов, как реляционные ограничения или триггеры. Если никаких действий не указано, операции работы с данными выполняются, только когда условие истинно.

- Если в плане доступа обрабатывается подзапрос ALL, ANY или EXISTS, выводится один из следующих операторов:
  - `ANY/ALL Subquery`
  - `EXISTS Subquery`
  - `EXISTS SINGLE Subquery`

- Перед некоторыми операциями UPDATE и DELETE необходимо установить позицию определенной строки в таблице. На это указывает следующий оператор:

```
Establish Row Position
```

- Если в программу возвращаются какие-либо строки, появится следующий оператор:

```
Return Data to Application
  #Columns = n
```

Если данная операция была внесена в операцию обращения к таблице, для нее требуется фаза завершения. Эта фаза будет отмечена так:

```
Return Data Completion
```

Примеры вывода db2expln и dynexpln

Чтобы лучше понять структуру и формат вывода db2expln и dynexpln, посмотрите пять приведенных здесь примеров. Эти примеры получены на базе данных SAMPLE, поставляемой с DB2. К каждому примеру приводится краткое описание. Важные различия между примерами выделены **жирным** шрифтом.

Пример первый: План без параллелизма

Этот пример - простой запрос списка всех сотрудников, их заданий, названий и местонахождений отделов, а также названий проектов, над которыми они работают. Суть этого плана доступа состоит в том, что для объединения нужных данных из каждой заданной таблицы используются объединения слияния. Так как индексов для этих данных нет, план доступа выполняет реляционный просмотр каждой таблицы, и перед тем, как таблицу можно будет использовать для объединения, ее нужно отсортировать.

```
******************** PACKAGE ****************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:47:58

Bind Timestamp = 2000-01-03-15.47.58.607455

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5

Partition Parallel       = No
Intra-Partition Parallel = No

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

---------- SECTION ----------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
```

```
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno


Estimated Cost        = 126
Estimated Cardinality = 153

Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
|  #Columns = 3
|  Relation Scan
|  |  Prefetch: Eligible
|  Lock Intents
|  |  Table: Intent Share
|  |  Row  : Next Key Share
|  Insert Into Sorted Temp Table  ID = t1
|  |  #Columns = 3
|  |  #Sort Key Columns = 1
|  |  |  Key 1: DEPTNO (Ascending)
|  |  Sortheap Allocation Parameters:
|  |  |  #Rows     = 40
|  |  |  Row Width = 48
|  |  Piped
Sorted Temp Table Completion  ID = t1
   Access Temp Table  ID = t1
|  #Columns = 3
|  Relation Scan
|  |  Prefetch: Eligible
Merge Join
|  Access Table Name = DOOLE.PROJECT  ID = 2,7
|  |  #Columns = 2
|  |  Relation Scan
|  |  |  Prefetch: Eligible
|  |  Lock Intents
|  |  |  Table: Intent Share
|  |  |  Row  : Next Key Share
|  |  |  Insert Into Sorted Temp Table  ID = t2
|  |  |  #Columns = 2
|  |  |  #Sort Key Columns = 1
|  |  |  |  Key 1: DEPTNO (Ascending)
|  |  |  Sortheap Allocation Parameters:
|  |  |  |  #Rows     = 38
|  |  |  |  Row Width = 28
|  |  |  Piped
|  |  Sorted Temp Table Completion  ID = t2
|  Access Temp Table  ID = t2
|  |  #Columns = 2
|  |  Relation Scan
|  |  |  Prefetch: Eligible
Merge Join
|  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
|  |  #Columns = 3
|  |  Relation Scan
|  |  |  Prefetch: Eligible
|  |  Lock Intents
|  |  |  Table: Intent Share
```

```
|  |  |  Row  : Next Key Share
|  |  Insert Into Sorted Temp Table  ID = t3
|  |  |  #Columns = 3
|  |  |  #Sort Key Columns = 1
|  |  |  |  Key 1: WORKDEPT (Ascending)
|  |  |  Sortheap Allocation Parameters:
|  |  |  |  #Rows     = 63
|  |  |  |  Row Width = 32
|  |  |  Piped
|  Sorted Temp Table Completion  ID = t3
   |  Access Temp Table  ID = t3
|  |  #Columns = 3
|  |  Relation Scan
|  |  |  Prefetch: Eligible
   Return Data to Application
|  #Columns = 5

End of section


Optimizer Plan:

RETURN
                (   1)
                  |
MSJOIN
                (   2)
               /      \
       MSJOIN            TBSCAN
        (   3)          (  12)
       /      \           |
  TBSCAN      TBSCAN     SORT
   (   4)     (   8)     (  13)
     |          |          |
    SORT       SORT      TBSCAN
   (   5)     (   9)     (  14)
     |          |          |
   TBSCAN     TBSCAN  Table:
   (   6)     (  10)   DOOLE
     |          |     EMPLOYEE
 Table:      Table:
 DOOLE       DOOLE
 DEPARTMENT  PROJECT
```

В первой части плана выполняется обращение к таблицам DEPARTMENT и PROJECT, а для их объединения используется объединение слияния. Результат этого объединения помещается в таблицу EMPLOYEE. Строки результата возвращаются программе.

Пример второй: План одноразделной базы данных с внутриразделным параллелизмом

Здесь показан тот же оператор SQL, что и в первом примере, но запрос скомпилирован для компьютера с четырьмя параллельными процессорами.

```
******************** PACKAGE ****************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:48:51

Bind Timestamp = 2000-01-03-15.48.51.402403

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5

Partition Parallel       = No
Intra-Partition Parallel = Yes (Bind Degree = 4)

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

---------- SECTION ---------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 142
Estimated Cardinality = 153

Process Using 4 Subagents
    |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
|   |  #Columns = 3
|   |  Parallel Scan
|   |  Relation Scan
|   |  |  Prefetch: Eligible
|   |  Lock Intents
|   |  |  Table: Intent Share
|   |  |  Row  : Next Key Share
|   |  Insert Into Sorted Shared Temp Table  ID = t1
|   |  |  #Columns = 3
|   |  |  #Sort Key Columns = 1
|   |  |  |  Key 1: DEPTNO (Ascending)
|   |  |  |  Use Round-Robin Sort
|   |  |  Sortheap Allocation Parameters:
|   |  |  |  |  #Rows     = 40
|   |  |  |  |  Row Width = 48
|   |  |  Piped
|  Sorted Shared Temp Table Completion  ID = t1
    |  Access Temp Table  ID = t1
|   |  #Columns = 3
|   |  Relation Scan
```

```
    |  |  |  Prefetch: Eligible
    |  |  Merge Join
    |  |  Access Table Name = DOOLE.PROJECT  ID = 2,7
    |  |  |  #Columns = 2
    |  |  |  Parallel Scan
    |  |  |  |  Relation Scan
    |  |  |  |  |  Prefetch: Eligible
    |  |  |  Lock Intents
    |  |  |  |  |  Table: Intent Share
    |  |  |  |  |  Row  : Next Key Share
    |  |  |  Insert Into Sorted Shared Temp Table  ID = t2
    |  |  |  #Columns = 2
    |  |  |  |  #Sort Key Columns = 1
    |  |  |  |  |  Key 1: DEPTNO (Ascending)
    |  |  |  Use Replicated Sort
    |  |  |  |  Sortheap Allocation Parameters:
    |  |  |  |  #Rows     = 38
    |  |  |  |  Row Width = 28
    |  |  |  |  Piped
    |  |  Sorted Shared Temp Table Completion  ID = t2
    |  |  Access Temp Table  ID = t2
    |  |  #Columns = 2
    |  |  |  Relation Scan
    |  |  |  |  Prefetch: Eligible
    |  Insert Into Sorted Shared Temp Table  ID = t3
    |  |  #Columns = 5
    |  |  #Sort Key Columns = 1
    |  |  |  Key 1: (Ascending)
    |  |  Use Partitioned Sort
    |  |  Sortheap Allocation Parameters:
    |  |  |  #Rows     = 61
    |  |  |  Row Width = 72
    |  |  Piped
    |  |  Access Temp Table  ID = t3
    |  |  #Columns = 5
    |  |  Relation Scan
    |  |  |  |  Prefetch: Eligible
    |  |  Merge Join
    |  |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
    |  |  |  #Columns = 3
    |  |  |  Parallel Scan
    |  |  |  |  Relation Scan
    |  |  |  |  |  Prefetch: Eligible
    |  |  |  Lock Intents
    |  |  |  |  |  Table: Intent Share
    |  |  |  |  |  Row  : Next Key Share
    |  |  |  Insert Into Sorted Shared Temp Table  ID = t4
    |  |  |  |  |  #Columns = 3
    |  |  |  |  |  #Sort Key Columns = 1
    |  |  |  |  |  Key 1: WORKDEPT (Ascending)
    |  |  |  |  |  Use Partitioned Sort
    |  |  |  |  |  Sortheap Allocation Parameters:
    |  |  |  |  |  #Rows     = 63
    |  |  |  |  |  Row Width = 32
    |  |  |  |  |  Piped
```

```
|  |  Sorted Shared Temp Table Completion  ID = t4
   |  |  Access Temp Table  ID = t4
|  |  |  #Columns = 3
   |  |  |  Relation Scan
   |  |  |  |  Prefetch: Eligible
|  Insert Into Asynchronous Local Table Queue  ID = q1
Access Local Table Queue  ID = q1  #Columns = 5
   Return Data to Application
|  #Columns = 5

End of section


Optimizer Plan:

RETURN
                (   1)
                  |
                 LTQ
                (   2)
                  |
MSJOIN
                (   3)
               /      \
       TBSCAN          TBSCAN
        (   4)         (  15)
          |              |
        SORT            SORT
        (   5)         (  16)
          |              |
       MSJOIN            TBSCAN
        (   6)         (  17)
       /      \          |
   TBSCAN    TBSCAN  Table:
   (   7)    (  11)  DOOLE
     |         |     EMPLOYEE
    SORT      SORT
   (   8)    (  12)
     |         |
   TBSCAN    TBSCAN
   (   9)    (  13)
     |         |
 Table:      Table:
 DOOLE       DOOLE
 DEPARTMENT  PROJECT
```

Этот план почти идентичен плану в первом примере. Основные различия - это создание четырех подагентов в начале плана и очередь таблиц для сбора результатов работы каждого подагента перед их возвратом в программу в конце плана.

Интересно также отметить, что перед объединением с таблицей EMPLOYEE требуется дополнительная сортировка. Это необходимо, так как подагенты,

обрабатывающие объединение слиянием таблиц DEPARTMENT и PROJECT, могут генерировать строки объединения не по порядку.

Пример третий: План многораздельной базы данных с межраздельным параллелизмом

Здесь показан тот же оператор SQL, что и в первом примере, но данный запрос был скомпилирован для многораздельной базы данных с тремя разделами.

```
******************** PACKAGE ************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:21:29

Bind Timestamp = 2000-01-03-15.21.29.990983

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5

Partition Parallel       = Yes
Intra-Partition Parallel = No

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno


Estimated Cost        = 118
Estimated Cardinality = 263

Coordinator Subsection:
   Distribute Subsection #2
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #3
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #1
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Access Table Queue  ID = q1  #Columns = 5
   Return Data to Application
   |  #Columns = 5

Subsection #1:
```

```
   Access Table Queue  ID = q2  #Columns = 3
   |  Output Sorted
   |  |  #Key Columns = 1
   |  |  |  Key 1: (Ascending)
   Merge Join
   |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
   |  |  #Columns = 3
   |  |  Relation Scan
   |  |  |  Prefetch: Eligible
   |  |  Lock Intents
   |  |  |  Table: Intent Share
   |  |  |  Row  : Next Key Share
   |  |  Insert Into Sorted Temp Table  ID = t1
   |  |  |  #Columns = 3
   |  |  |  #Sort Key Columns = 1
   |  |  |  |  Key 1: DEPTNO (Ascending)
   |  |  |  Sortheap Allocation Parameters:
   |  |  |  |  #Rows     = 40
   |  |  |  |  Row Width = 48
   |  |  |  Piped
   |  Sorted Temp Table Completion  ID = t1
   |  Access Temp Table  ID = t1
   |  |  #Columns = 3
   |  |  Relation Scan
   |  |  |  Prefetch: Eligible
   Merge Join
   |  Access Table Queue  ID = q3  #Columns = 2
   |  |  Output Sorted
   |  |  |  #Key Columns = 1
   |  |  |  |  Key 1: (Ascending)
   Insert Into Asynchronous Table Queue  ID = q1
   |  Broadcast to Coordinator Node
   |  Rows Can Overflow to Temporary Table

Subsection #2:
   Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
   |  #Columns = 3
   |  Relation Scan
   |  |  Prefetch: Eligible
   |  Lock Intents
   |  |  Table: Intent Share
   |  |  Row  : Next Key Share
   |  Insert Into Sorted Temp Table  ID = t2
   |  |  #Columns = 3
   |  |  #Sort Key Columns = 1
   |  |  |  Key 1: WORKDEPT (Ascending)
   |  |  Sortheap Allocation Parameters:
   |  |  |  #Rows     = 27
   |  |  |  Row Width = 32
   |  |  Piped
   Sorted Temp Table Completion  ID = t2
   Access Temp Table  ID = t2
   |  #Columns = 3
   |  Relation Scan
   |  |  Prefetch: Eligible
```

```
    |  Insert Into Asynchronous Table Queue  ID = q2
    |  |  Hash to Specific Node
    |  |  Rows Can Overflow to Temporary Tables
   Insert Into Asynchronous Table Queue Completion  ID = q2

Subsection #3:
   Access Table Name = DOOLE.PROJECT  ID = 2,7
   |  #Columns = 2
   |  Relation Scan
   |  |  Prefetch: Eligible
   |  Lock Intents
   |  |  Table: Intent Share
   |  |  Row  : Next Key Share
   |  Insert Into Sorted Temp Table  ID = t3
   |  |  #Columns = 2
   |  |  #Sort Key Columns = 1
   |  |  |  Key 1: DEPTNO (Ascending)
   |  |  Sortheap Allocation Parameters:
   |  |  |  #Rows     = 38
   |  |  |  Row Width = 28
   |  |  Piped
   Sorted Temp Table Completion  ID = t3
   Access Temp Table  ID = t3
   |  #Columns = 2
   |  Relation Scan
   |  |  Prefetch: Eligible
   |  Insert Into Asynchronous Table Queue  ID = q3
   |  |  Hash to Specific Node
   |  |  Rows Can Overflow to Temporary Tables
   Insert Into Asynchronous Table Queue Completion  ID = q3

End of section


Optimizer Plan:

RETURN
                (   1)
                  |
                 BTQ
                (   2)
                  |
MSJOIN
                (   3)
               /      \
       MSJOIN           MDTQ
       (   4)          (  14)
      /      \           |
    MDTQ      TBSCAN    TBSCAN
   (   5)     (  10)    (  15)
     |          |         |
   TBSCAN      SORT      SORT
   (   6)     (  11)    (  16)
     |          |         |
    SORT       TBSCAN    TBSCAN
```

```
  (   7)     (  12)    (  17)
    |          |         |
  TBSCAN   Table:      Table:
  (   8)   DOOLE       DOOLE
    |      DEPARTMENT  PROJECT
 Table:
 DOOLE
 EMPLOYEE
```

Данный план содержит те же части, что и план первого примера, но его раздел разбит на четыре подраздела. Эти подразделы выполняют следующие задачи:

- **Подраздел координатора**. Этот подраздел координирует выполнение остальных разделов. Он выполняет распределение для других подразделов, а затем использует очередь таблиц для сбора результатов, которые будут возвращены в программу.
- **Подраздел 1**. Этот подраздел просматривает очередь таблиц q2 и использует объединение слиянием для объединения с таблицей DEPARTMENT. Затем используется второе объединение слиянием для добавления данных из очереди таблиц q3. После этого объединенные строки отсылаются в подраздел координатора с использованием очереди таблиц q1.
- **Подраздел 2**. Этот подраздел просматривает таблицу EMPLOYEE, сортирует ее и выполняет хеш-распределение на определенный узел с результатами. Эти результаты читает подраздел 1.
- **Подраздел 3**. Этот подраздел просматривает таблицу PROJECT, сортирует ее и выполняет хеш-распределение на определенный узел с результатами. Эти результаты читает подраздел 1.

Пример четвертый: План многораздельной базы данных с межраздельным и внутрираздельным параллелизмом

В этом примере показан тот же оператор SQL, что и в первом примере, но данный запрос скомпилирован для многораздельной базы данных с тремя разделами, каждый из которых находится на компьютере с четырьмя параллельными процессорами.

```
******************** PACKAGE ************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:22:14

Bind Timestamp = 2000-01-03-15.22.14.659970

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5

Partition Parallel       = Yes
Intra-Partition Parallel = Yes (Bind Degree = 4)

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"
```

```
-------------------- SECTION --------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 140
Estimated Cardinality = 263

Coordinator Subsection:
   Distribute Subsection #2
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #3
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #1
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Access Table Queue  ID = q1  #Columns = 5
   Return Data to Application
   |  #Columns = 5

Subsection #1:
   Process Using 4 Subagents
   |  Access Table Queue  ID = q3  #Columns = 3
   |  Insert Into Sorted Shared Temp Table  ID = t1
   |  |  #Columns = 3
   |  |  #Sort Key Columns = 1
   |  |  |  Key 1: (Ascending)
   |  |  Use Partitioned Sort
   |  |  Sortheap Allocation Parameters:
   |  |  |  #Rows     = 27
   |  |  |  Row Width = 32
   |  |  Piped
   |  Access Temp Table  ID = t1
   |  |  #Columns = 3
   |  |  Relation Scan
   |  |  |  Prefetch: Eligible
   |  Merge Join
   |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
   |  |  |  #Columns = 3
   |  |  |  Parallel Scan
   |  |  |  Relation Scan
   |  |  |  |  Prefetch: Eligible
   |  |  |  Lock Intents
   |  |  |  |  Table: Intent Share
```

```
|  |  |  |  Row  : Next Key Share
|  |  |  Insert Into Sorted Shared Temp Table  ID = t2
|  |  |  |  #Columns = 3
|  |  |  |  #Sort Key Columns = 1
|  |  |  |  |  Key 1: DEPTNO (Ascending)
|  |  |  |  Use Partitioned Sort
|  |  |  |  Sortheap Allocation Parameters:
|  |  |  |  |  #Rows     = 40
|  |  |  |  |  Row Width = 48
|  |  |  |  Piped
|  |  Sorted Shared Temp Table Completion  ID = t2
|  |  Access Temp Table  ID = t2
|  |  |  #Columns = 3
|  |  |  Relation Scan
|  |  |  |  Prefetch: Eligible
|  Insert Into Sorted Shared Temp Table  ID = t3
|  |  #Columns = 6
|  |  #Sort Key Columns = 1
|  |  |  Key 1: (Ascending)
|  |  Use Partitioned Sort
|  |  Sortheap Allocation Parameters:
|  |  |  #Rows     = 44
|  |  |  Row Width = 76
|  |  Piped
|  Access Temp Table  ID = t3
|  |  #Columns = 6
|  |  Relation Scan
|  |  |  Prefetch: Eligible
|  Merge Join
|  |  Access Table Queue  ID = q5  #Columns = 2
|  |  Insert Into Sorted Shared Temp Table  ID = t4
|  |  |  #Columns = 2
|  |  |  #Sort Key Columns = 1
|  |  |  |  Key 1: (Ascending)
|  |  |  Use Partitioned Sort
|  |  |  Sortheap Allocation Parameters:
|  |  |  |  #Rows     = 38
|  |  |  |  Row Width = 28
|  |  |  Piped
|  |  Access Temp Table  ID = t4
|  |  |  #Columns = 2
|  |  |  Relation Scan
|  |  |  |  Prefetch: Eligible
|  Insert Into Asynchronous Local Table Queue  ID = q2
Access Local Table Queue  ID = q2  #Columns = 5
Insert Into Asynchronous Table Queue  ID = q1
|  Broadcast to Coordinator Node
|  Rows Can Overflow to Temporary Table

Subsection #2:
   Process Using 4 Subagents
   |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
   |  |  #Columns = 3
   |  |  Parallel Scan
   |  |  Relation Scan
```

```
      |  |  |  Prefetch: Eligible
      |  |  Lock Intents
      |  |  |  Table: Intent Share
      |  |  |  Row  : Next Key Share
      |  |  Insert Into Sorted Shared Temp Table  ID = t5
      |  |  |  #Columns = 3
      |  |  |  #Sort Key Columns = 1
      |  |  |  |  Key 1: WORKDEPT (Ascending)
      |  |  |  Use Round-Robin Sort
      |  |  |  Sortheap Allocation Parameters:
      |  |  |  |  #Rows     = 27
      |  |  |  |  Row Width = 32
      |  |  |  Piped
      |  Sorted Shared Temp Table Completion  ID = t5
      |  Access Temp Table  ID = t5
      |  |  #Columns = 3
      |  |  Relation Scan
      |  |  |  Prefetch: Eligible
      |  Insert Into Asynchronous Local Table Queue  ID = q4
      Access Local Table Queue  ID = q4  #Columns = 3
      Insert Into Asynchronous Table Queue  ID = q3
      |  Hash to Specific Node
      |  Rows Can Overflow to Temporary Tables

   Subsection #3:
      Process Using 4 Subagents
      |  Access Table Name = DOOLE.PROJECT  ID = 2,7
      |  |  #Columns = 2
      |  |  Parallel Scan
      |  |  Relation Scan
      |  |  |  Prefetch: Eligible
      |  |  Lock Intents
      |  |  |  Table: Intent Share
      |  |  |  Row  : Next Key Share
      |  |  Insert Into Sorted Shared Temp Table  ID = t6
      |  |  |  #Columns = 2
      |  |  |  #Sort Key Columns = 1
      |  |  |  |  Key 1: DEPTNO (Ascending)
      |  |  |  Use Round-Robin Sort
      |  |  |  Sortheap Allocation Parameters:
      |  |  |  |  #Rows     = 38
      |  |  |  |  Row Width = 28
      |  |  |  Piped
      |  Sorted Shared Temp Table Completion  ID = t6
      |  Access Temp Table  ID = t6
      |  |  #Columns = 2
      |  |  Relation Scan
      |  |  |  Prefetch: Eligible
      |  Insert Into Asynchronous Local Table Queue  ID = q6
      Access Local Table Queue  ID = q6  #Columns = 2
      Insert Into Asynchronous Table Queue  ID = q5
      |  Hash to Specific Node
      |  Rows Can Overflow to Temporary Tables

   End of section
```

```
Optimizer Plan:

RETURN
                (   1)
                  |
                 BTQ
                (   2)
                  |
                 LTQ
                (   3)
                  |
MSJOIN
                (   4)
               /      \
        TBSCAN          TBSCAN
        (   5)          (  20)
          |               |
         SORT            SORT
        (   6)          (  21)
          |               |
        MSJOIN           DTQ
        (   7)          (  22)
       /      \          |
  TBSCAN     TBSCAN     LTQ
  (   8)     (  16)     (  23)
    |          |          |
   SORT       SORT     TBSCAN
  (   9)     (  17)     (  24)
    |          |          |
   DTQ       TBSCAN     SORT
  (  10)     (  18)     (  25)
    |          |          |
   LTQ     Table:       TBSCAN
  (  11)   DOOLE        (  26)
    |      DEPARTMENT     |
  TBSCAN                Table:
  (  12)                DOOLE
    |                   PROJECT
SORT
  (  13)
    |
TBSCAN
  (  14)
    |
 Table:
 DOOLE
 EMPLOYEE
```

Этот план подобен плану третьего примера, кроме того, что каждый подраздел выполняется несколькими подагентами. Кроме того, в конце каждого

подраздела локальная очередь таблиц собирает результаты от всех подагентов перед тем, как нужные строки будут помещены во вторую очередь таблиц для хеширования на заданный узел.

Пример пятый: План базы данных объединения

Здесь показан тот же оператор SQL, что и в первом примере, но данный запрос скомпилирован для базы данных объединения, где таблицы DEPARTMENT и PROJECT находятся на источнике данных, а таблица EMPLOYEE - на сервере объединения.

```
******************** PACKAGE ************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 16:29:01

Bind Timestamp = 2000-01-03-16.29.01.479230

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5

Partition Parallel       = No
Intra-Partition Parallel = No

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

-------------------- SECTION -------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno

Estimated Cost        = 1954
Estimated Cardinality = 100800

Distribute Subquery #2
   |  #Columns = 3
   Insert Into Sorted Shared Temp Table  ID = t1
   |  #Columns = 3
   |  #Sort Key Columns = 1
   |  |  Key 1: Remote Query #2, Output Column 1 (Ascending)
   |  Sortheap Allocation Parameters:
   |  |  #Rows     = 1000
   |  |  Row Width = 56
   |  Piped
   Access Temp Table  ID = t1
   |  #Columns = 3
   |  Relation Scan
```

```
|  |  Prefetch: Eligible
Merge Join
|  Access Table Name = DOOLE.DEPARTMENT  ID = 2,5
|  #Columns = 3
|  Relation Scan
|  |  Prefetch: Eligible
|  |  Lock Intents
|  |  |  Table: Intent Share
|  |  |  Row  : Next Key Share
|  |  Insert Into Sorted Temp Table  ID = t2
|  |  |  #Columns = 3
|  |  |  #Sort Key Columns = 1
|  |  |  |  Key 1: WORKDEPT (Ascending)
|  |  |  Sortheap Allocation Parameters:
|  |  |  |  #Rows     = 63
|  |  |  |  Row Width = 32
|  |  |  Piped
|  Sorted Temp Table Completion  ID = t2
|  Access Temp Table  ID = t2
|  |  #Columns = 3
|  |  Relation Scan
|  |  |  Prefetch: Eligible
Merge Join
|  Distribute Subquery #1
|  |  #Columns = 2
|  Insert Into Sorted Temp Table  ID = t3
|  |  #Columns = 2
|  |  |  Key 1: Remote Query #1, Output Column 1 (Ascending)
|  |  Sortheap Allocation Parameters:
|  |  |  #Rows     = 1000
|  |  |  Row Width = 36
|  |  Piped
|  Access Temp Table  ID = t3
|  |  #Columns = 2
|  |  Relation Scan
|  |  |  Prefetch: Eligible
Return Data to Application
|  #Columns = 5

Distributed Subquery #1:
Server: REMOTE_SAMPLE  (DB2/CS 7.1)
Subquery SQL Statement:

     SELECT A0."DEPTNO", A0."PROJNAME"
     FROM "DOOLE"."PROJECT" A0

Nicknames Referenced:
   REMOTE.PROJECT  ID = 7  Base = DOOLE.PROJECT
#Output Columns = 2

Distributed Subquery #2:
Server: REMOTE_SAMPLE  (DB2/CS 7.1)
Subquery SQL Statement:

     SELECT A0."DEPTNO", A0."DEPTNAME", A0."LOCATION"
```

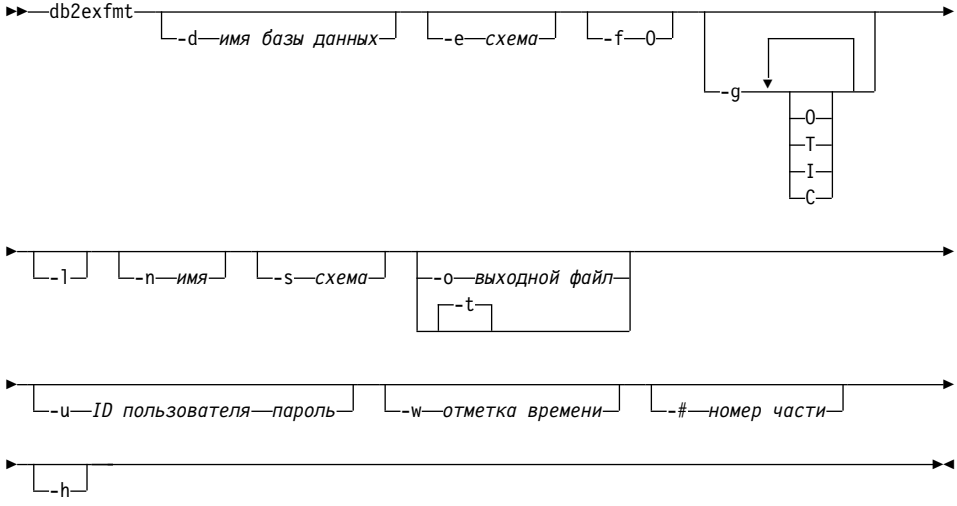
```
        FROM "DOOLE"."DEPARTMENT" A0

   Nicknames Referenced:
      REMOTE.DEPARTMENT  ID = 4  Base = DOOLE.DEPARTMENT
   #Output Columns = 3

End of section


Optimizer Plan:

RETURN
                (   1)
                  |
MSJOIN
                (   2)
               /       \
       MSJOIN            TBSCAN
       (   3)            (  13)
      /      \             |
  TBSCAN      TBSCAN      SORT
  (   4)      (   9)      (  14)
    |           |           |
   SORT         SORT      DSBQRY
  (   5)      (  10)      (  15)
    |           |           |
  DSBQRY      TBSCAN    Nickname:
  (   6)      (  11)    REMOTE
    |           |       PROJECT
 Nickname:    Table:
 REMOTE       DOOLE
 DEPARTMENT   EMPLOYEE
```

Этот план содержит все части плана первого примера, но данные двух таблиц поступают из источников данных. Обращение к этим двум таблицам выполняется с использованием распределенных подзапросов, которые в данном случае представляют собой простой выбор всех строк из данных таблиц. Когда данные возвращаются на сервер объединения, они объединяются с данными из локальной таблицы.

Приложение D. db2exfmt - Инструмент форматирования таблицы объяснения

Инструмент db2exfmt используется для форматирования содержимого таблиц объяснения. Он находится в подкаталоге `misc` каталога `sqllib` экземпляра.

Чтобы использовать этот инструмент, требуется доступ для чтения форматируемых таблиц объяснения.



-d имя базы данных
Имя базы данных, содержащей пакеты.

-e схема
Схема таблицы объяснения.

-f Флаги форматирования. В этом выпуске поддерживается только значение `O` (сводка операций).

-g Графический план. Если задана только опция `-g`, генерируется графический план, после которого выдается форматированная информация для всех таблиц. Можно также задать любую комбинацию из указанных ниже допустимых значений:

- **O** Генерировать только графический план. Не форматировать содержимое таблицы.
- **T** Включить общую стоимость для каждого оператора в графическом плане.
- **I** Включить стоимость ввода/вывода для каждого оператора в графическом плане.

C Включить ожидаемое число выводимых кортежей для каждой операции в графическом плане.

-l Учитывать регистр при обработке имен пакетов.

-n имя Имя источника требования объяснения (SOURCE_NAME).

-s схема
Схема или спецификатор источника требования объяснения (SOURCE_SCHEMA).

-o выходной файл
Имя выходного файла.

-t Направлять вывод на терминал.

-u ID пользователя пароль
Использовать при соединении с базой данных заданные ID пользователя и пароль.

Как ID пользователя, так и пароль должны быть допустимыми в соответствии с соглашениями об именовании и распознаваться базой данных.

-w отметка времени
Отметка времени объяснения. Чтобы получить последний объясненный оператор, задайте `-1`.

-# номер части
Номер части источника. Чтобы затребовать все части, задайте ноль.

-h Вывести справочную информацию. Если указана эта опция, прочие опции игнорируются, и выводится только справочная информация.

За исключением случаев, когда заданы опции `-h` и `-l`, вы получите приглашение ввести значения всех недостающих или не полностью заданных параметров.

Если схема таблицы объяснения не задана, по умолчанию используется значение переменной среды **USER**. Если эта переменная не найдена, пользователь получит приглашение задать схему таблицы объяснения.

Имя источника, схема источника и отметку времени объяснения можно указывать в форме предиката LIKE, что позволяет использовать знаки подстановки - процент (%) и подчеркивание (_), чтобы выбирать несколько источников за один вызов. Чтобы задать последний из объясненных операторов, укажите время объяснения `-1`.

Если опция `-o` указана без имени файла, а опция `-t` не задана, у пользователя будет запрошено имя файла (по умолчанию - `db2exfmt.out`). Если не заданы ни `-o`, ни `-t`, у пользователя будет запрошено имя файла (по умолчанию - вывод на терминал). Если заданы обе опции - `-o` и `-t`, вывод направляется на терминал.

Приложение E. Использование библиотеки DB2

Библиотека DB2 Universal Database состоит из электронной справки, книг (в формате PDF и HTML) и примеров программ в формате HTML. В этом разделе объясняется, какая информация содержится в ней и как ее получить.

Для оперативного доступа к этой информации можно использовать Информационный центр. Дополнительную информацию смотрите в разделе “Доступ к информации через Информационный центр” на стр. 654. Вы можете просматривать сведения о задачах, книги DB2, информацию по устранению неисправностей, программы примеров и информацию по DB2 в Web.

Файлы PDF и печатные книги DB2

Информация DB2

В следующей таблице книги DB2 разделены на 4 категории:

Руководства и справочники по DB2
В этих книгах содержится информация по DB2, общая для всех платформ.

Информация по установке и конфигурированию DB2
Эти книги применимы к DB2 для конкретной платформы. Например, есть отдельные книги *Быстрый старт* для DB2 на OS/2, Windows и на платформах на основе UNIX.

Кроссплатформенные программы примеров в формате HTML
Эти примеры - HTML-версии программ примеров, которые устанавливаются с клиентом разработки программ. Примеры используются для справок и не заменяют самих программ.

Замечания по выпуску
Эти файлы содержат самую свежую информацию, которую не успели включить в книги по DB2.

Руководства по установке, замечания по выпуску и обучающие книги в формате HTML можно просматривать прямо на компакт-диске. Большинство книг доступны в формате HTML на компакт-диске данного продукта (для просмотра) и в формате Adobe Acrobat (PDF) на компакт-диске публикаций DB2 (для просмотра и печати). Можно также заказать печатные копии в IBM; смотрите раздел “Заказ печатных копий” на стр. 650. Ниже в таблице перечислены книги, которые можно заказать.

На платформах OS/2 и Windows файлы в формате HTML можно установить в каталог `sqllib\doc\html`. Информация о DB2 переведена на различные языки, однако не на каждом языке доступна вся информация. Если информация на конкретном языке недоступна, приводится информация на английском языке.

На платформах UNIX вы можете установить версии файлов в формате HTML на нескольких языках в подкаталоги `doc/%L/html`, где *%L* - обозначение вашей национальной версии. Дополнительную информацию смотрите в соответствующей книге *Quick Beginnings* (Быстрый старт).

Вызвать книги DB2 и обратиться к информации в них можно разными способами:

- “Просмотр информации на экране” на стр. 653
- “Поиск электронной информации” на стр. 658
- “Заказ печатных копий” на стр. 650
- “Печать книг PDF” на стр. 649

Таблица 45. Информация DB2

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
	Руководства и справочники по DB2		
Administration Guide	*Administration Guide: Planning* содержит обзор понятий баз данных, информацию по вопросам разработки (в частности, по логическому и физическому проектированию баз данных) и обсуждение доступности баз данных. *Administration Guide: Implementation* содержит информацию о реализации ваших проектов, доступе к базам данных, аудите, резервном копировании и восстановлении. *Administration Guide: Performance* содержит информацию о среде баз данных, оценке и настройке производительности программ. Эти три тома *Administration Guide* можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8934.	SC09-2946 db2d1x70 SC09-2944 db2d2x70 SH43-0145 db2d3x70	db2d0

Таблица 45. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
Administrative API Reference	Описывает интерфейсы прикладного программирования (API) DB2 и структуры данных, которые можно использовать при работе с вашими базами данных. Эта книга также объясняет, как вызывать API из ваших программ.	SC09-2947 db2b0x70	db2b0
Application Building Guide	Содержит информацию о настройке среды и пошаговые инструкции для компиляции, компоновки и запуска программ DB2 в системах Windows, OS/2 и на платформах на базе UNIX.	SC09-2948 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	Содержит общие сведения о смысловых кодах APPC, CPI-C и SNA, которые могут встретиться вам при работе с продуктами DB2 Universal Database. Существует только в формате HTML.	Номера формы нет db2apx70	db2ap
Application Development Guide	Объясняет, как разрабатывать программы, обращающиеся к базам данных DB2 с использованием встроенного SQL или Java (JDBC и SQLJ). Эта книга содержит обсуждение программирования хранимых процедур, пользовательских функций, создания пользовательских типов, использования триггеров и разработки прикладных программ для работы в многораздельной среде и в системах объединения.	SC09-2949 db2a0x70	db2a0
CLI Guide and Reference	Объясняет, как разрабатывать программы, обращающиеся к базам данных DB2 при помощи интерфейса уровня вызовов (CLI) DB2 - интерфейса SQL, совместимого со спецификациями Microsoft ODBC.	SC09-2950 db2l0x70	db2l0
Command Reference	Объясняет, как использовать процессор командной строки, и описывает команды DB2, которые можно использовать для управления вашей базой данных.	SC09-2951 db2n0x70	db2n0

Таблица 45. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
Connectivity Supplement	Содержит установочную и справочную информацию по использованию DB2 for AS/400, DB2 for OS/390, DB2 for MVS, или DB2 for VM как реквестеров прикладных программ DRDA с серверами DB2 Universal Database. В этой книге описано также использование серверов прикладных программ DRDA с реквестерами прикладных программ DB2 Connect. Эта книга доступна только в форматах HTML и PDF.	Номера формы нет db2h1x70	db2h1
Data Movement Utilities Guide and Reference	Объясняет, как использовать утилиты DB2, в частности, import, export, load, AutoLoader и DPROP, которые упрощают перемещение данных.	SC09-2955 db2dmx70	db2dm
Data Warehouse Center Administration Guide	Содержит сведения о том, как построить и обслуживать хранилище данных при помощи Центра хранилища данных.	SC26-9993 db2ddx70	db2dd
Data Warehouse Center Application Integration Guide	Содержит информацию, которая поможет программистам интегрировать прикладные программы с Центром хранилища данных и Менеджером каталога данных.	SC26-9994 db2adx70	db2ad
DB2 Connect. Руководство пользователя	Содержит информацию по основным понятиям, программированию и общим вопросам использования продуктов DB2 Connect.	SC09-2954 db2c0x70	db2c0
DB2 Query Patroller Administration Guide	Содержит обзор системы DB2 Query Patroller, информацию по использованию и управлению, а также сведения по выполнению заданий при помощи утилит управления с графическим интерфейсом.	SC09-2958 db2dwx70	db2dw
DB2 Query Patroller User's Guide	Объясняет, как использовать средства и функции DB2 Query Patroller.	SC09-2960 db2wwx70	db2ww
Глоссарий	Содержит определения терминов, используемых в DB2 и его компонентах. Доступен в формате HTML, а также в книге *SQL Reference*.	Номера формы нет db2t0x70	db2t0

Таблица 45. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
Image, Audio, and Video Extenders Administration and Programming	Содержит общую информацию о модулях расширения DB2, о конфигурировании модулей расширения для работы с изображениями, звуком и видео (IAV), об управлении ими и о программировании с использованием модулей расширения IAV. Включает в себя справочную информацию, диагностическую информацию (с сообщениями) и примеры.	SC26-9929 dmbu7x70	dmbu7
Information Catalog Manager Administration Guide	Руководство по управлению каталогами данных.	SC26-9995 db2dix70	db2di
Information Catalog Manager Programming Guide and Reference	Содержит определения для проектирования интерфейсов менеджера каталогов данных.	SC26-9997 db2bix70	db2bi
Information Catalog Manager User's Guide	Содержит информацию об использовании пользовательского интерфейса менеджера каталога данных.	SC26-9996 db2aix70	db2ai
Дополнение по установке и конфигурированию	Помогает планировать, устанавливать и конфигурировать клиенты DB2 для конкретных платформ. Это дополнение содержит также информацию по связыванию, конфигурированию связей клиента и сервера, инструментам DB2 с графическим интерфейсом, DRDA AS, распределенной установке, конфигурации распределенных запросов и доступу к неоднородным источникам данных.	GC09-2957 db2iyx70	db2iy
Message Reference	Содержит список сообщений и кодов, выдаваемых DB2, Information Catalog Manager, и Data Warehouse Center, и описывает для них рекомендуемые действия. Оба тома Message Reference можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8932.	Том 1 SC09-2978 db2m1x70 Том 2 SC09-2979 db2m2x70	db2m0
OLAP Integration Server Administration Guide	Объясняет, как использовать менеджер управления сервером OLAP Integration Server.	SC27-0782 db2dpx70	нет

Таблица 45. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
OLAP Integration Server Metaoutline User's Guide	Объясняет, как создавать и заполнять метамакеты OLAP при помощи стандартного интерфейса метамакетов OLAP (а не при помощи Metaoutline Assistant).	SC27-0784 db2upx70	нет
OLAP Integration Server Model User's Guide	Объясняет, как создавать и заполнять метамакеты OLAP при помощи стандартного интерфейса моделей OLAP (а не при помощи Model Assistant).	SC27-0783 db2lpx70	нет
Руководство по установке и использованию OLAP	Содержит информацию о конфигурировании и установке для Начального комплекта OLAP.	SH43-0137 db2ipx70	db2ip
Руководство пользователя надстройки электронных таблиц для Excel	Описывает, как использовать программу электронных таблиц Excel для анализа данных OLAP.	SH43-0141 db2epx70	db2ep
Руководство пользователя надстройки электронных таблиц для Lotus 1-2-3	Описывает, как использовать программу электронных таблиц Lotus 1-2-3 для анализа данных OLAP.	SH43-0140 db2tpx70	db2tp
Replication Guide and Reference	Содержит информацию по планированию, конфигурированию, управлению и использованию инструментов IBM Replication, поставляемых с DB2.	SC26-9920 db2e0x70	db2e0
Spatial Extender User's Guide and Reference	Содержит информацию по установке, конфигурированию, управлению, программированию и устранению неисправностей для DB2 Spatial Extender. Кроме того, содержит содержательное описание понятий пространственных данных и справочную информацию (сообщения и SQL) по модулю Spatial Extender.	SC27-0701 db2sbx70	db2sb
SQL Getting Started	Введение в основные понятия SQL и примеры для многих конструкций и задач.	SC09-2973 db2y0x70	db2y0

Таблица 45. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
SQL Reference, Том 1 и Том 2	Описывает синтаксис SQL, его семантику и правила языка. Эта книга включает также информацию о совместимости версий, ограничения продукта и обзор каталогов. Оба тома *SQL Reference* можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8933.	Том 1 SC09-2974 db2s1x70 Том 2 SC09-2975 db2s2x70	db2s0
System Monitor Guide and Reference	Описывает сбор различной информации о базах данных и менеджере баз данных. Эта книга объясняет, как использовать информацию, чтобы понять работу с базой данных, улучшить производительность и найти причины ошибок.	SC09-2956 db2f0x70	db2f0
Text Extender Administration and Programming	Содержит общую информацию о модулях расширения DB2, о конфигурировании модуля расширения для работы с текстом, об управлении им и о программировании с использованием модулей расширения для работы с текстом. Включает в себя справочную информацию, диагностическую информацию (с сообщениями) и примеры.	SC26-9930 desu9x70	desu9
Troubleshooting Guide	Помогает определить причины ошибок, выполнить восстановительные операции, и использовать средства диагностики, консультируясь со Службой заказчиков DB2.	GC09-2850 db2p0x70	db2p0
Что нового	Описывает новые возможности, функции и усовершенствования в DB2 Universal Database Версии 7.	SC09-2976 db2q0x70	db2q0
Информация по установке и конфигурированию DB2			
DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings	Содержит информацию по планированию, установке и конфигурированию DB2 Connect Enterprise Edition в OS/2 и 32-битных системах Windows. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2953 db2c6x70	db2c6

Таблица 45. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
DB2 Connect Enterprise Edition for UNIX Quick Beginnings	Содержит информацию по планированию, установке, конфигурированию и выполнению заданий для DB2 Connect Enterprise Edition на платформах на основе UNIX. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2952 db2cyx70	db2cy
DB2 Connect Personal Edition Quick Beginnings	Содержит информацию по планированию, установке, конфигурированию и выполнению заданий для DB2 Connect Personal Edition в OS/2 и 32-битных средах Windows. Эта книга содержит также информацию по установке и настройке для всех поддерживаемых клиентов.	GC09-2967 db2c1x70	db2c1
DB2 Connect Personal Edition Quick Beginnings for Linux	Содержит информацию по планированию, установке, перенастройке и конфигурированию DB2 Connect Personal Edition во всех поддерживаемых версиях Linux.	GC09-2962 db2c4x70	db2c4
DB2 Data Links Manager Quick Beginnings	Содержит информацию по планированию, установке и конфигурированию DB2 Data Links Manager в AIX и 32-битных операционных системах Windows.	GC09-2966 db2z6x70	db2z6
DB2 Enterprise - Extended Edition for UNIX Quick Beginnings	Содержит информацию по планированию, установке и конфигурированию DB2 Enterprise - Extended Edition на платформах на основе UNIX. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2964 db2v3x70	db2v3
DB2 Enterprise - Extended Edition for Windows Quick Beginnings	Содержит информацию по планированию, установке и конфигурированию DB2 Enterprise - Extended Edition в 32-битных системах Windows. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2963 db2v6x70	db2v6

Таблица 45. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
DB2 for OS/2 Быстрый старт	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database Personal Edition в операционной системе OS/2. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2968 db2i2x70	db2i2
DB2 for UNIX Быстрый старт	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database на платформах на основе UNIX. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2970 db2ixx70	db2ix
DB2 for Windows Быстрый старт	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database в 32-битных системах Windows. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2971 db2i6x70	db2i6
DB2 Personal Edition Quick Beginnings	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database Personal Edition в OS/2 и в 32-битных системах Windows.	GC09-2969 db2i1x70	db2i1
DB2 Personal Edition Quick Beginnings for Linux	Содержит информацию по планированию, установке, перенастройке и конфигурированию DB2 Universal Database Personal Edition во всех поддерживаемых версиях Linux.	GC09-2972 db2i4x70	db2i4
DB2 Query Patroller Installation Guide	Содержит информацию по установке DB2 Query Patroller.	GC09-2959 db2iwx70	db2iw
DB2 Warehouse Manager Installation Guide	Содержит информацию по установке агентов хранилища, преобразователей хранилища и менеджера каталога данных.	GC26-9998 db2idx70	db2id
Кроссплатформенные программы примеров в формате HTML			

Таблица 45. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
Программы примеров в виде HTML	Содержит для справок программы примеров в виде HTML для языков программирования на всех платформах, поддерживаемых DB2. Эти программы примеров приводятся только в информационных целях. Не все из них доступны на всех языках программирования. Примеры HTML доступны, только если установлен клиент разработки программ DB2. Дополнительную информацию об этих программах смотрите в книге *Application Building Guide*.	Номера формы нет	db2hs
	Замечания по выпуску		
DB2 Connect Release Notes	Содержит самую свежую информацию, которую не успели включить в книги по DB2 Connect.	Смотрите примечание 2.	db2cr
DB2 Installation Notes	Содержит самую свежую информацию по установке, которую не успели включить в книги по DB2.	Доступна только на компакт-диске продукта.	
DB2 Release Notes	Содержит самую свежую информацию о всех продуктах DB2 и их возможностях, которую не успели включить в книги по DB2.	Смотрите примечание 2.	db2ir

Примечания:

1. Символ *x* на шестой позиции в имени файла указывает язык книги. Например, имя файла `db2d0e70` говорит о том, что это английская версия книги *Administration Guide*, а имя файла `db2d0f70` соответствует французской версии этой же книги. Для обозначений языков используются на шестой позиции имени файла используются следующие буквы:

Язык	Обозначение
Английский	e
Болгарский	u
Бразильский португальский	b
Венгерский	h
Голландский	q
Греческий	a
Датский	d

Итальянский	i
Испанский	z
Корейский	k
Немецкий	g
Норвежский	n
Польский	p
Португальский	v
Русский	r
Словенский	l
Традиционный китайский	t
Турецкий	m
Упрощенный китайский	c
Финский	y
Французский	f
Чешский	x
Шведский	s
Японский	j

2. Последнюю информацию, которую не успели включить в книги по DB2, смотрите в Замечаниях по выпуску в формате HTML и в виде ASCII-файла. HTML-версию можно вызвать через Информационный центр или с компакт-диска продукта. Чтобы посмотреть ASCII-файл:
   - На платформах на базе UNIX смотрите файл `Release.Notes`. Он расположен в каталоге `DB2DIR/Readme/%L`, где `%L` - национальная версия, а `DB2DIR`:
     - `/usr/lpp/db2_07_01` в AIX
     - `/opt/IBMdb2/V7.1` в HP-UX, PTX, Solaris, и Silicon Graphics IRIX
     - `/usr/IBMdb2/V7.1` в Linux.
   - На других платформах смотрите файл `RELEASE.TXT`. Он находится в каталоге, где установлен продукт. На платформах OS/2 можно также дважды щелкнуть по папке **IBM DB2**, а затем дважды щелкнуть по значку **Release Notes**.

Печать книг PDF

Если вы предпочитаете использовать печатные версии книг, можно напечатать файлы .pdf с компакт-диска публикаций по DB2. При помощи Adobe Acrobat Reader можно напечатать книгу целиком или же определенный диапазон страниц. Имена файлов для каждой книги в библиотеке приводятся в Табл. 45 на стр. 640.

Последнюю версию Adobe Acrobat Reader можно получить с Web-сайта фирмы Adobe, http://www.adobe.com.

Файлы PDF (расширения файлов - .PDF) входят в состав компакт-диска публикаций DB2. Для доступа к этим файлам:

1. Вставьте в устройство CD-ROM компакт-диск с публикациями DB2. На платформах на основе UNIX смонтируйте компакт-диск с публикациями DB2. Процедуру монтирования посмотрите в книге *Быстрый старт*.
2. Запустите Acrobat Reader.
3. Откройте требуемый файл PDF из одного из следующих мест:
   - На платформах OS/2 и Windows:

     Из каталога *x*:\doc*язык*, где *x* - буква компакт-диска, а *язык* двухсимвольный код страны, соответствующий вашему языку (например, RU для русского).
   - На платформах на основе UNIX:

     Из каталога /*cdrom*/doc/*%L* на компакт-диске, где */cdrom* - точка установки компакт-диска, а *%L* - имя требуемой национальной версии.

Можно также скопировать файлы PDF с компакт-диска на локальный или сетевой диск и читать их оттуда.

Заказ печатных копий

Печатные копии книг DB2 можно заказать по отдельности или в комплекте (только в Северной Америке) по номеру SBOF. Чтобы заказать книги, обратитесь к вашему авторизованному дилеру или торговому представителю IBM, или позвоните по телефону 1-800-879-2755 в Соединенных Штатах или 1-800-IBM-4YOU в Канаде. Можно также заказать книги на Web-странице Publications по адресу http://www.elink.ibmlink.ibm.com/pbl/pbl.

Есть два комплекта книг. SBOF-8935 содержит справочную и пользовательскую информацию для DB2 Warehouse Manager. SBOF-8931 содержит справочную и пользовательскую информацию для всех остальных продуктов и возможностей DB2 Universal Database. Содержимое каждого комплекта SBOF приводится в следующей таблице:

Таблица 46. Заказ печатных книг

Номер SBOF	Содержит книги	
SBOF-8931	• Administration Guide: Planning • Administration Guide: Implementation • Administration Guide: Performance • Administrative API Reference • Application Building Guide • Application Development Guide • CLI Guide and Reference • Command Reference • Data Movement Utilities Guide and Reference • Data Warehouse Center Administration Guide • Data Warehouse Center Application Integration Guide • DB2 Connect User's Guide • Installation and Configuration Supplement • Image, Audio, and Video Extenders Administration and Programming • Справочник по сообщениям, Тома 1 и 2	• OLAP Integration Server Administration Guide • OLAP Integration Server Metaoutline User's Guide • OLAP Integration Server Model User's Guide • OLAP Integration Server User's Guide • OLAP Setup and User's Guide • OLAP Spreadsheet Add-in User's Guide for Excel • OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3 • Replication Guide and Reference • Spatial Extender Administration and Programming Guide • SQL Getting Started • SQL Reference, Volumes 1 and 2 • System Monitor Guide and Reference • Text Extender Administration and Programming • Troubleshooting Guide • What's New
SBOF-8935	• Information Catalog Manager Administration Guide • Information Catalog Manager User's Guide • Information Catalog Manager Programming Guide and Reference	• Query Patroller Administration Guide • Query Patroller User's Guide

Электронная документация DB2

Обращение к электронной справке

Для всех компонентов DB2 доступна электронная справка. Различные типы справки перечислены в следующей таблице.

Тип справки	Содержание	Как вызвать...
Справка по командам	Объясняет синтаксис команд процессора командной строки.	В процессоре командной строки в интерактивном режиме введите: `? команда` где *команда* - ключевое слово для команды целиком. Например, `? catalog` выводит справку по всем командам CATALOG, а `? catalog database` выводит справку по команде CATALOG DATABASE.
Справка по Ассистенту конфигурирования клиента *Справка по Командному центру* *Справка по Центр управления* *Справка по Data Warehouse Center* *Справка по анализатору событий* *Справка по менеджеру каталога данных* *Справка по центру управления сателлитами* *Справка по центру сценариев*	Объясняет задания, которые можно выполнить в окне или в записной книжке. Справка содержит обзор и предварительную информацию, которую надо знать, и описывает, как использовать управляющие элементы окна или записной книжки.	В окне или в записной книжке нажмите кнопку **Справка** или клавишу **F1**.

Тип справки	Содержание	Как вызвать...

Справка по сообщениям	Описывает для сообщения причину и действия, которые следует предпринять.	В процессоре командной строки в интерактивном режиме введите:
		`? XXXnnnnn`
		где *XXXnnnnn* - идентификатор допустимого сообщения.
		Например, `? SQL30081` выводит справку по сообщению SQL30081.
		Чтобы смотреть справку по сообщению поэкранно, введите:
		`? XXXnnnnn | more`
		Чтобы записать справку по сообщению в файл, введите:
		`? XXXnnnnn > имяфайла.рсш`
		где *имяфайла.рсш* - имя файла, где вы хотите сохранить справку.

Справка по SQL	Объясняет синтаксис операторов SQL.	В процессоре командной строки в интерактивном режиме введите: `help оператор` где *оператор* - оператор SQL. Например, `help` SELECT выводит справку по оператору SELECT. **Примечание:** Справка по SQL недоступна на платформах на основе UNIX.
Справка по SQLSTATE	Объясняет состояния SQL и коды классов.	В процессоре командной строки в интерактивном режиме введите: `? sqlstate или ? код класса` где *sqlstate* - допустимый пятизначный код SQL, а *код класса* - первые две цифры sqlstate. Например, `? 08003` выводит справку по состоянию SQL 08003, а `? 08` выводит справку по коду класса 08.

Просмотр информации на экране

Книги, поставляемые с этим продуктом, записаны в формате HTML. Этот формат позволяет искать и просматривать информацию и поддерживает гипертекстовые ссылки. Он упрощает также совместное использование библиотеки на сайте.

Электронные книги и примеры программ можно просматривать в любом браузере, который поддерживает спецификации HTML Версии 3.2.

Чтобы просмотреть книги или примеры программ:

- Если вы работаете с инструментами администратора DB2, используйте Информационный центр.
- В браузере выберите **Файл —>Открыть страницу**. На открытой странице приводятся описания и ссылки на информацию по DB2:
  - На платформах на базе UNIX откройте страницу:

    INSTHOME/sqllib/doc/*%L*/html/index.htm

    где *%L* - имя национальной версии.
  - На других платформах откройте страницу:

    sqllib\doc\html\index.htm

    на диске, где установлена DB2.

    Если вы не установили Информационный центр, эту страницу можно открыть, щелкнув дважды по значку **Информация DB2**. В зависимости от того, в какой системе вы работаете, этот значок может находиться в основной папке продукта или в меню Windows Пуск.

Установка браузера Netscape

Если у вас еще не установлен браузер Web, можно установить Netscape с компакт-диска Netscape, включенного в состав продукта. Чтобы получить подробные указания по установке, выполните следующие действия:

1. Вставьте в устройство CD-ROM компакт-диск Netscape.
2. На платформах на основе UNIX смонтируйте компакт-диск. Процедуру монтирования посмотрите в книге *Быстрый старт*.
3. Прочтите инструкции по установке в файле CDNAV*nn*.txt, где *nn* - двухсимвольный идентификатор языка. Этот файл находится в корневом каталоге компакт-диска.

Доступ к информации через Информационный центр

Информационный центр обеспечивает быстрый доступ к информации о продуктах DB2. Информационный центр доступен на всех платформах, где есть инструменты администратора DB2.

Чтобы открыть Информационный центр, щелкните дважды по значку Информационный центр. В зависимости от того, в какой системе вы работаете, этот значок может находиться в основной папке продукта или в меню **Пуск**.

На платформах Windows можно также вызвать Информационный центр через панель задач и через меню **Справка** DB2.

Информационный центр дает шесть типов информации. Для обращения к информации одного из этих типов выберите соответствующую закладку.

Задания Основные задания, которые вы можете выполнить в DB2.

Справочник Справочная информацию по таким элементам DB2, как ключевые слова, команды и API.

Книги Книги DB2.

Устранение неисправностей
Список сообщений об ошибках и рекомендуемых действий по категориям.

Программы примеров
Программы примеров, поставляемые с клиентом разработки программ DB2. Если вы не установили клиент разработки программ DB2, эта закладка не выводится.

Web Информация по DB2 в WWW. Чтобы посмотреть эту информацию, ваша система должна быть подключена к Web.

Когда вы выбираете пункт в одном из списков, информационный центр запускает программу просмотра для вывода информации. Этой программой может быть программа просмотра системной справки, редактор или браузер Web в зависимости от того, какую информацию вы выбрали.

Информационный центр поддерживает возможность поиска, и вы можете искать определенную тему, не просматривая книги целиком.

Для полнотекстового поиска выберите гипертекстовую ссылку в Информационном центре и откройте поисковую форму **Поиск электронной информации DB2**.

Обычно сервер поиска HTML запускается автоматически. Если поиск информации HTML не работает, вам, возможно надо запустить сервер поиска одним из следующих способов:

В Windows
Выберите **Пуск**, затем **Программы—> IBM DB2 —> Информация —> Запустить сервер поиска HTML**.

В OS/2 Щелкните дважды по папке **DB2 for OS/2**, а затем щелкните дважды по значку **Запустить сервер поиска HTML**.

Если у вас есть проблемы с использованием поиска информации HTML, посмотрите замечания по выпуску.

Примечание: Функция поиска недоступна в средах Linux, PTX и Silicon Graphics IRIX.

Использование мастеров DB2

Мастера помогают вам выполнять конкретные задачи управления, ведя последовательно по шагам необходимых действий. Мастера доступны в Центре управления и в Ассистенте конфигурирования клиента. Список мастеров с соответствующими задачами приведен в следующей таблице.

Примечание: Мастера по созданию баз данных, индексов, конфигурированию многоузлового изменения и производительности доступны в среде многораздельных баз данных.

Мастер	Помогает вам...	Как вызвать...
по добавлению баз данных	Каталогизировать базу данных на клиентской рабочей станции	В Ассистенте конфигурирования клиента нажмите кнопку **Добавить**.
по резервному копированию базы данных	Определить, создать и включить в расписание резервное копирование.	В Центре управления щелкните правой кнопкой мыши по базе данных, для которой вам нужна резервная копия, и выберите **Резервное копирование —> Базы данных при помощи мастера**.
по конфигурированию многоузлового изменения	Конфигурировать многоузловые изменения, распределенные транзакции или двухфазное принятие.	В Центре управления щелкните правой кнопкой мыши по папке **Базы данных** и выберите **Многоузловое изменение**.
по созданию баз данных	Создать базу данных и выполнить основные задачи конфигурирования.	В Центре управления щелкните правой кнопкой мыши по папке **Базы данных** и выберите **Создать —> Базу данных при помощи мастера**.
по созданию таблиц	Выбрать типы основных данных и создать первичные ключи для таблицы.	В Центре управления щелкните правой кнопкой мыши по значку **Таблицы** и выберите **Создать —> Таблицу при помощи мастера**.
по созданию табличных пространств	Создать новое табличное пространство.	В Центре управления щелкните правой кнопкой мыши по значку **Табличные пространства** и выберите **Создать —> Табличное пространство при помощи мастера**.
Создать индекс	Выбрать, какие индексы создать или отбросить для всех ваших запросов.	В Центре управления щелкните правой кнопкой мыши по значку **Индекс** и выберите **Создать —> Индекс при помощи мастера**.

Мастер	Помогает вам...	Как вызвать...
по настройке производительности	Настроить производительность базы данных, изменив параметры конфигурации в соответствии с вашими требованиями.	В Центре управления щелкните правой кнопкой мыши по базе данных, которую вы хотите настроить, и выберите **Конфигурировать производительность при помощи мастера**. Для многораздельной среды баз данных в окне Разделы баз данных щелкните правой кнопкой мыши по первому разделу баз данных, который вы хотите настроить, и выберите **Конфигурировать производительность при помощи мастера**.
по восстановлению баз данных	Восстановить базу данных после сбоя. Он поможет понять, какую резервную копию использовать, и какие журналы использовать при повторе.	В Центре управления щелкните правой кнопкой мыши по базе данных, которую вы хотите восстановить, и выберите **Восстановить —> Базу данных при помощи мастера**.

Установка сервера документации

По умолчанию информация по DB2 устанавливается в вашей локальной системе. Это значит, что каждый, кому требуется доступ к информации по DB2, должен устанавливать одни и те же файлы. Чтобы держать информацию по DB2 в едином месте, выполните следующие действия:

1. Скопируйте все файлы и подкаталоги каталога `\sqllib\doc\html` вашей локальной системы на сервер Web. Каждая книга находится в своем собственном подкаталоге, где записаны все необходимые для нее файлы HTML и GIF. Структура подкаталогов должна остаться без изменений.
2. Сконфигурируйте сервер Web на поиск файлов на новом месте. Дополнительную информацию смотрите в приложении NetQuestion руководства *Дополнение по установке и конфигурированию*.
3. Если вы используете Java-версию Информационного центра, можно задать базовый URL для всех файлов HTML. Этот URL надо использовать для списка книг.
4. Когда вы сможете просматривать файлы книг, можно пометить закладками часто используемые темы. Вероятно, вы захотите пометить закладками следующие страницы:
   - Список книг
   - Содержания часто используемых книг

- Часто требуемые статьи, например, тему ALTER TABLE
- Форму поиска

Информацию о том, как работать с файлами электронной документации на центральном компьютере, смотрите в приложении NetQuestion руководства *Дополнение по установке и конфигурированию*.

Поиск электронной информации

Для поиска информации в файлах HTML используйте один из следующих способов:

- Нажмите кнопку **Поиск** в верхнем фрейме. При помощи формы поиска найдите нужную тему. Эта функция недоступна в средах Linux, PTX и Silicon Graphics IRIX.
- Нажмите кнопку **Индекс** в верхнем фрейме. При помощи индекса найдите в книге нужную тему.
- Выведите содержание или индекс справки или книги HTML, затем при помощи функции поиска браузера Web найдите в книге нужную тему.
- При помощи функции закладок браузера Web можно быстро вернуться к определенной теме.
- Используйте для поиска определенных тем функцию поиска информационного центра. Подробности смотрите в разделе “Доступ к информации через Информационный центр” на стр. 654.

Приложение F. Замечания

IBM может предлагать описанные продукты, услуги и возможности не во всех странах. Сведения о продуктах и услугах, доступных в настоящее время в вашей стране, можно получить в местном представительстве IBM. Любые ссылки на продукты, программы или услуги IBM не означают явным или неявным образом, что можно использовать только продукты, программы или услуги IBM. Разрешается использовать любые функционально эквивалентные продукты, программы или услуги, если при этом не нарушаются права IBM на интеллектуальную собственность. Однако ответственность за оценку и проверку работы любых продуктов, программ и услуг других фирм лежит на пользователе.

Фирма IBM может располагать патентами или рассматриваемыми заявками на патенты, относящимися к предмету данного документа. Получение этого документа не означает предоставления каких-либо лицензий на эти патенты. Запросы по поводу лицензий следует направлять в письменной форме по адресу:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

По поводу лицензий, связанных с использованием наборов двухбайтных символов (DBCS), обращайтесь в отдел интеллектуальной собственности IBM в вашей стране или направьте запрос в письменной форме по адресу:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

Следующий абзац не применяется в Великобритании или в любой другой стране, где подобные заявления противоречат местным законам: КОРПОРАЦИЯ INTERNATIONAL BUSINESS MACHINES ПРЕДСТАВЛЯЕТ ДАННУЮ ПУБЛИКАЦИЮ “КАК ЕСТЬ” БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ПРЕДПОЛАГАЕМЫЕ ГАРАНТИИ СОВМЕСТИМОСТИ, РЫНОЧНОЙ ПРИГОДНОСТИ И СООТВЕТСТВИЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. В некоторых странах для определенных сделок подобные оговорки не допускаются, таким образом, это утверждение может не относиться к вам.

Данная информация может содержать технические неточности и типографские опечатки. Периодически в информацию вносятся изменения, они будут включены в новые издания этой публикации. Фирма IBM может в любое время без уведомления вносить изменения и усовершенствования в продукты и программы, описанные в этой публикации.

Любые ссылки в данной информации на Web-сайты, не принадлежащие IBM, приводятся только для удобства и никоим образом не означают поддержки IBM этих Web-сайтов. Материалы этих Web-сайтов не являются частью данного продукта IBM и вы можете использовать их только на собственную ответственность.

IBM может использовать или распространять присланную вами информацию любым способом, как фирма сочтет нужным, без каких-либо обязательств перед вами.

Если обладателю лицензии на данную программу понадобятся сведения о возможности: (i) обмена данными между независимо разработанными программами и другими программами (включая данную) и (ii) совместного использования таких данных, он может обратиться по адресу:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

Такая информация может быть предоставлена на определенных условиях (в некоторых случаях к таким условиям может относиться оплата).

Лицензированная программа, описанная в данной публикации, и все лицензированные материалы, доступные с ней, предоставляются IBM на условиях IBM Customer Agreement (Соглашения IBM с заказчиком), Международного соглашения о лицензиях на программы IBM или эквивалентного соглашения.

Приведенные данные о производительности измерены в контролируемой среде. Таким образом, результаты, полученные в других операционных средах, могут существенно отличаться от них. Некоторые показатели измерены в системах разработки и нет никаких гарантий, что в общедоступных системах эти показатели будут теми же. Более того, некоторые результаты могут быть получены путем экстраполяции. Реальные результаты могут отличаться от них. Пользователи должны проверить данные для своих конкретных сред.

Информация о продуктах других фирм получена от поставщиков этих продуктов, из их опубликованных объявлений или из других общедоступных

источников. Фирма IBM не проверяла эти продукты и не может подтвердить точность измерений, совместимость или прочие утверждения о продуктах других фирм. Вопросы о возможностях продуктов других фирм следует направлять поставщикам этих продуктов.

Все утверждения о будущих планах и намерениях IBM могут быть изменены или отменены без уведомлений, и описывают исключительно цели фирмы.

Эта информация может содержать примеры данных и отчетов, иллюстрирующие типичные деловые операции. Чтобы эти примеры были правдоподобны, в них включены имена лиц, названия компаний и товаров. Все эти имена и названия вымышлены и любое их сходство с реальными именами и адресами полностью случайно.

ЛИЦЕНЗИЯ НА КОПИРОВАНИЕ:

Эта информация может содержать примеры прикладных программ на языках программирования, иллюстрирующих приемы программирования для различных операционных платформ. Разрешается копировать, изменять и распространять эти примеры программ в любой форме без оплаты фирме IBM для целей разработки, использования, сбыта или распространения прикладных программ, соответствующих интерфейсу прикладного программирования операционных платформ, для которых эти примера программ написаны. Эти примеры не были всесторонне проверены во всех возможных условиях. Поэтому IBM не может гарантировать их надежность, пригодность и функционирование.

Каждая копия программ примеров или программ, созданных на их основе, должна содержать следующее замечание об авторских правах:

© (название вашей фирмы) (год). Части этого кода построены на основе примеров программ IBM Corp. © Copyright IBM Corp. _введите год или годы_. Все права защищены.

Товарные знаки

Следующие термины (они могут быть помечены звездочкой - *) являются товарными знаками корпорации International Business Machines в Соединенных Штатах и/или в других странах:

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

Следующие термины являются товарными знаками или зарегистрированными товарными знаками других компаний:

Microsoft, Windows и Windows NT - товарные знаки или зарегистрированные товарные знаки Microsoft Corporation.

Java, все товарные знаки и логотипы на основе Java и Solaris - товарные знаки Sun Microsystems, Inc. в Соединенных Штатах и/или в других странах.

Tivoli и NetView - товарные знаки Tivoli Systems Inc. в Соединенных Штатах и/или других странах.

UNIX - зарегистрированный товарный знак в Соединенных Штатах и в других странах, его использование лицензируется исключительно фирмой X/Open Company Limited.

Названия других компаний, продуктов и услуг (они могут быть отмечены двойной звездочкой - **) могут быть товарными знаками или марками сервиса других фирм.

Индекс

E

F

H

I

K

L

M

N

P

R

S

T

V

A

Г

Д

Ж

З

И

К

Л

M

H

O

П

P

C

T

У

Ф

Х

Ц

Ч

Э

Как связаться с IBM

Если у вас имеется техническая проблема, пожалуйста, перед обращением к службе поддержки пользователей DB2 просмотрите еще раз и выполните действия, рекомендуемые в руководстве *Troubleshooting Guide*. В этом руководстве описано, какую информацию надо собрать, чтобы служба поддержки пользователей DB2 могла лучше помочь вам.

Чтобы получить информацию или заказать любой из продуктов DB2 Universal Database, обратитесь к представителю IBM в местном отделении или к авторизованному продавцу программных продуктов IBM.

Если вы находитесь в США, позвоните по одному из следующих номеров:

- 1-800-237-5511, чтобы обратиться в службу поддержки
- 1-888-426-4343, чтобы узнать о доступных формах обслуживания.

Информация о продукте

Если вы находитесь в США, позвоните по одному из следующих номеров:

- 1-800-IBM-CALL (1-800-426-2255) или 1-800-3IBM-OS2 (1-800-342-6672), чтобы заказать продукты или получить общую информацию.
- 1-800-879-2755, чтобы заказать публикации.

http://www.ibm.com/software/data/
На страницах DB2 в WWW содержится текущая информация DB2: новости, описания продуктов, учебные планы и т.д.

http://www.ibm.com/software/data/db2/library/
DB2 Product and Service Technical Library содержит ответы на часто задаваемые вопросы, исправления, книги и свежую техническую информацию по DB2.

Примечание: Эта информация может быть только в английском варианте.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
На сайте заказов International Publications приводится информация о том, как заказывать книги.

http://www.ibm.com/education/certify/
На этом сайте представлена программа Professional Certification Program IBM и приводится информация о сертификационных испытаниях для многих продуктов IBM, в том числе DB2.

ftp.software.ibm.com

Зарегистрируйтесь как аноним. В каталоге `/ps/products/db2` можно найти демо-версии, исправления, информацию и инструменты для DB2 и многих других продуктов.

comp.databases.ibm-db2, bit.listserv.db2-l

В этих группах новостей пользователи обмениваются опытом работы с продуктами DB2.

В Compuserve: GO IBMDB2

Введите эту команду, чтобы попасть на форумы IBM DB2 Family. Через эти форумы поддерживаются все продукты DB2.

Информацию о том, как связаться с IBM из других стран, смотрите в Приложении A книги *IBM Software Support Handbook*. Этот документ можно найти в Web, обратившись по адресу: http://www.ibm.com/support/ и выбрав ссылку на IBM Software Support Handbook у нижнего края страницы.

Примечание: В некоторых странах авторизованные дилеры IBM должны обращаться не в центр поддержки IBM, а в структуры поддержки дилеров.

Напечатано в Дании

SH43-0145-01