IBM® DB2® Universal Database

Руководство администратора: Планирование

Версия 7

SH43-0146-00

IBM® DB2® Universal Database

Руководство администратора: Планирование

Версия 7

SH43-0146-00

Перед тем как использовать данный документ и продукт, описанный в нем, прочтите общие сведения под заголовком “Приложение F. Замечания” на стр. 451.

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 *Руководство администратора: Планирование*)
- Информация о реализации баз данных и управлении ими (том *Руководство администратора: Реализация*)
- Информация о конфигурировании и настройке среды вашей базы данных для повышения производительности (том *Руководство администратора: Производительность*).

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

- **Командный процессор**, 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 командного процессора смотрите в книге *Command Reference*.
- **Интерфейс прикладного программирования**, позволяющий вызывать утилиты DB2 из прикладной программы. Дополнительную информацию об использовании интерфейса 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 можно получить, выбрав **С чего начать** в выпадающем меню **Справка** окна Центра 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 по следующим основным темам:

В мире DB2 Universal Database

- В разделе Глава 1. Управление DB2 Universal Database содержатся предварительные сведения о DB2 Universal Database и ее обзор.

Концепции баз данных

- В разделе Глава 2. Основные понятия реляционных баз данных приводится обзор объектов баз данных, включая объекты восстановления, объекты хранения и системные объекты.
- В разделе Глава 3. 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 Глава 4. Параллельные системы баз данных излагаются начальные сведения о типах параллелизма, доступных при работе с DB2.
- В разделе Глава 5. О хранилищах данных дается обзор работы с хранилищами данных и возникающих при этом задач.

- В разделе Глава 6. О модуле Spatial Extender дается понятие о модуле Spatial Extender, объясняется его назначение и обсуждаются обрабатываемые с его помощью данные.

Проектирование баз данных

- В разделе Глава 7. Логическое проектирование базы данных обсуждаются концепции логической структуры базы данных и даются указания по разработке баз данных.
- В разделе Глава 8. Физическая структура базы данных содержатся указания по разработке физической структуры базы данных, в том числе по вопросам хранения данных.
- В разделе Глава 9. Проектирование распределенных баз данных обсуждается, как обращаться к нескольким базам данных в ходе одной транзакции.
- В разделе Глава 10. Планирование использования менеджеров транзакций обсуждается, как использовать ваши базы данных в среде обработки распределенных транзакций, например, CICS.
- Раздел Глава 11. Проектирование высокой доступности содержит обзор поддержки высокой доступности восстановления после отказов, обеспечиваемой DB2.

Системы высокой доступности

- В разделе Глава 12. HACMP ES для AIX (High Availability Cluster Multi-processing, Enhanced Scalability - Мультипроцессорная кластерная обработка с высокой доступностью и улучшенной масштабируемостью) обсуждается поддержка DB2 высокой доступности для восстановления после отказов в AIX.
- В разделе Глава 13. Высокая доступность в среде Windows NT обсуждается поддержка DB2 высокой доступности для восстановления после отказов в Windows NT.
- В разделе Глава 14. DB2 и высокая доступность в Sun Cluster 2.2 обсуждается поддержка DB2 высокой доступности для восстановления после отказов в операционной системе Sun Solaris.

Дополнения

- В разделе Приложение B. Правила именования приводятся правила именования баз данных и объектов.
- В разделе Приложение C. Планирование перенастройки баз данных приводится информация о перенастройке баз данных в Версию 7.
- В разделе Приложение D. Несовместимость выпусков рассматриваются виды несовместимости, возникающие при переходе от выпуска к выпуску, вплоть до Версии 7.

- В разделе Приложение E. Поддержка национальных языков (NLS) дается понятие о поддержке национальных языков в DB2, включая информацию о странах, языках и кодовых страницах.
- В разделе Приложение A. Использование библиотеки DB2 приводится информация о структуре библиотеки DB2, включая мастера, электронную справку, сообщения и книги.

Краткий обзор других томов Руководства администратора

Руководство администратора: Реализация

Книга *Руководство администратора: Реализация* посвящена реализации вашего проекта базы данных. Отдельные главы и приложения из этого тома кратко описаны здесь:

Управление при помощи Центра управления

- Раздел ″Управление DB2 при помощи инструментов GUI″ знакомит с инструментами графического пользовательского интерфейса (GUI) для управления базой данных.

Реализация вашего проекта

- В разделе ″Перед созданием базы данных″ обсуждаются предварительные требования для создания базы данных.
- В разделе ″Создание базы данных″ рассматриваются задачи по созданию базы данных и связанных с ней объектов базы данных.
- В разделе ″Изменение базы данных″ обсуждается, что необходимо сделать перед изменением базы данных, и рассматриваются 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 разделе ″Восстановление базы данных″ обсуждаются факторы, которые следует принимать во внимание при выборе методов восстановления баз данных и табличных пространств, включая их резервное копирование с последующим восстановлением и метод восстановления с повтором транзакций.

Приложения

- В разделе ″Использование служб каталога распределенной вычислительной среды (DCE)″ приводится информация о том, как пользоваться службами каталога DCE.
- В разделе ″Обработчик пользователя для восстановления базы данных″ обсуждается, как 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.

Руководство администратора: Производительность

Книга *Руководство администратора: Производительность* посвящена вопросам производительности, а именно, темам и вопросам, связанным с заданием, тестированием и повышением производительности вашей прикладной программы, а также самого продукта DB2 Universal Database. Отдельные главы и приложения из этого тома кратко описаны здесь:

Вводные замечания по производительности

- Раздел ″Элементы производительности″ знакомит с идеями и особенностями управления производительностью DB2 UDB и ее повышения.
- В разделе ″Обзор архитектуры и процессов обработки″ излагаются основы архитектуры и процессов DB2 Universal Database.

Настройка производительности прикладных программ

- В разделе ″Особенности прикладных программ″ описываются некоторые методы повышения производительности базы данных при разработке ваших прикладных программ.
- В разделе ″Особенности среды″ описываются некоторые методы повышения производительности базы данных при задании параметров среды вашей базы данных.
- В разделе ″Статистика системного каталога″ описывается, как можно собрать статистику о ваших данных и использовать ее для обеспечения оптимальной производительности.
- В разделе ″Основные сведения о компиляторе SQL″ описывается, что происходит с оператором SQL при его компиляции с помощью компилятора SQL.
- В разделе ″Средства объяснения SQL″ описываются средства объяснения, позволяющие исследовать варианты доступа к вашим данным, выбранные компилятором SQL.

Настройка и конфигурирование вашей системы

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

Приложения

- В приложении ″Реестр и переменные среды DB2″ приводятся значения реестра профиля и переменных среды.
- В приложении ″Таблицы и определения объяснения″ приводится информация о таблицах, используемых средствами объяснения DB2, и о том, как создавать эти таблицы.
- В приложении ″Инструменты объяснения SQL″ приводится информация об инструментах объяснения DB2: db2expln и dynexpln.
- В приложении ″db2exfmt – инструмент форматирования таблиц объяснения″ приводится информация об использовании этого инструмента объяснения DB2 для форматирования данных таблиц объяснения.

Часть 1. Мир DB2 Universal Database

Глава 1. Управление DB2 Universal Database

DB2 обеспечивает гибкость выбора для работы аппаратных конфигураций в широком диапазоне. Она позволяет вам выбрать, как наилучшим образом подобрать аппаратуру и прикладные программы для определенной конфигурации продукта DB2.

DB2 поддерживается также среды баз данных различных уровней сложности, и для каждой среды есть свои определенные особенности и свои задачи. Они обсуждаются подробно в руководстве *Руководство администратора* и других книгах библиотеки DB2 (смотрите приложение “Приложение A. Использование библиотеки DB2” на стр. 353). В некоторых случаях целые разделы этих книг относятся только к определенным средам. Узнать, с какими главами этого и других томов книги *Руководство администратора* (*Руководство администратора: Реализация* и *Руководство администратора: Производительность*) вам стоит ознакомиться, можно из предисловия к настоящей книге (″Об этой книге″).

Если вы мало знакомы с системами управления реляционными базами данных (relational database management systems, RDBMS) или с DB2, вам будет полезен раздел ″Основные понятия реляционных баз данных″. Если вы знакомы с этими понятиями или не нуждаетесь в таком обзоре, можно пропустить этот раздел и перейти прямо к разделам, подробно описывающим более конкретные темы:

- Системы объединения. В этом разделе описаны системы управления базами данных (СУБД), которые дают возможность программам и пользователям выполнять операторы SQL, обращающиеся (в одном операторе) к нескольким СУБД или базам данных.
- Системы параллельных баз данных. Этот раздел содержит введение в типы параллелизма, доступные в системе DB2. Компоненты задачи, например, запроса к базам данных, можно выполнять параллельно, что существенно повышает производительность.
- Распределенная обработка транзакций. В этом разделе обсуждается, как обращаться к нескольким базам данных в одной транзакции и как использовать базы данных в среде распределенной обработки транзакций.
- Системы высокой доступности. В этом разделе приводится обзор восстановления при отказах для поддержания высокой доступности в системе DB2. Возможность восстановления при отказах позволяет автоматически передавать рабочую нагрузку с одного процессора на другой при отказах аппаратного обеспечения.

DB2 позволяет удовлетворить ваши специальные нужды в области управления информацией, такие как:

- *Репликация*, позволяющая регулярно копировать информацию в несколько удаленных баз данных. Если вам нужно автоматически копировать изменения в главной базе данных в остальные базы, вы можете воспользоваться возможностями репликации в DB2, чтобы задать, какую информацию нужно копировать, в какие таблицы баз данных и как часто. Если вы хотите использовать возможности репликации DB2, обратитесь к книге *Replication Guide and Reference*. В ней описаны понятия репликации информации в DB2, планирование, конфигурирование среды репликации и управление ей.
- *Хранилище данных*, в котором вы можете хранить "информативные данные" - данные, извлеченные из рабочих данных и преобразованные для принятия решений конечными пользователями. Например, средствами хранилища данных можно скопировать данные продаж из рабочей базы данных, выполнить вычисления для составления по этой информации сводных данных и записать полученные сводные данные в другую базу данных, отдельную от рабочей. Можно строить запросы к этой отдельной базе данных (*хранилищу*), не затрагивая рабочих баз данных. Подробную информацию о хранилищах данных смотрите в руководстве *Data Warehouse Center Administration Guide*.
- *Географическая информационная система* (ГИС), которую можно создать с помощью модуля Spatial Extender. ГИС - это совокупность объектов, информации и прикладных программ, позволяющая генерировать и анализировать географическую информацию о географических объектах. В модуле Spatial Extender географический объект может быть представлен строкой в таблице или производной таблице или частью такой строки. Подробную информацию об использовании Spatial Extender смотрите в руководстве *Spatial Extender User's Guide and Reference*.

В руководстве *Руководство администратора: Планирование* также описано проектирование баз данных и особенности проектирования логических и физических баз данных для DB2. Изложены и другие вопросы планирования, такие как планирование перенастройки баз данных, описание несовместимостей, которые могли бы затронуть ваши прикладные программы (*несовместимость* - это часть DB2 Universal Database, которая работает иначе, чем в предыдущем выпуске DB2; в случае использования в существующей прикладной программе, несовместимость будет приводить к неожиданным результатам, вызывать изменение прикладной программы или снижать производительность), и использование поддержки национальных языков.

В книге *Руководство администратора: Реализация* описываются подробности реализации проекта ваших баз данных. Изложены такие темы, как создание и изменение баз данных, защита баз данных, восстановление баз данных и управление системой DB2 с помощью Центра управления, графического пользовательского интерфейса DB2.

В книге *Руководство администратора: Производительность* рассматриваются темы и вопросы, касающиеся установки, проверки и повышения производительности вашей программы и самой системы DB2.

Часть 2. Концепции баз данных

Глава 2. Основные понятия реляционных баз данных

В этой главе рассматриваются следующие темы:

Обзор объектов базы данных

В этом разделе дается обзор следующих ключевых объектов базы данных:

- Экземпляры
- Базы данных
- Группы узлов
- Таблицы
- Производные таблицы
- Индексы
- Схемы
- Таблицы системного каталога

На рис. 1 на стр. 10 показаны отношения между некоторыми из этих объектов. Здесь также показано, что таблицы, индексы и длинные данные хранятся в табличных пространствах.

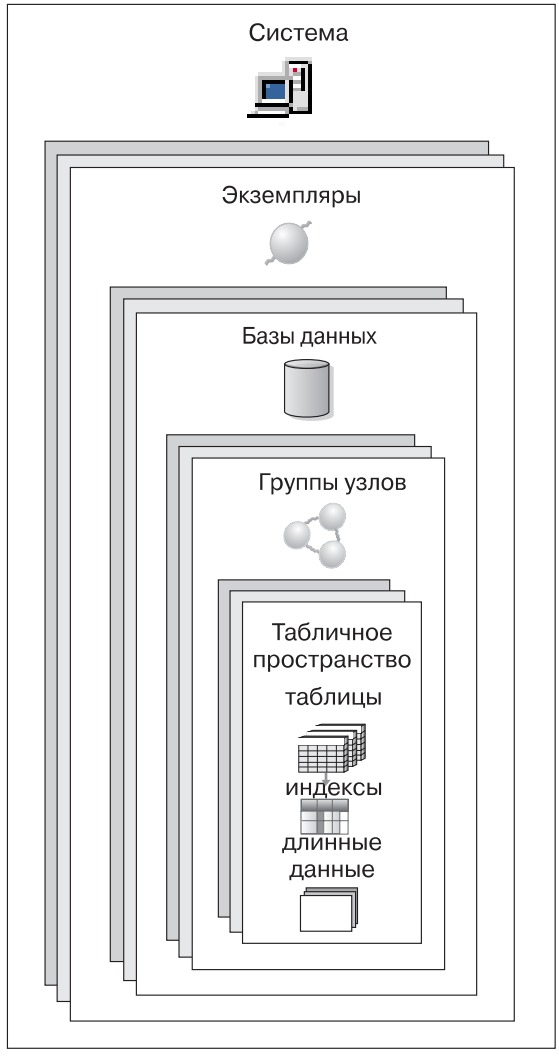


Рисунок 1. Отношения между некоторыми объектами баз данных

Экземпляры

Экземпляр (другое название - *менеджер баз данных*) - это код DB2, предназначенный для работы с данными. Он управляет возможными действиями над данными и назначенными для них системными ресурсами. Каждый экземпляр является полнофункциональной средой. Он содержит все разделы баз данных для данной параллельной системы баз данных (смотрите “Глава 4. Параллельные системы баз данных” на стр. 67). У экземпляра есть собственные базы данных (к ним не могут обращаться другие экземпляры), а все

его разделы баз данных совместно используют одни и те же системные каталоги. Кроме того, у экземпляра есть своя защита, отдельная от остальных экземпляров, установленных на этом же компьютере (или в системе).

Базы данных

В *реляционной базе данных* данные представлены в виде собрания таблиц. Таблица состоит из определенного числа столбцов и произвольного числа строк. Каждая база данных включает в себя набор таблиц системного каталога, описывающий логическую и физическую структуру данных, файл конфигурации, содержащий значения параметров этой базы данных, и журнал восстановления с текущими и архивными транзакциями.

Группы узлов

Группа узлов - это набор из одного или нескольких разделов базы данных. При создании таблиц для базы данных сначала вы создаете группу узлов, где будут храниться табличные пространства, а затем создаете табличное пространство, где будут храниться таблицы. Дополнительную информацию о группах узлов смотрите в разделе “Группы узлов и разделение данных” на стр. 68. “Глава 4. Параллельные системы баз данных” на стр. 67 содержит определение раздела базы данных. Дополнительную информацию о табличных пространствах смотрите в разделе “Табличные пространства” на стр. 16.

Таблицы

В реляционной базе данных данные представлены как собрание таблиц. *Таблица* состоит из данных, логически упорядоченных по столбцам и строкам. Все данные базы данных и таблиц собраны в табличных пространствах. Дополнительную информацию о табличных пространствах смотрите в разделе “Табличные пространства” на стр. 16. Данные в таблице логически связаны, а между таблицами могут быть заданы отношения. Данные можно просматривать и работать с ними на основе математических правил и операций.

Доступ к табличным данным осуществляется с помощью SQL (Structured Query Language - язык структурированных запросов, смотрите справочник *SQL Reference*), стандартизированного языка для определения данных и работы с ними в реляционных базах данных. *Запрос* применяется в программах или пользователями для получения данных из базы данных. Запрос использует SQL для создания оператора следующей формы:

```
SELECT <имя_данных> FROM <имя_таблицы>
```

Производные таблицы

Производная таблица - это эффективная форма представления данных, не требующая работы с самими данными. Производная таблица не является реальной таблицей и не требует постоянной памяти. Вместо этого создается и используется ″виртуальная таблица″.

Производная таблица может включать все или некоторые столбцы или строки из таблиц, на которых она определена. Например, в производной таблице

можно объединить таблицу отделов и таблицу сотрудников, чтобы было можно вывести список всех сотрудников по заданному отделу.

На рис. 2 показаны отношения между таблицами и производными таблицами.

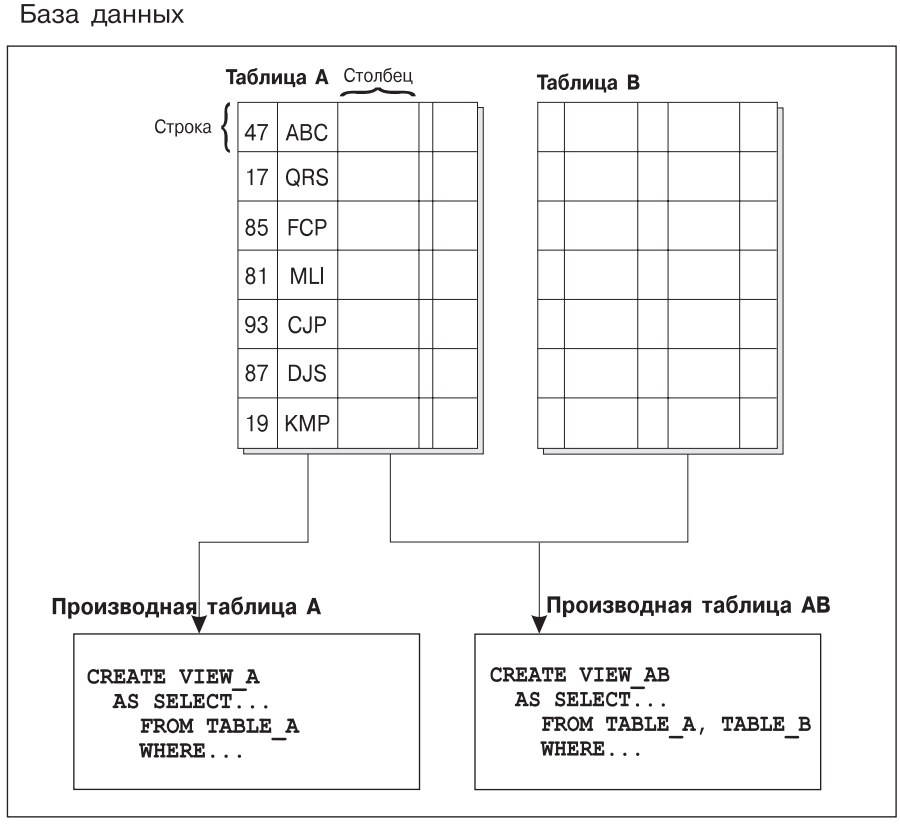


Рисунок 2. Отношение между таблицами и производными таблицами

Индекс

Индекс - это набор ключей, каждый из которых указывает на строки в таблице. Например, таблица A (на рис. 3 на стр. 13) имеет индекс по номерам сотрудников в таблице. По значению ключа можно получить указатель на строки в таблице: так, по номеру сотрудника 19 можно найти сотрудника KMP. Индекс позволяет эффективнее получать доступ к строкам в таблице, создавая прямой путь к данным через указатели.

Оптимизатор SQL автоматически выбирает наиболее эффективный путь доступа к данным в таблицах. При определении наиболее быстрого доступа к данным он принимает во внимание индексы.

Для гарантии уникальности индексного ключа могут быть созданы индексы уникальности. *Индексный ключ* - это столбец или упорядоченное собрание столбцов, на которых определяется индекс. Использование индекса уникальности гарантирует, что значение каждого индексного ключа в индексном столбце или столбцах уникально. Более подробно ключи и индексы описаны в разделе “Логические правила для данных” на стр. 24.

На рис. 3 показано отношение между индексом и таблицей.

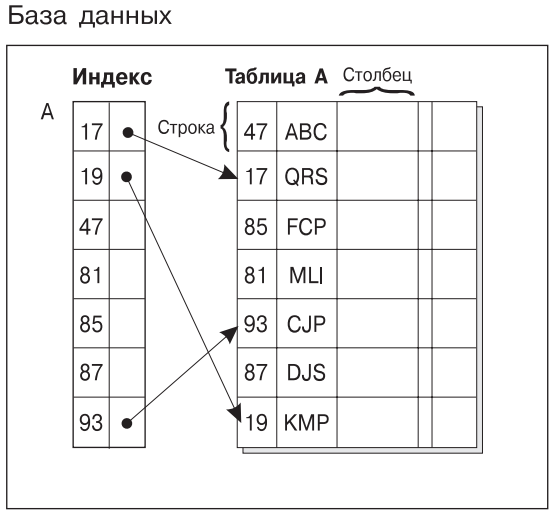


Рисунок 3. Отношение между индексом и таблицей

Схемы

Схема - это идентификатор, например ID пользователя, помогающий группировать таблицы и другие объекты баз данных. Схема может принадлежать отдельному пользователю, и ее владелец может управлять доступом к данным и объектам схемы.

Схема также является объектом баз данных. Она может быть создана автоматически, когда для нее создается первый объект. Этим объектом может быть любой объект, который можно специфицировать по имени схемы, например, таблица, индекс, производная таблица, пакет, особый тип, функция или триггер. Чтобы схема создавалась автоматически, у вас должны быть полномочия IMPLICIT_SCHEMA; можно также создать схему явно.

Имя схемы используется как первая часть двухчастного имени объекта. При создании объекта его можно назначить в заданную схему. Если не задать схему, объект назначается в схему по умолчанию; обычно это схема ID пользователя,

создавшего данный объект. Вторая часть такого имени - это собственно имя объекта. Например, у пользователя по имени Smith может быть таблица с именем SMITH.PAYROLL.

Таблицы системного каталога

Каждая база данных включает в себя набор *таблиц системного каталога*, описывающий логическую и физическую структуры данных. DB2 создает и поддерживает набор таблиц системного каталога для каждой базы данных. Эти таблицы содержат информацию об определениях объектов баз данных, например, для таблиц, производных таблиц и индексов, а также информацию о полномочиях пользователей для этих объектов. Таблицы системного каталога создаются при создании базы данных и изменяются во время нормальной работы. Их нельзя явно создать или отбросить, но можно запросить и просмотреть их содержание с помощью производных таблиц каталога.

Обзор объектов восстановления

Файлы журналов и файл хронологии восстановления создаются автоматически при создании базы данных (рис. 4 на стр. 15). Файл журнала или файл хронологии восстановления нельзя изменять непосредственно; однако они незаменимы при использовании образа резервной копии базы данных для восстановления потерянных или поврежденных данных.

Объект или понятие базы данных	Эквивалентный физический объект

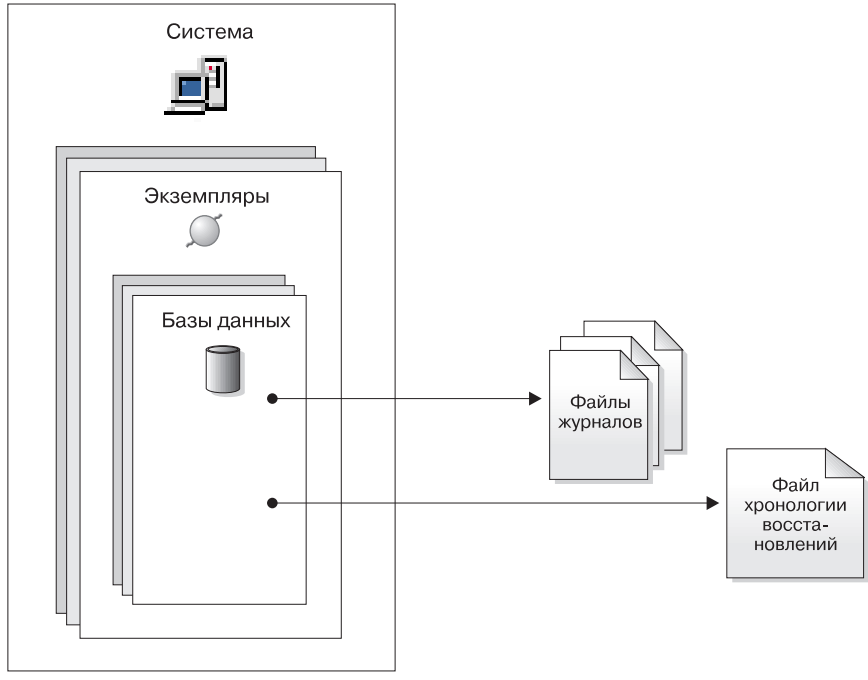


Рисунок 4. Файлы журнала и файл хронологии восстановления

Файлы журнала восстановления

Каждая база данных включает *журналы восстановления*, которые используются для восстановления информации при ошибках программ или системы. В сочетании с резервными копиями базы данных они используются для восстановления согласованности базы данных до момента, когда случилась ошибка. Восстановление базы данных обсуждается более подробно в разделе “Восстановление базы данных” на стр. 28.

Файл хронологии восстановления

Файл хронологии восстановления содержит сводную информацию о резервном копировании, которую можно использовать, когда нужно восстановить всю базу данных или ее часть до заданного момента времени. Он используется для отслеживания событий, связанных с восстановлением, например, резервного копирования и операций восстановления и загрузки. Процедура резервного копирования и восстановления базы данных описана в разделе “Восстановление базы данных” на стр. 28. Утилита загрузки описана в книге *Data Movement Utilities Guide and Reference*.

Обзор объектов хранения

Следующие объекты базы данных позволяют вам определить, как данные будут храниться в системе и как можно повысить производительность (связанную с доступом к данным):

- Табличное пространство
- Контейнер
- Пул буферов

Табличные пространства

База данных разбивается на части, которые называются *табличными пространствами*. Табличное пространство - это пространство для хранения таблиц. При создании таблицы можно решить, что определенные объекты (например, данные индексов и больших объектов) будут храниться отдельно от остальных табличных данных. Табличное пространство может быть размещено на одном или нескольких физических устройствах хранения. Следующая диаграмма показывает гибкость возможностей при распределении данных по табличным пространствам:

Рисунок 5. Табличные пространства

Табличные пространства размещаются в группах узлов (смотрите раздел “Группы узлов” на стр. 11). Определения и атрибуты табличных пространств записываются в системный каталог базы данных (смотрите раздел “Таблицы системного каталога” на стр. 14).

Для табличных пространств назначаются контейнеры. *Контейнер* - это объект физической памяти (например, файл или устройство).

Табличное пространство может управляться либо системой (SMS), либо базой данных (DMS). Каждый контейнер табличного пространства SMS является каталогом в файловом пространстве операционной системы, а пространством хранения управляет менеджер файлов данной операционной системы. Каждый контейнер табличного пространства DMS является либо предварительно выделенным файлом фиксированного размера, либо физическим устройством, например диском, а пространством хранения управляет менеджер баз данных.

На рис. 6 показано отношение между таблицами, табличными пространствами и этими двумя типами пространств. Здесь также показано, что таблицы, индексы и длинные данные хранятся в табличных пространствах.

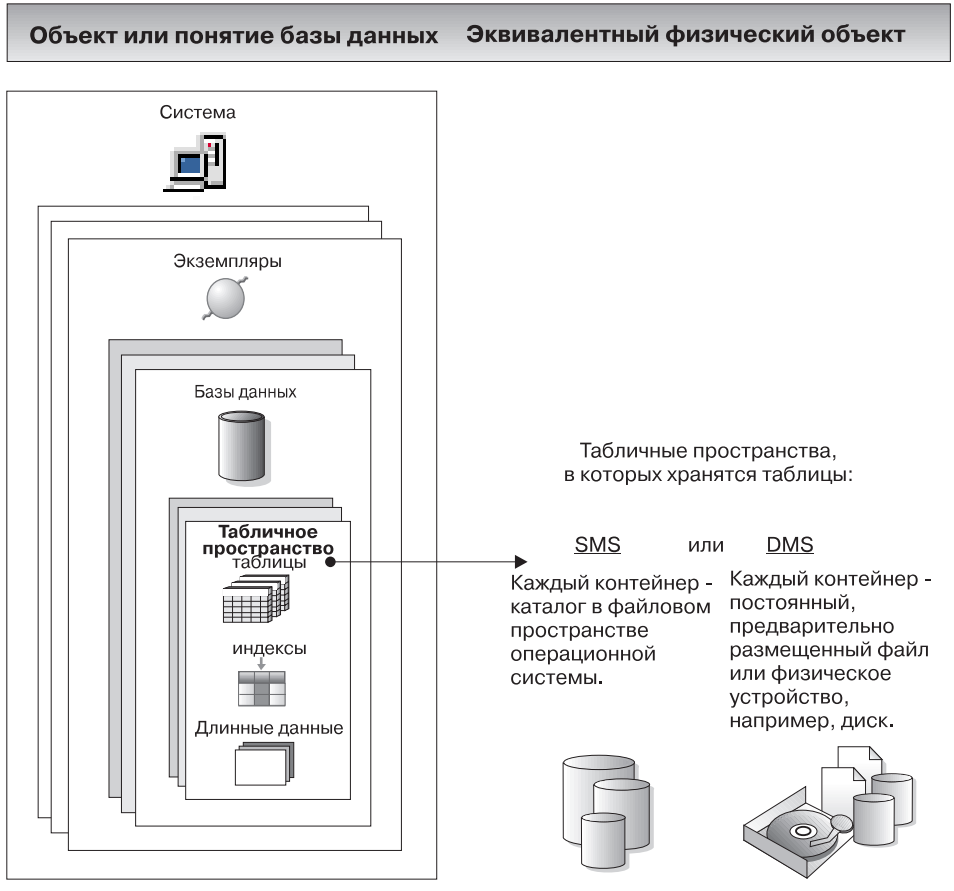


Рисунок 6. Табличные пространства и таблицы

На рис. 7 на стр. 20 показаны три типа табличных пространств: *обычное*, *временное* и *длинное*.

Таблицы, содержащие пользовательские данные, находятся в обычных табличных пространствах. Пользовательское табличное пространство по умолчанию называется USERSPACE1. Индексы также хранятся в обычных табличных пространствах. Таблицы системного каталога находятся в обычном табличном пространстве. Табличное пространство системного каталога по умолчанию называется SYSCATSPACE.

Таблицы, содержащие данные длинных полей или данные длинных объектов, например, объектов мультимедиа, размещаются в длинных табличных пространствах.

Временные табличные пространства подразделяются на системные и пользовательские. *Системные временные табличные пространства* используются для хранения внутренних временных данных, требующихся при таких операциях SQL, как сортировка, реорганизация таблиц, создание индексов и объединение таблиц. Можно создать любое число системных временных табличных пространств, но рекомендуется создать только одно с размером страниц, который чаще всего используется в ваших таблицах. Системное временное табличное пространство по умолчанию называется TEMPSPACE1. *Пользовательские временные табличные пространства* используются для хранения объявленных глобальных временных таблиц, в которых хранятся временные данные программы. Пользовательские временные табличные пространства при создании базы данных по умолчанию *не* создаются.

Рисунок 7. Три типа табличных пространств

Контейнеры

Контейнер - это физическое устройство хранения. Он может быть идентифицирован именем каталога, именем устройства или именем файла.

Контейнер назначается для табличного пространства. Одно табличное пространство может содержать несколько контейнеров, но каждый контейнер может принадлежать только одному табличному пространству.

На рис. 8 на стр. 21 показано отношение между таблицами и табличным пространством в базе данных, а также приведены связанные с ними контейнеры и диски.

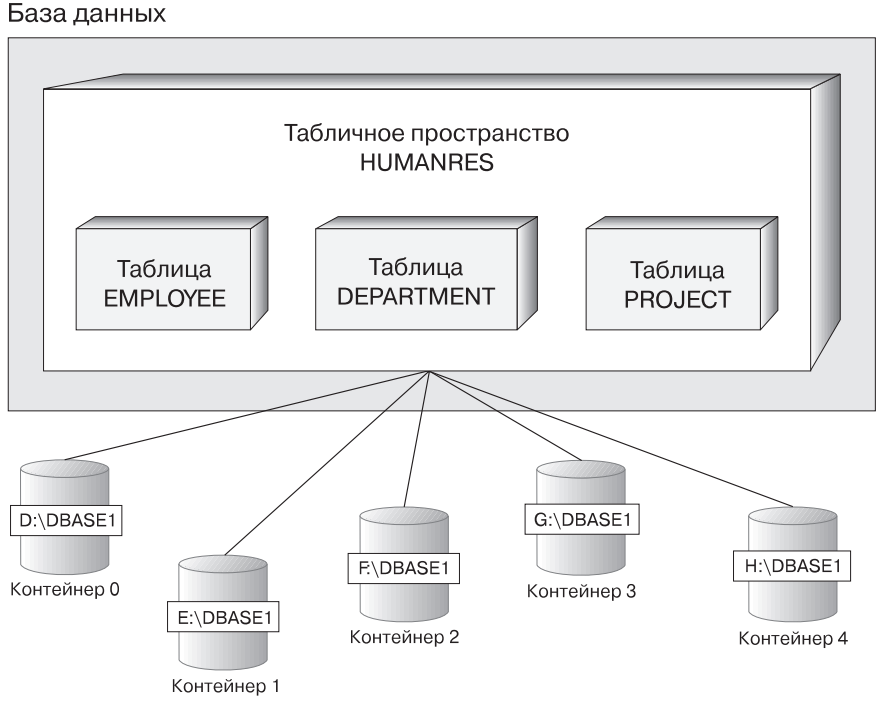


Рисунок 8. Табличные пространства и таблицы в базе данных

Таблицы EMPLOYEE, DEPARTMENT и PROJECT располагаются в табличном пространстве HUMANRES, которое занимает контейнеры 0, 1, 2, 3 и 4. В этом примере каждый контейнер находится на отдельном диске.

Данные для любой таблицы будут записываться по очереди во все контейнеры табличного пространства по принципу карусели. При таком способе хранения сохраняется баланс данных по контейнерам данного табличного пространства. Число страниц, записываемых менеджером баз данных в один контейнер перед использованием следующего, называется *размером экстента*.

Пул буферов

Пул буферов - это объем основной памяти, выделенной для кэширования страниц данных таблиц и индексов при их чтении с диска или изменении. Цель пула буферов - улучшение производительности системы. Обращаться к данным можно гораздо быстрее, если они находятся в памяти, а не на диске, поэтому чем меньше менеджеру баз данных приходится читать или записывать на диск (операции ввода/вывода), тем выше производительность. (Можно создать несколько пулов буферов, но в большинстве ситуаций требуется только один.)

Конфигурирование пула буферов - это очень важная область настройки, так как оно позволяет сократить задержку, вызванную медленными операциями ввода/вывода.

На рис. 9 показана взаимосвязь между пулом буферов и контейнерами.

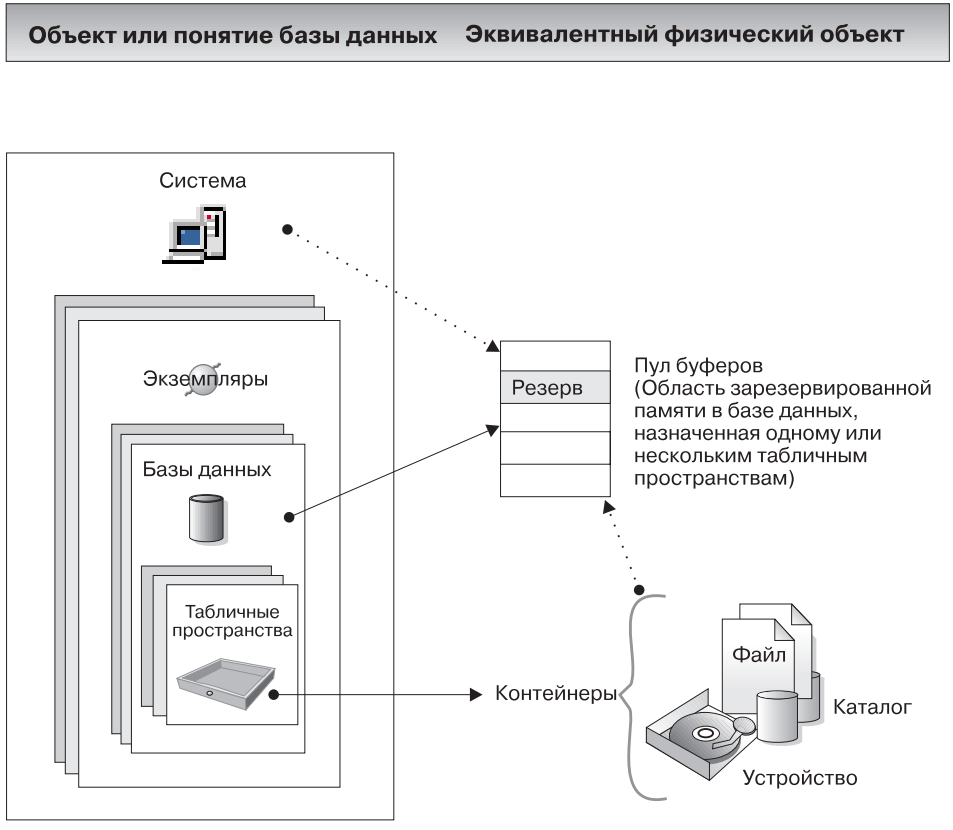


Рисунок 9. Пул буферов и контейнеры

Обзор системных объектов

При создании экземпляра DB2 или базы данных создается соответствующий файл конфигурации со значениями параметров по умолчанию. Эти значения можно изменить, чтобы улучшить производительность.

Параметры конфигурации

Файлы конфигурации содержат параметры, значения которых определяют ресурсы, выделяемые для продуктов DB2 и для отдельных баз данных, а также уровень диагностики. Существует два типа файлов конфигурации: файл

конфигурации менеджера баз данных для каждого экземпляра DB2 и файл конфигурации базы данных для каждой отдельной базы данных (смотрите рис. 10 на стр. 24).

Файл конфигурации менеджера баз данных создается при создании экземпляра DB2. Содержащиеся в нем параметры влияют на системные ресурсы на уровне экземпляра, независимо от баз данных, составляющих части этого экземпляра. Системные значения по умолчанию для многих из этих параметров можно изменить, чтобы улучшить производительность или увеличить емкость в зависимости от конфигурации данной системы.

Кроме того, по одному файлу конфигурации менеджера баз данных существует для каждой клиентской установки. В этом файле содержится информация о программе инициализации клиента для конкретной рабочей станции. Набор параметров клиента образует подмножество набора параметров, доступных для сервера.

Файл конфигурации базы данных создается при создании базы данных и размещается там же, где и сама база. Для одной базы данных существует один файл конфигурации. Помимо прочего, его параметры задают количество ресурсов, выделяемых для базы данных. Значения для многих из этих параметров можно изменить для улучшения производительности или увеличения емкости. В зависимости от рода работы, выполняемой с базой данных, могут потребоваться разные изменения.

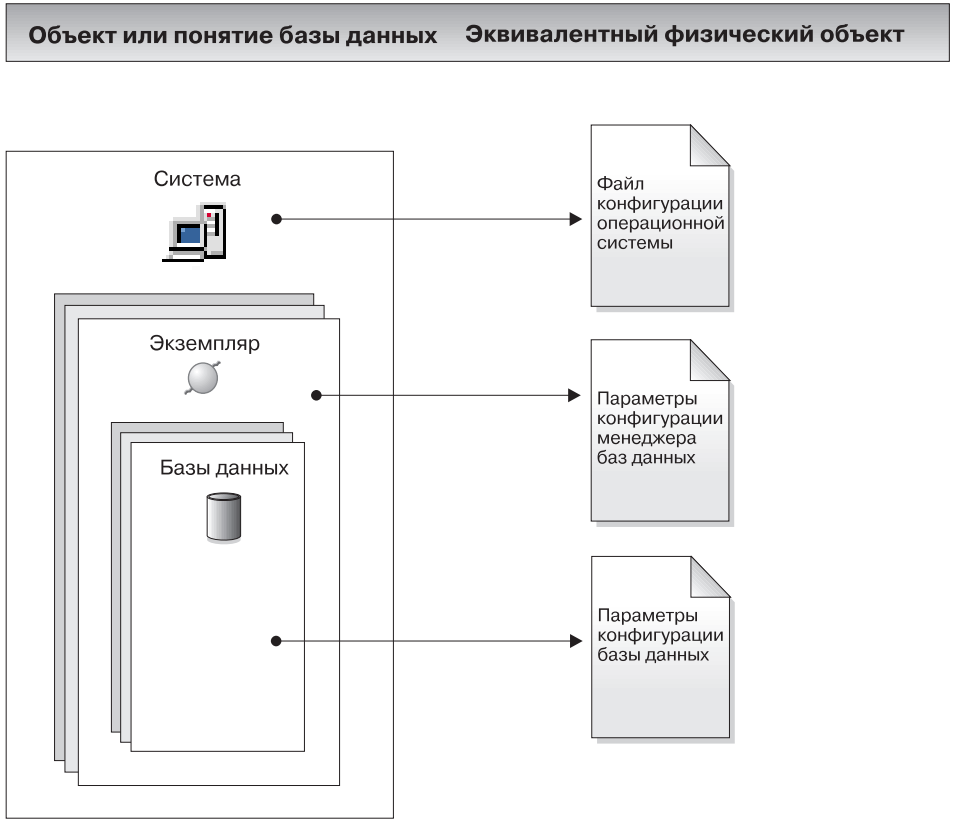


Рисунок 10. Файлы параметров конфигурации

Логические правила для данных

В любой сфере деятельности данные чаще всего должны подчиняться определенным ограничениям или правилам. Например, номера сотрудников должны быть уникальны. DB2 обеспечивает способ задания таких правил, называемый *ограничениями*.

DB2 поддерживает следующие типы ограничений:

- Ограничение NOT NULL
- Ограничение уникальности
- Ограничение первичного ключа
- Ограничение внешнего ключа
- Проверочное ограничение

ограничение NOT NULL

Ограничение NOT NULL запрещает вставлять в столбец пустые значения.

ограничение уникальности

Ограничение уникальности гарантирует, что значения в наборе столбцов уникальны и непусты для всех строк в таблице. Например, обычное ограничение уникальности в таблице DEPARTMENT может гарантировать, что номер отдела будет уникальным и непустым.

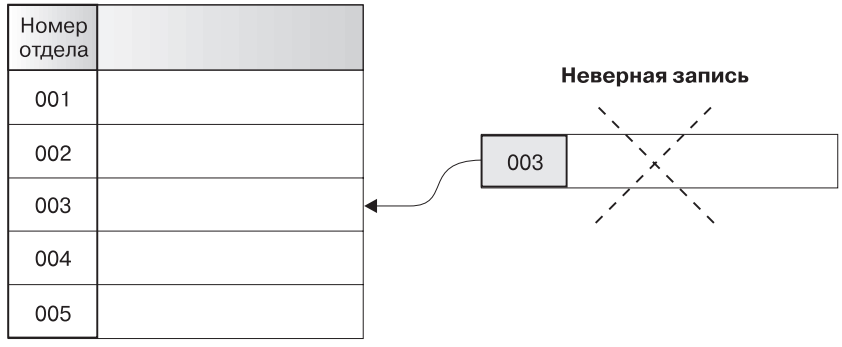


Рисунок 11. Ограничения уникальности предотвращают повторение данных

Менеджер баз данных проверяет это ограничение при операциях вставки и изменения, чтобы гарантировать целостность данных.

ограничение первичного ключа

В каждой таблице может быть только один первичный ключ. Первичный ключ - это столбец или сочетание столбцов с такими же свойствами, что и для ограничения уникальности. Ограничения первичного и внешнего ключей можно использовать для определения отношений между таблицами.

Так как первичный ключ используется для идентификации строк в таблице, он должен быть уникален и по возможности не должен меняться. У таблицы может быть только один первичный ключ, но несколько ключей уникальности. Первичный ключ необязателен; его можно определить при создании или изменении таблицы. Первичные ключи позволяют упорядочить данные при их экспорте или реорганизации.

DEPTNO и EMPNO - первичные ключи в показанных ниже таблицах DEPARTMENT и EMPLOYEE.

Таблица 1. Таблица DEPARTMENT

DEPTNO (Первичный ключ)	DEPTNAME	MGRNO
A00	Сервисный отдел компании Spiffy Computer	000010
B01	Плановый отдел	000020
C01	Информационный центр	000030
D11	Производственный отдел	000060

Таблица 2. Таблица EMPLOYEE

EMPNO (Первичный ключ)	FIRSTNAME	LASTNAME	WORKDEPT (Внешний ключ)	PHONENO
000010	Christine	Haas	A00	3978
000030	Sally	Kwan	C01	4738
000060	Irving	Stern	D11	6423
000120	Sean	O'Connell	A00	2167
000140	Heather	Nicholls	C01	1793
000170	Masatoshi	Yoshimura	D11	2890

ограничение внешнего ключа

Ограничения внешних ключей (другое название - ограничения реляционной целостности) позволяют определить требуемые отношения между таблицами и внутри них.

Например, типичным ограничением внешнего ключа можно установить, что каждый сотрудник в таблице EMPLOYEE должен работать в одном из существующих отделов, заданных в таблице DEPARTMENT.

Чтобы задать это отношение, нужно определить в качестве внешнего ключа номер отдела в таблице EMPLOYEE, а в качестве первичного ключа - номер отдела в таблице DEPARTMENT.

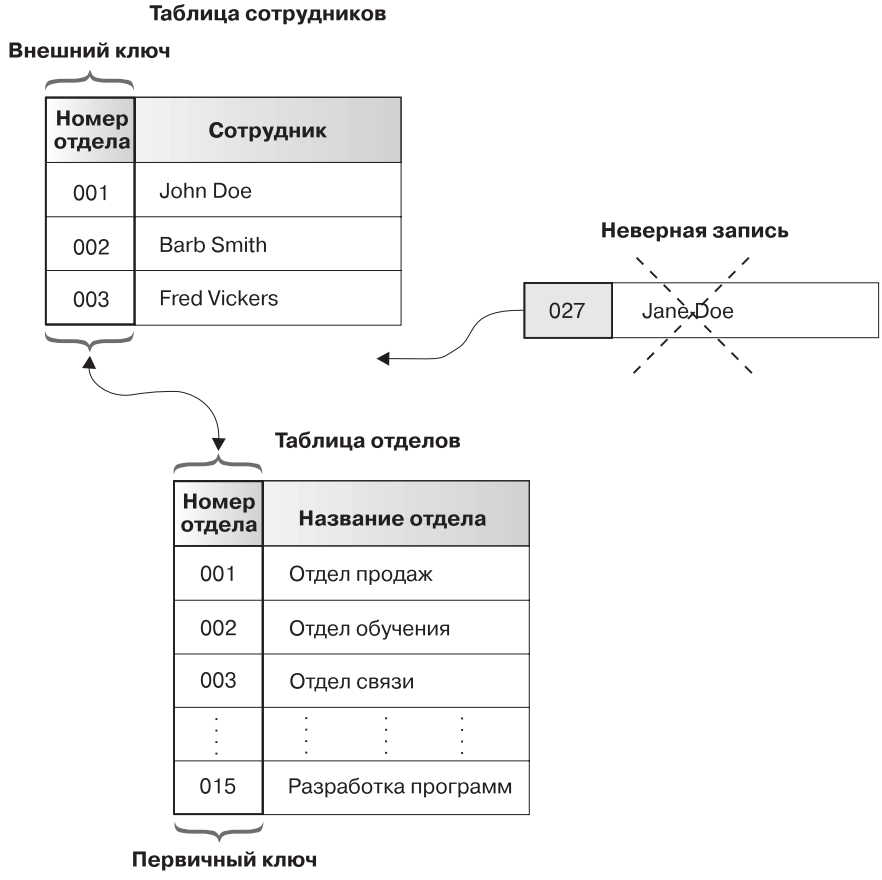


Рисунок 12. Ограничения внешнего и первичного ключей определяют отношения и защищают данные

проверочное ограничение

Проверочное ограничение - это правило базы данных, которое задает допустимость значений в одном или нескольких столбцах каждой строки таблицы.

Например, для столбца Type of Job (тип работы) в таблице EMPLOYEE можно определить допустимые значения ″Sales″, ″Manager″ и ″Clerk″. При таком ограничении все записи с другими значениями в столбце Type of Job считаются недопустимыми и будут отклоняться, то есть на тип данных, допустимых в таблице, налагается дополнительное ограничение.

В базе данных можно также использовать *триггеры*. По сравнению с ограничениями триггеры - более сложное и потенциально более мощное средство. Они определяют набор действий, выполняемых в сочетании с

операциями INSERT, UPDATE или DELETE или запускаемых этими операциями для заданной базовой таблицы. Триггеры можно использовать для поддержки общих требований целостности или выполнения логических правил. Например, триггер может проверять лимит кредита клиента перед принятием заказа или поднять тревогу в банковской программе, если снятие со счета не соответствует стандартным образцам снятия со счетов клиентами. Дополнительную информацию о триггерах смотрите в книге *Application Development Guide*.

Восстановление базы данных

База данных может оказаться непригодной для использования из-за ошибки аппаратных средств, программного обеспечения или общей ошибки; при различных сценариях ошибок могут потребоваться разные действия по восстановлению. Для защиты базы данных от возможных сбоев должна существовать продуманная стратегия.

В этом разделе обсуждаются различные методы восстановления и показано, как определить, какой метод восстановления лучше подходит для вашей конкретной среды. Рассматриваются следующие темы:

Обзор восстановления

При возникновении ошибок в базе данных нужно знать, какие возможности восстановления вам доступны. К ошибкам баз данных относятся ошибки носителей и памяти, сбои питания и ошибки программ. Можно выполнить резервное копирование базы данных или отдельных табличных пространств, а затем восстановить их, если они будут как-либо повреждены или испорчены. Идея *резервного копирования* базы данных - та же, что и для любых других данных: копия данных снимается и сохраняется на другом носителе на случай ошибки или повреждения оригинала. В простейшем случае резервного копирования работу с базой данных прекращают для гарантии, чтобы при копировании не выполнялись последующие транзакции, а затем просто создают ее резервную копию.

Воссоздание базы данных называется *восстановлением*. Возможность *восстановления после сбоя* позволяет автоматически восстановить базу данных после ошибки, если это возможно. При повреждении базы данных существует два способа ее восстановления: *восстановление версии* и *восстановление с повтором транзакций*.

У невосстановимых баз данных оба параметра конфигурации, *logretain* и *userexit*, отключены. Это значит, что хранятся только журналы, требующиеся для восстановления после сбоя. Эти журналы называются *активными журналами*; они содержат данные текущих транзакций. Восстановление версии при помощи *автономных* резервных копий - основное средство восстановления невосстановимой базы данных. (Автономность резервной копии означает, что во время резервного копирования другие программы не используют базу данных.) Такую базу данных можно восстановить только в автономном режиме. Для нее восстанавливается состояние, предшествующее созданию образа резервной копии.

В восстановимой базе данных параметр конфигурации *logretain* имеет значение ″RECOVERY″ или включен параметр конфигурации *userexit*, либо выполнены оба эти условия. Активные журналы для восстановления после сбоя по-прежнему доступны, но кроме того, появляются *архивные журналы* с данными принятых транзакций. Такая база данных может быть восстановлена только в автономном режиме. Она восстанавливается в состояние, предшествующее созданию образа ее резервной копии. Однако при восстановлении с повтором транзакций можно выполнить *повтор транзакций* (то есть пройти дальше момента создания образа резервной копии), используя активные и архивные журналы либо до заданного момента времени, либо до окончания активных журналов.

Операции резервного копирования восстановимой базы данных можно выполнять либо в автономном, либо в *оперативном* режиме (оперативный режим означает, что во время выполнения резервного копирования с базой данных могут соединяться другие программы). Операции восстановления базы данных и повтора транзакций должны всегда выполняться в автономном режиме. При резервном копировании в оперативном режиме восстановление с повтором транзакций гарантирует, что *все* изменения таблиц захватываются и будут снова применены при восстановлении такой резервной копии.

Если у вас восстановимая база данных, резервное копирование и повтор транзакций можно выполнять не для всей базы данных, а для отдельных табличных пространств. Во время выполнения резервного копирования табличного пространства в оперативном режиме оно доступно для использования, при этом изменения одновременно записываются в журналы. При выполнении в оперативном режиме операций восстановления или повтора транзакций само табличное пространство недоступно для использования до окончания операции, но пользователям не запрещен доступ к таблицам в других табличных пространствах.

Восстановление после сбоев защищает базу данных от возможности остаться в несогласованном или непригодном к использованию состоянии. Транзакции (или единицы работы), выполняемые на базе данных, могут быть неожиданно прерваны. Если ошибка произойдет прежде, чем будут завершены и приняты все

изменения, являющиеся частью единицы работы, база данных останется в несогласованном и непригодном к использованию состоянии.

В таком случае базу данных нужно перевести в согласованное или пригодное состояние. Это достигается откатом незавершенных транзакций и завершением принятых транзакций, которые еще оставались в памяти к моменту сбоя (смотрите рис. 13).
Когда база данных находится в согласованном и пригодном к использованию

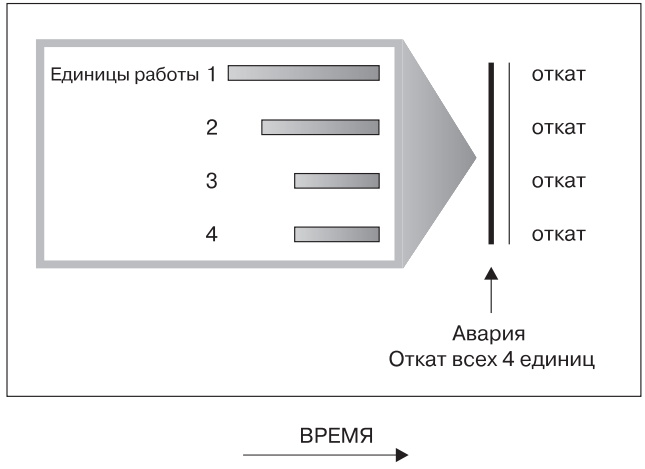


Рисунок 13. Откат единиц работы

состоянии, говорят, что она достигла так называемой ″точки согласованности″. Автономная резервная копия базы данных снимается в точке согласованности. Достижение точки согласованности означает, что все транзакции завершены (принятием или откатом) и данные доступны другим пользователям или программам.

После сбоя перейти к точке согласованности можно, введя команду RESTART DATABASE (смотрите книгу *Command Reference*). Если вы хотите автоматически выполнять такой переход после каждой ошибки, обдумайте использование параметра конфигурации **разрешение автоматического перезапуска** (*autorestart*). По умолчанию этот параметр конфигурации базы данных вызывает команду RESTART DATABASE, когда бы она ни потребовалась. Если параметр *autorestart* включен, вызов команды RESTART DATABASE происходит при первом после ошибки требовании соединения с базой данных.

Восстановление после сбоя переводит базу данных в согласованное и пригодное к использованию состояние. Но, если восстановление после сбоя применяется к базе данных, для которой доступно восстановление с повтором (то есть параметр конфигурации *logretain* имеет значение ″RECOVERY″, или включен параметр конфигурации *userexit*) и во время восстановления после сбоя

происходит ошибка, касающаяся отдельного табличного пространства, это табличное пространство переводится в автономный режим, и становится недоступным до окончания восстановления. Восстановление после сбоя продолжается. По завершении восстановления после сбоя остальные табличные пространства в этой базе данных будут пригодными для использования, и с базой данных могут быть установлены соединения. (Есть исключения, касающиеся табличных пространств с временными таблицами или таблицами системного каталога. Они обсуждаются при описании восстановления с повтором транзакций.)

Как отмечено ранее, DB2 поддерживает два метода восстановления поврежденной базы данных:

- *Восстановление версии* - это восстановление предыдущей версии базы данных с использованием образа, созданного при резервном копировании.

  Эта операция восстановления воссоздает всю базу данных с использованием резервной копии, сделанной ранее. Резервная копия базы данных позволяет восстановить базу данных до состояния на момент создания данной резервной копии. Все единицы работы с момента резервного копирования до момента ошибки теряются (смотрите рис. 14 на стр. 32).

  При использовании метода восстановления версии вы должны запланировать и регулярно выполнять снятие полных резервных копий базы данных.

  В среде многораздельной базы данных база данных размещена на нескольких серверах разделов баз данных (или узлах). Восстанавливать надо все разделы, причем образы резервных копий, которые будут использоваться при восстановлении базы данных, должны быть созданы в одно и то же время. (Резервное копирование и восстановление каждого раздела базы данных выполняется по отдельности.) Резервное копирование всех разделов базы данных, выполняемое одновременно, называется *резервным копированием версии*.

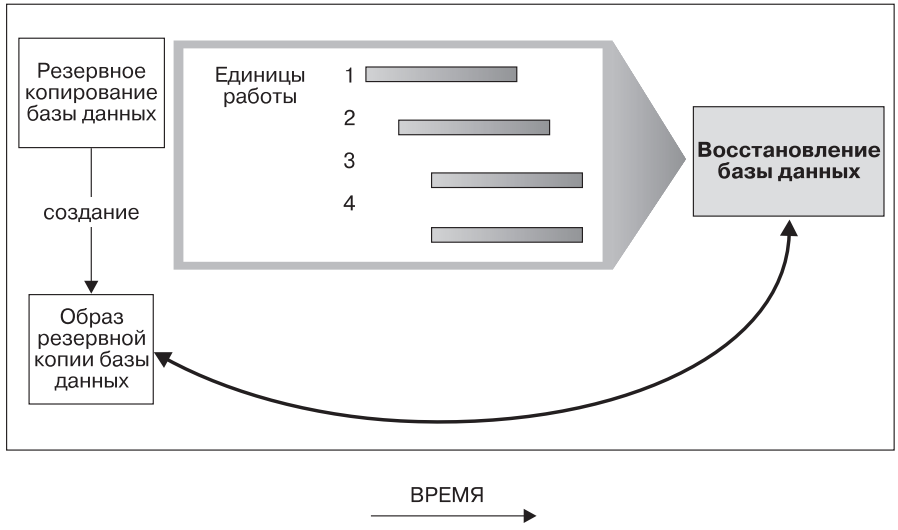


Рисунок 14. Восстановление базы данных

- Для использования метода *восстановления с повтором транзакций* у вас должны быть готовая резервная копия базы данных и архивированные журналы. (Это достигается заданием параметра конфигурации базы данных *logretain* или *userexit*. Информацию о выборе вариантов процедуры записи в журналы смотрите в разделе “Журналы базы данных” на стр. 36.) Восстановление базы данных с заданной опцией WITHOUT ROLLING FORWARD эквивалентно использованию метода восстановления версии. База данных восстанавливается до состояния на момент создания образа автономной резервной копии. Если восстановить базу данных, *не* задав опцию WITHOUT ROLLING FORWARD, база данных окажется в состоянии отложенного повтора транзакций. Это состояние допускает выполнение повтора транзакций.

  Рассмотрим два типа восстановления с повтором транзакций:

  – *Восстановление базы данных с повтором транзакций*. При этом типе восстановления после операции восстановления базы данных применяются записи транзакций журналов базы данных (смотрите рис. 15 на стр. 33). В журналы базы данных записываются все изменения, сделанные в базе данных. Этот метод используется для восстановления базы данных до состояния на определенный момент времени или непосредственно предшествующего ошибке (то есть до окончания активных журналов.)

    В среде многораздельной базы данных база данных распределена по нескольким разделам. При восстановлении с повтором транзакций до момента времени повтор транзакций нужно выполнить для всех разделов базы данных, чтобы все разделы находились на одном уровне. Если нужно восстановить один раздел базы данных, чтобы привести его к уровню

остальных разделов этой базы данных, можно выполнить восстановление с повтором транзакций до окончания журналов.

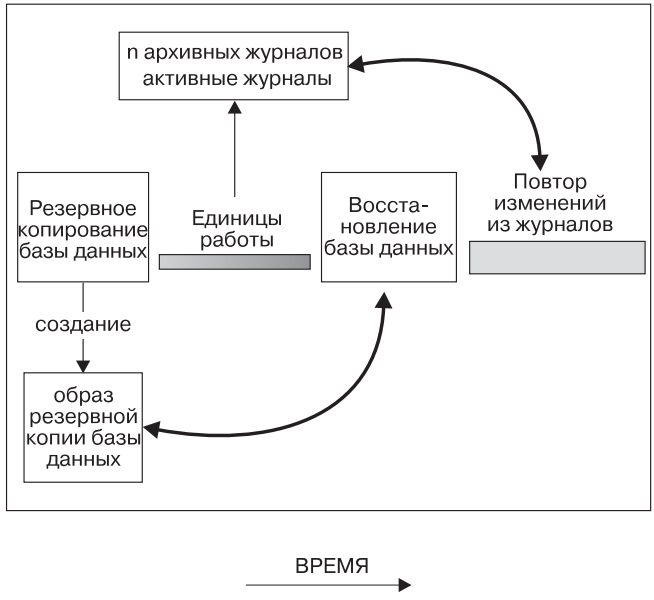


Рисунок 15. Восстановление базы данных с повтором транзакций

– *Восстановление и повтор транзакций табличного пространства*. Если восстановление с повтором транзакций поддерживается базой данных, для ее табличных пространств также возможно резервное копирование, восстановление и повтор транзакций. Для выполнения восстановления и повтора транзакций табличного пространства требуются образ резервной копии всей базы данных (то есть всех табличных пространств) или резервные копии одного или нескольких восстанавливаемых табличных пространств. Кроме того, нужны записи журналов, относящиеся к восстанавливаемым табличным пространствам. Повтор транзакций можно выполнить до одной из двух точек:
  - До окончания журналов
  - До заданной точки во времени (такое восстановление называется восстановлением *до момента времени*).

Примечания:

1. Табличные пространства, не выбранные во время операции резервного копирования, окажутся в состоянии, отличном от состояния табличных пространств, которые были восстановлены.
2. При использовании метода восстановления с повтором транзакций для табличных пространств требуется идентифицировать ″ключевые″ табличные пространства в восстанавливаемой базе данных, а также

запланировать и регулярно выполнять резервное копирование базы данных (или ″ключевых″ табличных пространств).

Восстановление с повтором транзакций для табличных пространств может использоваться в двух случаях:

– После операции восстановления табличного пространства оно всегда остается в состоянии отложенного повтора транзакций и для него требуется выполнить повтор транзакций. Чтобы выполнить повтор транзакций для табличных пространств с использованием журналов либо до указанного момента времени, либо до окончания журналов, введите команду ROLLFORWARD DATABASE (смотрите книгу *Command Reference*).
– Если по завершении восстановления после сбоя одно или несколько табличных пространств оказываются в состоянии *отложенного повтора транзакций*, сначала исправьте в этих табличных пространствах ошибки. В некоторых случаях исправление ошибки для этих табличных пространств не требует привлечения операции восстановления базы данных. Например, табличное пространство может оказаться в состоянии отложенного повтора транзакций при отключении питания. Если такая ошибка будет исправлена перед восстановлением после сбоя, восстановления после сбоя может оказаться достаточно для перевода базы данных в согласованное, пригодное к использованию состояние. Восстановления базы данных в этом случае не потребуется. После исправления ошибок табличного пространства можно воспользоваться командой ROLLFORWARD DATABASE, чтобы выполнить повтор транзакций для табличных пространств с использованием журналов либо до указанного момента времени, либо до окончания журналов.

  Примечание: Если табличное пространство с ошибкой содержит таблицы системного каталога, базу данных запустить не удастся. Нужно будет восстановить табличное пространство SYSCATSPACE, а затем выполнить повтор транзакций до окончания журналов.

В среде многораздельной базы данных при повторе транзакций для табличного пространства *до указанного момента времени* список узлов (разделов базы данных), на которых расположено данное табличное пространство, не потребуется. DB2 передает требование повтора транзакций всем разделам. Это значит, что табличное пространство должно быть восстановлено на всех разделах, где оно находится.

В среде многораздельной базы данных при повторе транзакций табличного пространства *до окончания журналов*, если вы хотите выполнить повтор транзакций табличного пространства *не* на всех разделах, нужно задать список разделов базы данных. При выполнении повтора транзакций всех

табличных пространств по всем разделам, которые находятся в состоянии отложенного повтора транзакций, список разделов базы данных не требуется. По умолчанию требование повтора транзакций базы данных посылается всем разделам.

Факторы, влияющие на восстановление

Чтобы решить, какой метод восстановления базы данных использовать, нужно рассмотреть следующие ключевые вопросы:

- Является база данных восстанавливаемой или невосстанавливаемой?
- До какого момента времени относительно момента ошибки нужно восстановить базу данных (точка восстановления)?
- Сколько времени можно потратить на восстановление базы данных? Надо учесть:
  - Время между операциями резервного копирования (которое влияет на восстановление с повтором транзакций)
  - Время, которое база данных находится в пригодном для использования или доступном состоянии (определяет выбор оперативного или автономного резервного копирования, исходя из требований доступности данных)
- Сколько места можно выделить для резервных копий и архивных журналов?
- Будут использоваться резервные копии уровня табличного пространства или уровня всей базы данных?

В целом поддержка базы данных и стратегия восстановления должны гарантировать, что вся информация будет доступной, когда она потребуется для восстановления базы данных. Такая стратегия должна включать регулярное резервное копирование по расписанию, а также плановые резервные копирования во время создания базы данных или (для системы многораздельных баз данных) в случае изменения размера системы при добавлении или отбрасывании серверов (узлов) разделов базы данных. Помимо этих базовых требований хорошая стратегия включает элементы, снижающие вероятность и влияние ошибок базы данных.

Дополнительную информацию смотрите в следующих темах:

- “Восстановимые и невосстановимые базы данных” на стр. 36
- “Журналы базы данных” на стр. 36
- “Сокращение записи в журнал для рабочих таблиц” на стр. 42
- “Точка восстановления” на стр. 43
- “Частота резервного копирования и требуемое время” на стр. 44
- “Требуемое время восстановления” на стр. 46
- “Требования к памяти” на стр. 46
- “Сохранение связей данных” на стр. 47

- “Ограничения по использованию различных операционных систем” на стр. 48
- “Восстановление поврежденного табличного пространства” на стр. 48
- “Вопросы производительности восстановления” на стр. 50.

В этом разделе основное внимание уделяется восстановлению базы данных, но в полном плане восстановления восстанавливаться должны также:

- Операционная система и выполняемые программы DB2
- Коды программ, пользовательских функций и хранимых процедур в библиотеках операционной системы
- Команды создания экземпляров DB2 и прочих (не входящих в DB2) ресурсов
- Защита операционной системы
- Копии загрузки для операции загрузки (если в команде LOAD задается COPY YES)

Восстановимые и невосстановимые базы данных

Если вы можете легко воссоздать данные, база данных, где они хранятся, может быть невосстановимой. Например:

- Таблицы, хранящие данные из внешнего источника, который используется для программ только для чтения (и эти данные не смешиваются с существующими данными); для них следует рассмотреть использование невосстановимой базы данных.
- Таблицы с небольшим количеством данных. Здесь восстановление не является проблемой. Скорее всего, небольшой объем данных журналов просто не оправдает дополнительную сложность управления файлами журналов и выполнения повтора транзакций после восстановления.
- Большие таблицы, куда периодически добавляется небольшое число строк. Как и выше, небольшой объем изменений не оправдывает затраты на управление файлами журналов и на повтор транзакций после операции восстановления.

Если нельзя легко воссоздать данные, базу данных, где они хранятся, надо сделать восстановимой базой данных. Приведем примеры данных, для которых надо использовать восстановимые базы данных:

- Данные, которые вы не можете воссоздать. К ним относятся данные, источник которых удаляется после их загрузки, и данные, которые вводятся в таблицы вручную.
- Данные, изменяемые после загрузки в базу данных программами и пользователями рабочих станций.

Журналы базы данных

У каждой базы данных есть связанные с ней журналы. В этих журналах хранятся записи изменений базы данных. Если базу данных нужно восстановить до точки

позже времени последнего полного автономного резервного копирования, требуются журналы для повтора транзакций вплоть до момента ошибки.

В DB2 существует два типа регистрации в журналах: *циклическая* и *архивная*, каждая из которых обеспечивает свой уровень возможностей восстановления.

Циклическая запись выбирается по умолчанию, когда создается новая база данных. Для этого типа регистрации подходят только полные автономные резервные копии. Из названия понятно, что для восстановления с целью устранения ошибок транзакций и сбоев системы при циклической записи используется ″кольцо″ журналов. Эти журналы используются и сохраняются только до точки, гарантирующей целостность текущих транзакций. Циклическая запись не позволяет выполнять повтор транзакций базы данных для транзакций, предшествующих последней полной резервной копии. При восстановлении в случае ошибок и при авариях носителя используется полная автономная резервная копия. Все изменения со времени последней резервной копии теряются. При выполнении резервного копирования база данных должна находиться в автономном состоянии (недоступном для пользователей). Так как этот тип восстановления восстанавливает данные до конкретной точки времени создания полной резервной копии, он называется *восстановлением версии*.

На рис. 16 показано, что, когда активна циклическая запись, активный журнал использует файлы журналов в циклическом порядке.
Активные журналы используются во время восстановления после сбоя, чтобы не

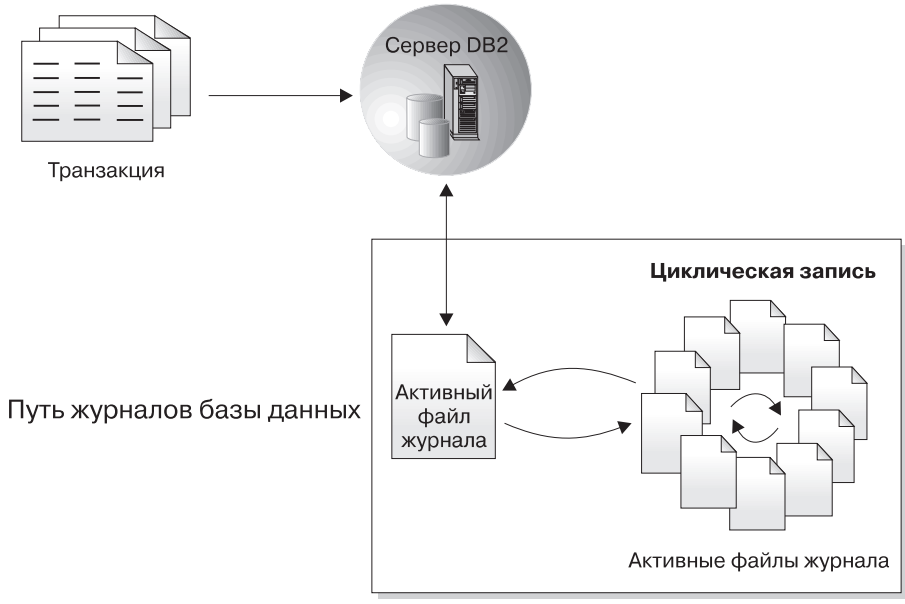


Рисунок 16. Циклическая запись журналов

допустить ситуации, когда база данных после сбоя останется в несогласованном состоянии. Команда RESTART DATABASE использует активные журналы, когда базу данных нужно перевести в согласованное и пригодное к использованию состояние. Во время восстановления после сбоя для изменений, записанных в журналы, но не принятых из-за сбоя, выполняется откат. Изменения, которые были приняты, но физически не были записаны из памяти (пула буферов) на диск (в контейнер базы данных), повторяются. Эти действия гарантируют целостность базы данных. При необходимости активные журналы может также использовать команда ROLLFORWARD DATABASE при восстановлении до указанного момента времени или до окончания журналов. Активные журналы находятся в каталоге пути журналов базы данных.

Архивные журналы используются специально для восстановления с повтором транзакций. Это могут быть:

оперативные архивные журналы
Когда изменения в активном журнале больше не требуются для обычного восстановления, такой журнал закрывается и становится архивным. Архивный журнал считается *оперативным*, если он хранится в каталоге пути журналов базы данных (смотрите рис. 17 на стр. 39).

автономные архивные журналы

Архивный журнал считается *автономным*, если он больше не находится в каталоге пути журналов базы данных (смотрите рис. 18 на стр. 40). Если используется обработчик пользователя, архивные журналы можно хранить и не в каталоге пути журналов базы данных. (Дополнительную информацию смотрите в теме ″Обработчик пользователя для восстановления базы данных″ книги *Руководство администратора: Реализация*.)

Рисунок 17. Архивные журналы

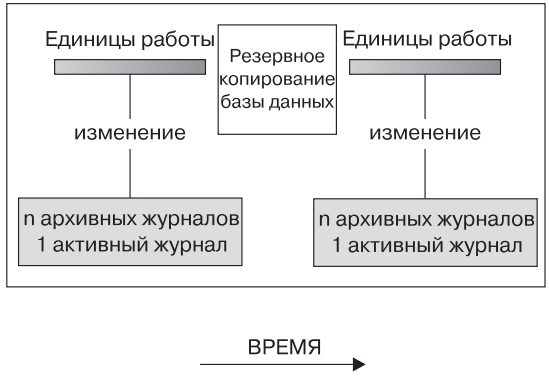
Рисунок 18. Автономные архивные журналы

При восстановлении с повтором транзакций для воссоздания базы данных до окончания журналов или до указанного момента времени могут использоваться и архивные, и активные журналы. Функция повтора транзакций восстанавливает базу, применяя повторно принятые изменения из архивных и активных журналов.

При восстановлении с повтором транзакций журналы могут также использоваться для воссоздания табличного пространства с помощью повторного применения принятых изменений как из архивных, так и из активных журналов. Табличное пространство можно восстановить до окончания журналов или до указанного момента времени.

Во время резервного копирования без отключения все действия над базой данных регистрируются. При восстановлении с использованием оперативной резервной копии должен быть выполнен повтор транзакций журналов, по крайней мере, до момента завершения создания этой резервной копии. Чтобы сделать это, надо заархивировать журналы и сделать их доступными на время восстановления базы данных. Используемый во время резервного копирования

файл журнала может остаться открытым, когда резервное копирование завершится. Опция FLUSH LOG для оперативного резервного копирования в команде BACKUP DATABASE закрывает активный журнал по завершении оперативного резервного копирования. Это позволит сделать активный журнал архивным, чтобы у вас была полная резервная копия и все журналы, требующиеся для ее восстановления.



Между моментами снятия резервных копий для отслеживания изменений в базах данных используются журналы.

Рисунок 19. Активные и архивные журналы базы данных при восстановлении с повтором транзакций

Изменить место, откуда будут браться архивные журналы для восстановления, позволяют два параметра конфигурации базы данных: *newlogpath* и *userexit*. Изменение параметра *newlogpath* влияет также и на место сохранения активных журналов. Дополнительную информацию об этих параметрах конфигурации смотрите в книге *Руководство администратора: Производительность*.

Чтобы определить, какие *экстенты* журналов (смотрите разделе “Контейнеры” на стр. 20) в каталоге пути журналов базы данных являются архивными, проверьте значение параметра конфигурации базы данных *loghead*. Этот параметр указывает журнал с наименьшим номером, который является активным. Журналы с последовательными номерами меньше, чем *loghead* - архивные, и их можно переместить. Значение этого параметра можно проверить, вызвав Центр управления или введя в командной строке команду GET DATABASE CONFIGURATION и посмотрев номер первого активного файла журнала. Дополнительную информацию об этом параметре конфигурации смотрите в книге *Руководство администратора: Производительность*.

Примечания:

1. Если вы сотрете активный журнал, база данных станет непригодной и, чтобы ее снова можно было использовать, нужно будет выполнить восстановление. Повтор транзакций можно будет выполнить только для того журнала, который был стерт.
2. Если вы боитесь, что активные журналы могут быть повреждены (например, в результате сбоя диска), обдумайте возможность создания зеркальной копии томов, на которых хранятся эти журналы.

Сокращение записи в журнал для рабочих таблиц

Если ваша программа создает и заполняет рабочие таблицы из основных таблиц и восстановимость этих рабочих таблиц не требуется, так как, используя основные таблицы, их можно легко воссоздать, при создании рабочих таблиц можно задать в операторе CREATE TABLE параметр NOT LOGGED INITIALLY. Преимущество использования параметра NOT LOGGED INITIALLY состоит в том, что никакие изменения, сделанные в этой таблице (включая операции вставки, удаления, изменения или создания индексов), в единице работы, создающей эту таблицу, регистрироваться не будут. Это не только сокращает запись в журналы, но может также увеличить производительность программы. Такого же результата можно добиться для существующих таблиц, задав параметр NOT LOGGED INITIALLY для оператора ALTER TABLE.

Примечания:

1. В одной единице работы с параметром NOT LOGGED INITIALLY можно создать несколько таблиц.
2. Изменения для таблиц каталога и других пользовательских таблиц по-прежнему будут регистрироваться.

Так как при использовании параметра NOT LOGGED INITIALLY изменения для таблицы не регистрируются, следует учесть следующее:

- *Все* изменения для таблицы должны быть сброшены на диск во время принятия. Это означает, что принятие может занять больше времени.
- Ошибка, возвращенная любой операцией в единице работы, где создается таблица, приведет к откату всей единицы работы (SQLCODE -1476, SQLSTATE 40506).
- При повторе транзакций такую таблицу восстановить невозможно. Если при повторе транзакций встречается таблица, созданная с опцией NOT LOGGED INITIALLY, она помечается как недоступная. После восстановления базы данных при любой попытке обращения к такой таблице возвращается код SQL1477N.

  Примечание: При создании таблицы, пока не будет выполнено принятие, для таблиц каталога удерживаются блокировки строк. Чтобы получить преимущество от отмены записи в журнал, таблицу следует заполнять в той единице работы, где она создается. Это

требуется для параллелизма. Дополнительную информацию о параллелизме смотрите в книге *Руководство администратора: Производительность*.

Дополнительную информацию о создании таблиц смотрите в справочнике *SQL Reference*.

Если в качестве рабочих таблиц планируется использовать объявленные временные таблицы, обратите внимание на следующее:

- Объявленные временные таблицы не создаются в системных каталогах, следовательно, блокировки не удерживаются.
- Для объявленных временных таблиц запись в журнал не выполняется даже после первого принятия.
- Для хранения строк в таблице после принятия используйте опцию ON COMMIT PRESERVE, иначе все строки будут удалены.
- К экземпляру таблицы может обратиться только программа, создавшая объявленную временную таблицу.
- Таблица неявно отбрасывается при отбрасывании соединения программы с базой данных.
- Ошибки операции во время использования единицей работы объявленной временной таблицы не приводят к полному откату единицы работы. Однако ошибка операции в операторе, изменяющем содержимое объявленной временной таблицы, приведет к удалению всех строк этой таблицы. Откат единицы работы (или точки сохранения) приведет к удалению всех строк в объявленных временных таблицах, которые были изменены в данной единице работы (или точке сохранения).

Подробнее об объявленных временных таблицах и их ограничениях можно узнать, посмотрев информацию об операторе DECLARE GLOBAL TEMPORARY TABLE в справочнике *SQL Reference*.

Точка восстановления

Методы восстановления версии и восстановления с повтором транзакций позволяют дойти до различных точек восстановления. Метод восстановления версии включает в себя создание автономной полной резервной копии для базы данных в запланированные моменты времени. При этом методе восстановленная база данных будет соответствовать моменту создания резервной копии, использовавшейся для ее восстановления. Например, если создавать резервную копию в конце каждого дня и потерять базу данных в середине дня, будут потеряны изменения за первую половину дня.

В методе восстановления с повтором транзакций изменения, внесенные в базу данных, сохраняются в журналах. В этом методе база данных или табличное пространство сначала восстанавливается с помощью резервной копии, а затем

для повторного внесения изменений, сделанных в базе данных со времени создания резервной копии, используются журналы.

При включенной поддержке восстановления с повтором транзакций можно использовать преимущества оперативного резервного копирования и резервного копирования уровня табличного пространства. Для полного восстановления базы данных и табличного пространства с повтором транзакций можно выбрать восстановление до окончания журналов или до указанного момента времени. Например, если базу данных повредила некоторая программа, можно запустить восстановленную копию базы данных и выполнить повтор транзакций до момента времени, предшествующего запуску этой программы. Никакие единицы работы, записанные в журналы после указанного момента времени, не применяются.

Повтор транзакций до окончания журналов или до указанного момента времени можно выполнять и для табличных пространств.

Частота резервного копирования и требуемое время

План восстановления должен предполагать регулярное создание резервных копий, поскольку резервное копирование базы данных требует времени и системных ресурсов.

Следует регулярно создавать полные резервные копии базы данных, даже если вы архивируете журналы (нужные для восстановления с повтором транзакций). Если в стратегию входит восстановление с повтором транзакций, использование более новой резервной копии означает, что применяется меньше архивных журналов, что уменьшает время, затрачиваемое утилитой ROLLFORWARD на восстановление базы данных.

Кроме того, лучше не перезаписывать резервные копии и журналы, а сохранять из дополнительной предосторожности несколько полных резервных копий и связанных с ними журналов.

Если главный фактор - время, необходимое на применение архивных журналов при восстановлении или повторе транзакций, рассмотрите возможные затраты на более частое резервное копирование. При этом сократится число архивных журналов, необходимых для повтора транзакций.

Резервное копирование можно выполнять как при *подключенной*, так и при *отключенной* базе данных. При резервном копировании без отключения базы данных другие программы и процессы могут не разрывать с ней соединение и не прекращать чтения и изменения данных во время копирования. При отключенной базе данных на протяжении резервного копирования никто не сможет ей воспользоваться.

Для сокращения времени, которое база данных остается недоступной, используется оперативное резервное копирование. Оперативное резервное копирование поддерживается, только если включена поддержка восстановления с повтором транзакций. При поддержке восстановления с повтором транзакций и наличии полного набора журналов базу данных можно восстановить, когда бы в этом ни возникла потребность.

Примечания:

1. Оперативную резервную копию можно использовать только при наличии журнала (или журналов) базы данных, покрывающих время создания данной резервной копии.
2. Автономное резервное копирование быстрее оперативного.

Если база данных содержит много данных длинных полей и больших объектов, резервное копирование может отнять очень много времени. Команда BACKUP позволяет выполнять резервное копирование выбранных табличных пространств. При использовании табличных пространств DMS различные типы данных можно хранить в своих собственных табличных пространствах, что сокращает время, требующееся на операцию резервного копирования. Табличные данные можно хранить в одном табличном пространстве, данные длинных полей и больших объектов - в другом табличном пространстве, а индексы - в третьем. При хранении данных длинных полей и больших объектов в отдельных табличных пространствах можно сократить время, затрачиваемое на резервное копирование, если не копировать табличные пространства, содержащие данные длинных полей и больших объектов. Если данные длинных полей и больших объектов важны для вашей работы, следует принять во внимание время резервного копирования этих табличных пространств и время их восстановления. Если данные больших объектов можно взять из отдельного источника, при создании или изменении таблицы, содержащей столбцы больших объектов, выберите опцию NOT LOGGED.

После выполнения операции по реорганизации таблицы следует выполнить резервное копирование всех задействованных табличных пространств. При восстановлении табличных пространств повтор транзакций, связанных с реорганизацией данных, не потребуется.

Примечание: Если выполнить резервное копирование табличного пространства, которое содержит не все данные таблицы, выполнение восстановления с повтором транзакций до указанного момента времени для такого табличного пространства окажется невозможным. Для всех табличных пространств, которые содержат любые типы данных для таблиц, повтор транзакций должен выполняться одновременно, до одного и того же момента времени.

Требуемое время восстановления

Время, требуемое на восстановление базы данных, можно разбить на две составляющие: время, затрачиваемое на восстановление резервной копии и (если для базы данных включена поддержка восстановления с повтором) время, затрачиваемое на применение журналов при повторе транзакций. При составлении плана восстановления следует учесть эти затраты и их влияние на ваши деловые операции. Проверка полного плана восстановления поможет определить, отвечает ли время, необходимое на восстановление базы данных, вашим требованиям. После каждой проверки может потребоваться увеличить частоту создания резервных копий. Если в состав стратегии входит восстановление с повтором транзакций, это уменьшит число журналов, архивируемых между резервными копированиями и, как результат, сократит время, затрачиваемое на повтор транзакций базы данных после операции восстановления.

Примечание: Значение параметра конфигурации менеджера баз данных ″разрешение внутриразделного параллелизма″ (*intra_parallel*) не оказывает влияния на производительность ни для резервного копирования, ни для восстановления. Для каждой из этих операций, независимо от значения параметра *intra_parallel*, может быть использовано несколько процессов.

Требования к памяти

После выбора метода восстановления следует рассмотреть требуемый объем памяти.

При использовании метода восстановления версии требуется место для хранения резервной копии базы данных и восстановленной базы данных. При использовании метода восстановления с повтором транзакций требуется место для хранения резервной копии базы данных или табличных пространств, восстановленной базы данных и архивных журналов базы данных.

Если таблица содержит столбцы длинных полей или больших объектов, следует обдумать размещение этих данных в отдельном табличном пространстве. Это скажется на требованиях к объему памяти и на плане восстановления. Сохраняя данные длинных полей и больших объектов в отдельном табличном пространстве и зная время, требующееся для резервного копирования таких данных, можно выбрать использование плана восстановления, в котором сохранение резервной копии этого табличного пространства будет происходить только изредка. Кроме того, при создании или изменении таблицы со столбцами больших объектов можно указать, чтобы не регистрировать изменения для этих столбцов. Это сократит размер требующегося пространства журнала и соответствующего пространства архива журналов.

Резервная копия табличного пространства SMS, где содержатся большие объекты, может быть больше исходного табличного пространства. Эта разница

может составлять до 40 процентов в зависимости от размера данных больших объектов в табличном пространстве. Например, если создается резервная копия табличного пространства SMS размером 1 Гбайт (содержащего данные больших объектов), при ее восстановлении потребуется дисковое пространство больше 1 Гбайта. Это происходит только в файловых системах, которые поддерживают разреженное распределение (например, в операционных системах на основе UNIX).

Чтобы ошибки носителя не приводили к повреждению базы данных и не лишали возможности ее воссоздания, храните резервную копию базы данных, журналы базы данных и саму базу данных на разных устройствах. По этой причине настоятельно рекомендуем использовать параметр конфигурации *newlogpath*, который позволяет поместить журналы базы данных на отдельное устройство сразу после ее создания. (Этот и другие параметры конфигурации, связанные с регистрацией в журналах, описаны в разделе ″Повтор транзакций в базе данных″ в книге *Руководство администратора: Реализация*.)

Журналы базы данных могут занимать большой объем памяти. При планировании использования метода восстановления с повтором транзакций нужно решить вопрос управления архивными журналами. Возможны следующие варианты:

- Выделить для хранения журналов достаточно места в каталоге пути журналов базы данных.
- Вручную копировать журналы на устройство хранения или в каталог, отличающийся от каталога пути журналов базы данных, после того как они перестают быть активными.
- Воспользоваться обработчиком пользователя, чтобы скопировать эти журналы на другое устройство хранения в вашей среде. (Дополнительную информацию смотрите в разделе ″Обработчик пользователя для восстановления базы данных″ книги *Руководство администратора: Реализация*.)

Примечание: В OS/2 DB2 поддерживает программу обработчика пользователя для управления хранением образов резервных копий базы данных и журналов базы данных как на стандартных, так и на нестандартных устройствах. Дополнительную информацию смотрите в разделе ″Обработчик пользователя для восстановления базы данных″ книги *Руководство администратора: Реализация*.

Сохранение связей данных

При разработке своей базы данных вы учитываете отношения, которые существуют между таблицами. Эти отношения могут быть выражены на уровне программы, когда транзакции изменяют сразу несколько таблиц, или на уровне базы данных, где между таблицами существует реляционная целостность или где триггеры одной таблицы влияют на другую таблицу. Эти отношения следует

рассмотреть при разработке плана восстановления. Вероятно, вы захотите создать совместную резервную копию связанных наборов данных. Такие наборы могут быть заданы как на уровне табличного пространства, так и на уровне базы данных. При сохранении связей наборов данных возможно восстановление до момента времени, где эти данные согласованы. Это особенно важно, если вам нужно выполнять восстановление с повтором транзакций до указанного момента времени для табличных пространств.

Ограничения по использованию различных операционных систем

При работе в среде, где используется несколько операционных систем, следует учитывать, что планы резервного копирования и восстановления могут не интегрироваться. Это значит, что, получив копию при помощи команды BACKUP DATABASE в одной операционной системе, нельзя восстановить ее командой RESTORE DATABASE в другой операционной системе. Для каждой операционной системы планы восстановления должны быть отдельными и независимыми.

Следует переместить таблицы из одной операционной системы в другую, воспользовавшись командой **db2move** или командой EXPORT с командами IMPORT или LOAD. Дополнительную информацию смотрите в руководстве *Data Movement Utilities Guide and Reference*.

Восстановление поврежденного табличного пространства

Поврежденное табличное пространство содержит один или несколько контейнеров, к которым невозможен доступ. Часто это происходит из-за ошибок носителя - постоянных (например, в связи выходом из строя) или временных (из-за отключения диска или демонтирования файловой системы).

Если поврежденное табличное пространство - табличное пространство системного каталога, базу данных перезапустить не удастся. Если ошибки контейнеров нельзя исправить, взяв неповрежденные исходные данные, есть только две возможности:

- Восстановить базу данных или
- Восстановить табличное пространство каталога.

Примечание: Табличное пространство восстановимо только для восстановимых баз данных, поскольку для базы данных должен быть выполнен повтор транзакций.

Если поврежденное табличное пространство не является табличным пространством системного каталога, DB2 пытается сделать доступной максимально возможную часть базы данных; успех в этом случае зависит от стратегии записи в журналы.

Если поврежденное табличное пространство - единственное временное табличное пространство, следует создать новое временное табличное

пространство, как только с базой данных будет установлено соединение. Новое временное табличное пространство может использоваться сразу после создания; можно также возобновить обычные операции базы данных, для которых требуется временное табличное пространство. Автономное временное табличное пространство можно отбросить по желанию. Реорганизация таблиц с использованием системного временного табличного пространства имеет определенные особенности:

- Если для параметра конфигурации *indexrec* базы данных или менеджера баз данных задано значение ″RESTART″, все недопустимые индексы должны быть перестроены при активации базы данных; к ним относятся индексы реорганизации, поврежденные на этапе построения.
- Если существуют незаконченные требования реорганизации в поврежденном временном табличном пространстве, возможно придется задать для параметра конфигурации *indexrec* значение ″ACCESS″ во избежание ошибок при перезапуске.

Восстановление табличных пространств для восстановимых баз данных:
Поврежденное табличное пространство переводится в автономное и недоступное состояние, а также в состояние отложенного повтора транзакций, так как необходимо восстановление после сбоя. Перезапуск пройдет успешно, если нет дополнительных ошибок. Поврежденным табличным пространством снова можно будет пользоваться, если вы:

- Исправите поврежденные контейнеры, не потеряв исходные данные, а затем выполните для табличного пространства повтор транзакций. (Повтор транзакций будет первой попыткой перевода табличного пространства из автономного состояния в рабочее.)
- Выполните после исправления поврежденных контейнеров операцию восстановления табличного пространства (с потерей или без потери исходных данных), а затем и повтор транзакций.

Восстановление табличных пространств для невосстановимых баз данных:
Поскольку необходимым методом восстановления является восстановление после сбоя, а журналы не хранятся неограниченно, перезапуск может быть успешным только в случае, если пользователь решит отбросить поврежденные табличные пространства. (Успешное выполнение восстановления после сбоя подразумевает, что будут сделаны записи журнала, необходимые для восстановления поврежденных табличных пространств в состояние согласованности, поэтому единственным допустимым действием в отношении таких табличных пространств является их отбрасывание.)

Это можно сделать, вызвав операцию перезапуска базы данных без задания опций. Она завершится успешно, если не будет поврежденных табличных пространств. При неудачном завершении (SQL0290N) в файле `db2diag.log` можно найти полный список поврежденных на данный момент табличных пространств.

- Если вы решите отбросить все эти табличные пространства по завершении операции перезапуска базы данных, можно запустить другую операцию перезапуска базы данных, перечислив все поврежденные табличные пространства в опции DROP PENDING TABLESPACES. При включении табличного пространства в список DROP PENDING TABLESPACES оно переводится в состояние отложенного отбрасывания, и после восстановления у вас остается единственный выбор - отбросить это табличное пространство. Операция перезапуска проводится без восстановления данного табличного пространства. Если поврежденное табличное пространство *не* включить в список DROP PENDING TABLESPACES, операция восстановления базы данных завершится неудачно с кодом SQL0290N.
- Если отбрасывание поврежденных табличных пространств (с потерей в них данных) нежелательно, можно выбрать следующее:
  - Подождать и исправить поврежденные контейнеры (без потери исходных данных), а затем снова попытаться запустить операцию перезапуска
  - Выполнить операцию восстановления базы данных.

Примечание: Помещение имени табличного пространства в список DROP PENDING TABLESPACES не означает, что табличное пространство будет переведено в состояние отложенного отбрасывания. Это произойдет, только если оно будет оставаться поврежденным во время операции перезапуска. После успешного завершения операции перезапуска для каждого табличного пространства, находящегося в состоянии отложенного отбрасывания, нужно выполнить оператор DROP TABLESPACE (чтобы выяснить, какие табличные пространства находятся в этом состоянии, вызовите команду LIST TABLESPACES). Этим способом пространство может быть восстановлено или создано заново.

Вопросы производительности восстановления

Решая вопрос повышения производительности восстановления, следует учитывать следующее:

- Производительность для часто обновляемых баз данных можно улучшить, разместив журналы на отдельном устройстве. В среде оперативной обработки транзакций (OLTP) часто требуется больше операций ввода/вывода для записи данных в журналы, чем для сохранения строк данных. Размещение журналов на отдельном устройстве минимизирует позиционирование головок, необходимое для перемещения между файлами журналов и базы данных.

  Следует также учитывать, какие еще файлы находятся на этом диске. Например, перемещение журналов на диск, используемый для подкачки страниц в системе, где не хватает реальной памяти, сведет на нет усилия по настройке.

- Чтобы сократить время, требующееся на выполнение операции восстановления:
  - Скорректируйте размер буфера восстановления. Размер этого буфера должен быть кратен размеру буфера, использовавшегося при резервном копировании.
  - Увеличьте число буферов.

    Чтобы при использовании нескольких буферов и каналов ввода/вывода каналы не ожидали данные, число буферов должно быть, как минимум, вдвое больше числа каналов. Размер используемого буфера повлияет также на производительность операции восстановления. Идеальный размер буфера восстановления должен быть кратен размеру экстента табличных пространств.

    Если у вас несколько табличных пространств с разными размерами экстентов, укажите кратное самому большому размеру экстента.

    Минимальное рекомендуемое число буферов равно числу устройств носителей плюс число, заданное для опции PARALLELISM.
  - Воспользуйтесь несколькими исходными устройствами.
  - Установите значение опции PARALLELISM для операции восстановления как минимум на единицу больше числа исходных устройств.
- Если таблица содержит много данных длинных полей и больших объектов, ее восстановление может отнять очень много времени. В случае доступности для базы данных восстановления с повтором транзакций команда RESTORE позволяет восстанавливать табличные пространства выборочно. Если данные длинных полей и больших объектов важны для вашего бизнеса, следует принять во внимание время восстановления этих табличных пространств и время их резервного копирования. При хранении данных длинных полей и больших объектов в отдельных табличных пространствах можно сократить время, затрачиваемое на восстановление, не восстанавливая табличные пространства, содержащие данные длинных полей и больших объектов. Если данные больших объектов воспроизводятся из отдельного источника, при создании или изменении таблицы, содержащей столбцы больших объектов, выберите опцию NOT LOGGED. Если вы решили не восстанавливать табличные пространства, содержащие данные длинных полей и больших объектов, но вам нужно восстановить табличные пространства, содержащие данную таблицу, следует выполнить повтор транзакций до окончания журналов, чтобы все табличные пространства, содержащие табличные данные, были согласованы.

  Примечание: При выполнении резервного копирования табличного пространства, которое содержит табличные данные, без связанных с ними длинных полей или больших объектов, выполнение восстановления с повтором транзакций до указанного момента времени для такого табличного пространства окажется невозможным. Для всех табличных

пространств таблицы повтор транзакций должен быть выполнен одновременно, до одного и того же момента времени.

- Следующие рекомендации относятся как к резервному копированию, так и к восстановлению:
  - Используйте несколько буферов и устройств ввода вывода.
  - Выделяйте буферов как минимум в два раза больше, чем используемых устройств.
  - Не превышайте пропускную способность контроллера устройства ввода/вывода.
  - По возможности используйте больше буферов мелкого размера вместо небольшого количества крупных буферов.
  - Настраивайте число и размер буферов в соответствии с системными ресурсами.

Особенности аварийного восстановления

Термин *аварийное восстановление* используется при описании необходимых действий по восстановлению базы данных в случае пожара, землетрясения, актов вандализма или других катастрофических событий. В плане аварийного восстановления можно предусмотреть:

- Место, которое будет использоваться при случае опасности
- Отдельный компьютер, на котором будет производиться восстановление базы данных
- Удаленное хранение резервных копий базы данных и архивных журналов.

Если план аварийного восстановления предусматривает сохранение всей базы данных на другом компьютере, требуются как минимум полная резервная копия и все архивные журналы базы данных. Можно хранить самую новую резервную базу данных, применяя к ней журналы по мере их архивирования. Либо хранить резервную базу данных и архивы журналов на резервном узле и выполнять восстановление и повтор транзакций только после аварии. (В последнем случае, естественно, желательна наиболее свежая резервная копия базы данных.) Однако в случае аварии обычно невозможно восстановить все транзакции до момента самой аварии.

Ценность резервной копии табличного пространства для аварийного восстановления зависит от области ошибки. Обычно при аварийном восстановлении требуется восстановление всей базы данных, поэтому следует хранить полную резервную копию базы данных в отдельном месте. Даже если существуют отдельные образы резервных копий для всех табличных пространств, вы не сможете ими воспользоваться для восстановления базы данных. Если при аварии повреждается диск, для восстановления могут использоваться резервные копии табличных пространств для каждого табличного пространства на этом диске. Если из-за ошибки на диске (или по какой-то иной причине) потерян доступ к контейнеру, можно восстановить этот

контейнер в другом месте. Дополнительную информацию смотрите в разделе ″Переопределение контейнеров табличных пространств при восстановлении″ в книге *Руководство администратора: Реализация*.

В любом плане аварийного восстановления могут играть роль как резервные копии табличных пространств, так и полные резервные копии базы данных. Основа плана аварийного восстановления обеспечивается средствами DB2, доступными для резервного копирования, восстановления и повтора транзакций. Для защиты результатов вашей работы следует убедиться в наличии проверенных процедур восстановления.

Уменьшение воздействия ошибок носителя

Чтобы уменьшить вероятность возникновения ошибок носителя и упростить восстановление при таких ошибках:

- Создайте зеркальные копии или дубликаты дисков, где хранятся данные и журналы для важных баз данных.
- В среде многораздельной базы данных установите строгую процедуру для обработки данных и журналов на узле каталога. Поскольку этот узел критичен для поддержки базы данных:
  - Убедитесь, что он находится на надежном диске
  - Сдублируйте его
  - Чаще создавайте резервные копии
  - Не помещайте на него пользовательские данные.

Защита от ошибок диска

Так как существует возможность повреждения данных или журналов из-за сбоев диска, рассмотрим некоторые средства поддержки его отказоустойчивости. Обычно это достигается использованием *массива дисков*. Массив дисков состоит из собрания дисководов, которое воспринимается программами как один большой диск.

В массивах дисков используется *чередование дисков* - распределение файлов по нескольким дискам, зеркальное копирование дисков и проверка данных на четность.

Дисковые массивы иногда называют RAID (Redundant Array of Independent Disks - дублирующий массив независимых дисков). Строго говоря, термин RAID относится только к аппаратным массивам дисков. Массив дисков можно также создать программно на уровне операционной системы или прикладной программы. Аппаратные и программные массивы дисков отличаются между собой способом обработки процессором запросов ввода/вывода. Для аппаратных массивов дисков управление вводом/выводом дисков осуществляется дисковыми контроллерами, а для программных массивов дисков - операционной системой или программой.

Аппаратные массивы дисков (RAID): В массиве дисков RAID несколько дисков управляются дисковым контроллером со своим собственным процессором. Вся логика, необходимая для управления дисками, формирующими этот массив, содержится в контроллере диска, поэтому такое исполнение обеспечивает независимость от операционной системы.

Существует пять типов архитектуры RAID (от RAID-1 до RAID-5), и каждый тип обеспечивает поддержку отказоустойчивости дисков. Все типы отличаются функциональностью и производительностью. Строго говоря, термин RAID относится к массиву с дублированием. RAID-0, где обеспечивается только чередование данных (но не обеспечивается поддержка отказоустойчивости), здесь не обсуждается. Хотя спецификацией RAID определяется пять архитектур, в настоящее время обычно используются только RAID-1 и RAID-5.

RAID-1 называют также зеркальным копированием или дублированием дисков. Зеркальное копирование дисков подразумевает дублирование данных (целых файлов) одного диска на другом с использованием одного дискового контроллера. Дублирование дисков отличается от зеркального копирования только тем, что к дискам присоединяется второй дисковый контроллер (например, два адаптера SCSI). Это обеспечивает надежную защиту данных. При ошибке на одном диске данные остаются доступны на другом. При использовании дублирования возможная ошибка контроллера диска не ставит под угрозу защиту данных. Производительность в архитектуре RAID-1 также высока, но в этом варианте необходимая емкость диска в два раза превышает количество данных, так как данные дублируются по парам дисков.

В RAID-5 по всем дискам используется чередование данных и информации о четности по секторам. Информация о четности остается вместе с информацией о данных, а не сохраняется на особом дисководе. Это обеспечивает надежную защиту данных. При ошибке любого диска данные все равно остаются доступными, так как можно использовать информацию с других дисков, а также информацию о четности. Производительность чтения высокая, но производительность записи гораздо хуже, чем в RAID-1 или на обычном диске. Для конфигурации RAID-5 требуется минимум три одинаковых диска. Величина дополнительного дискового пространства, расходуемого на служебную информацию, зависит от количества дисков в массиве. Если в конфигурации RAID-5 пять дисков, дополнительный расход дискового пространства составляет 20 процентов.

При использовании дискового массива RAID (кроме RAID-0) ошибка на диске не приводит к нарушению доступа к данным в массиве. Если в массиве используются быстросъемные или быстропереустанавливаемые диски, диск с ошибкой можно заменить другим, переустановив его, пока массив находится в использовании. Если в RAID-5 ошибка случается на двух дисках одновременно, все данные теряются (но вероятность такой ошибки очень мала).

Для журналов можно рассмотреть возможность использования RAID-1 или программных зеркальных копий дисков (смотрите в разделе “Программные массивы дисков”); это обеспечит возможность восстановления до момента времени ошибки и высокую производительность записи (что важно для журналов). В случаях, где критична надежность (нельзя терять время на восстановление данных после ошибки), а производительность записи не так критична, рассмотрите возможность использования дисков RAID-5. Наоборот, если критична производительность записи и ее нужно повысить, невзирая на стоимость дополнительного дискового пространства, рассмотрите возможность использования и для данных, и для журналов архитектуры RAID-1.

Программные массивы дисков: Программный массив дисков выполняет те же функции, что и аппаратный (смотрите раздел “Аппаратные массивы дисков (RAID)” на стр. 54), но управление трафиком дисков осуществляется либо операционной системой, либо программой, запускаемой на сервере. Программный массив дисков вынужден делить ресурсы процессора и системы с другими работающими программами. Это недостаток систем с центральным процессором, и следует помнить, что общая производительность такого массива дисков зависит от нагрузки процессора сервера и его емкости.

Типичный программный массив дисков поддерживает зеркальную копию дисков (смотрите раздел “Аппаратные массивы дисков (RAID)” на стр. 54). Хотя для программного массива дисков и требуются дополнительные диски, реализация его обходится дешевле, так как не нужны дорогостоящие контроллеры дисков RAID.

Примечание: Если загрузочный диск системы находится в самом дисковом массиве, вы не сможете запустить систему при ошибке этого диска. Если перед запуском массива дисков происходит ошибка дисковода, массив может не дать допуск к диску. Загрузочный диск нужно отделить от массива дисков.

Уменьшение воздействия ошибок транзакций

Для уменьшения воздействия ошибок транзакций попытайтесь гарантировать:

- Непрерывное электропитание
- Достаточное дисковое пространство для журналов базы данных
- Надежные каналы связи по серверам разделов базы данных в среде многораздельных баз данных
- Синхронизацию системного времени в среде многораздельных баз данных (смотрите раздел “Синхронизация системного времени в системе многораздельных баз данных” на стр. 56).

Синхронизация системного времени в системе многораздельных баз данных

Для успешной работы базы данных и восстановления с повтором транзакций без ограничений необходимо поддерживать реляционную синхронизацию системного времени по серверам разделов базы данных. Различие по серверам разделов базы данных плюс все потенциальные задержки обработки и связи для транзакций не должны превышать значение, заданное для параметра конфигурации менеджера баз данных *max_time_diff* (максимальное различие времени по узлам).

Чтобы временные отметки записей журналов в системе многораздельных баз данных отражали последовательность транзакций, в DB2 в записях журналов в качестве базиса для временных отметок используется системное время каждого компьютера. Если часы системы переводят вперед, в журнал автоматически будет записано новое время системы. Хотя часы системы можно перевести назад, время журналов идти назад не может и оно остается *неизменным*, пока системные часы не дойдут до этого времени. После этого время синхронизируется. Это значит, что кратковременная ошибка системного времени на узле базы данных может долго сказываться на отметках времени журналов базы данных.

Предположим, что системное время на сервере A разделов базы данных было ошибочно установлено на 7 ноября 1999 года (хотя на самом деле был 1997 год), и предположим, что ошибка была исправлена *после* принятия транзакции изменения в разделе на этом сервере разделов базы данных. Если база данных продолжает использоваться и периодически изменяться, все точки между 7 ноября 1997 года и 7 ноября 1999 года будут практически недостижимы для восстановления с повтором транзакций. При выполнении оператора COMMIT на сервере A разделов баз данных отметка времени в журнале базы данных устанавливается на 1999 год, и время журнала остается установленным на 7 ноября 1999 года, пока с ним не совпадет системное время. При попытке повтора транзакций до момента времени в этом промежутке времени операция остановится на первой отметке времени после заданной точки остановки, то есть повтор не продвинется дальше 7 ноября 1997 года.

Хотя DB2 не может управлять изменениями системного времени, параметр конфигурации менеджера баз данных *max_time_diff* сокращает количество ошибок этого типа:

- Конфигурируемые значения этого параметра лежат в диапазоне от 1 минуты до 24 часов. Дополнительную информацию об установке значений параметра *max_time_diff* смотрите в книге *Руководство администратора: Производительность*.
- При первом требовании соединения с узлом, не являющимся узлом каталога, сервер разделов базы данных посылает его время на узел каталога базы данных. После этого узел каталога проверяет, находится ли разница между

временем на узле, требующем соединения, и его собственным временем, в диапазоне, заданном параметром *max_time_diff*. Если этот диапазон превышен, требование на соединение отклоняется.

- Транзакция изменения, обращающаяся к нескольким серверам разделов базы данных, перед тем, как изменение можно будет принять, должна убедиться, что часы участвующих серверов разделов базы данных синхронизированы. Если у двух или более серверов разделов баз данных отличие во времени превышает предел, допускаемый параметром *max_time_diff*, выполняется откат транзакции для предотвращения распространения неправильного времени на другие серверы баз данных.

Чтобы исправить ситуацию и предотвратить дальнейшее распространение неправильной отметки времени в журнале базы данных:

1. Установите для системных часов верное время.
2. Восстановите раздел базы данных на соответствующем сервере разделов базы данных с резервной копией, сделанной до установки неправильного времени.
3. Выполните повтор транзакций до окончания журнала для данного раздела базы данных.
4. Снимите резервную копию раздела базы данных сразу же после повтора транзакций.

По завершении этих действий время журнала будет скорректировано, неправильная отметка времени распространяться не будет, и можно будет выполнить восстановление до указанного момента времени с последней резервной копии, снятой по данному разделу базы данных.

Реорганизация таблиц в базе данных

После большого числа изменений таблица может стать фрагментированной, что приведет к ухудшению производительности. Если после сбора статистики производительность заметно не улучшится, может помочь реорганизация данных таблицы. При реорганизации таблицы данные перестраиваются согласно заданному индексу в физическую последовательность, а свободное пространство, образовавшееся при фрагментации, удаляется. Такая реорганизация может повысить скорость доступа к данным, тем самым улучшив производительность.

Перед реорганизацией таблицы рекомендуется вызвать команду REORGCHK и собрать статистику по данной таблице. Эта команда поможет определить, нужна ли реорганизация таблицы. Информацию о команде REORGCHK смотрите в книге *Command Reference*.

Обзор защиты DB2

Для защиты данных и ресурсов, связанных с сервером баз данных, в DB2 используется сочетание внешних служб защиты и внутренней информации управления доступом. При обращении к серверу баз данных, прежде чем вы получите доступ к базе данных или ресурсам, нужно пройти несколько проверок защиты. Первый шаг в защите базы данных называется *аутентификацией*, на нем вы должны доказать, что вы тот, кем себя называете. Второй шаг защиты называется *авторизация*, на нем менеджер баз данных решает, разрешить или нет этому пользователю выполнить требуемые им действия или получить доступ к требуемым данным.

Аутентификация

Аутентификация пользователя выполняется с использованием внешних по отношению к DB2 средств. Это средство защиты может быть частью операционной системы, отдельным продуктом, а в некоторых случаях может и вовсе не существовать. В операционных системах на основе UNIX такое средство защиты есть в самой операционной системе. DCE Security Services - самостоятельный продукт, предоставляющий средство защиты для распределенной среды. В операционных системах Windows 9x и Windows 3.1 средств защиты нет.

Для аутентификации пользователя защита требует два элемента: ID пользователя и пароль. ID пользователя идентифицирует пользователя для средства защиты. Пароль (информация, известная только пользователю и средству защиты) служит для проверки идентичности пользователя (его соответствия этому ID пользователя).

После аутентификации:

- Пользователь должен быть идентифицирован в DB2 при помощи имени авторизации SQL или *ID авторизации*. Это имя может быть тем же ID пользователя или отображенным значением. Например, в системах на основе UNIX *ID авторизации* DB2 получается преобразованием ID пользователя UNIX в верхний регистр, что соответствует соглашениям об именах DB2. В DCE Security Services *ID авторизации* DB2 содержится в реестре DCE и извлекается оттуда после успешного завершения аутентификации.
- Должен быть получен список групп, к которым принадлежит данный пользователь. Членство в группе может использоваться при авторизации пользователя. Группы - это объекты защиты, которые также нужно отобразить на имена авторизации DB2. Это отображение проводится подобно тому, как это делается для ID пользователей.

  DB2 получит список максимум из 64 групп. Если пользователь входит больше, чем в 64 группы, в список групп DB2 будут добавлены только первые 64 группы, которые будут отображены на допустимые имена авторизации DB2. Никакие ошибки не возвращаются, а все группы, начиная с 65, игнорируются DB2.

DB2 использует средство защиты для аутентификации пользователей одним из двух способов:

- DB2 принимает в качестве доказательства идентичности успешную регистрацию в системе защиты и позволяет:
  - Использовать локальные команды для доступа к локальным данным
  - Использовать удаленные соединения, в которых сервер доверяет аутентификации клиента.
- DB2 принимает комбинацию ID пользователя и пароля. DB2 использует в качестве доказательства идентичности успешную проверку этой пары защитой и позволяет:
  - Использовать удаленные соединения, в которых сервер требует доказательства аутентификации клиента.
  - Использовать операции, где пользователь пытается запустить команду, применив параметры идентификации, отличающиеся от использовавшихся при регистрации.

Администраторы DB2 могут позволять другим пользователям изменять пароли в системах AIX и Windows NT EEE через переменную реестра профиля DB2CHGPWD_EEE. Значение по умолчанию для этой переменной - NOT SET (отключено). DB2CHGPWD_EEE принимает стандартные логические значения, используемые другими переменными профиля DB2.

Администратор DB2 отвечает за централизованную поддержку паролей для всех узлов и использует для этого либо Windows NT Domain Controller в Windows NT, либо NIS в AIX.

Примечание: Если пароли не поддерживаются централизованно, включение переменной DB2CHGPWD_EEE может привести к несогласованности паролей по узлам. В таком случае при использовании возможности ″изменить пароль″ пароль будет изменен только на узле, с которым вы соединяетесь.

DB2 UDB в AIX может регистрировать попытки ввода неверных паролей средствами операционной системы и определять, когда клиент превысит число разрешаемых попыток регистрации, заданное параметром LOGINRETRIES.

Дополнительную информацию о системной проверке правильности ввода (это особенно важно, когда к базе данных обращаются удаленные клиенты), смотрите в разделе ″Выбор метода аутентификации для сервера″ книги *Руководство администратора: Реализация*.

Авторизация

Авторизация - это процесс, при котором DB2 получает информацию о своем аутентифицированном пользователе с указанием операций, которые он может выполнять, и объектов, к которым можно обращаться. Для каждого требования

пользователя может выполняться несколько проверок авторизации в зависимости от вовлекаемых объектов и операций.

Авторизация выполняется средствами DB2. Для записи разрешений, связанных с каждым именем авторизации используются таблицы и файлы конфигурации DB2. Имя авторизации аутентифицированного пользователя и имена тех групп, к которым он принадлежит, сравниваются с записанными разрешениями. На основе этого сравнения DB2 решает вопрос о допустимости требуемого доступа.

Существует два типа записанных DB2 разрешений: привилегии и полномочия. *Привилегия* определяет одиночное разрешение для имени авторизации, позволяющее пользователю создавать ресурсы базы данных или получать к ним доступ. Привилегии хранятся в каталогах базы данных. *Уровни полномочий* дают способ группировки привилегий и управления высокоуровневой поддержкой менеджера баз данных и операциями утилит. Полномочия, определяемые базой данных, хранятся в каталогах базы данных; системные полномочия связываются с членством в группах и хранятся в файле конфигурации менеджера баз данных для данного экземпляра.

Группы обеспечивают удобство выполнения авторизации для нескольких пользователей, устраняя необходимость предоставления или отзыва привилегий для каждого пользователя отдельно. Для авторизации, если не задано иначе, имена авторизации групп могут использоваться везде, где используются имена авторизации. В целом членство в группе рассматривается для авторизации динамического SQL и объектов, не являющихся объектами базы данных (например, команд и утилит уровня экземпляра), но не рассматривается для статического SQL. Исключением из общего случая является предоставление привилегий PUBLIC: они рассматриваются при обработке статического SQL. Особые случаи, когда членство в группах не используется, отмечены в документации DB2 везде, где это применимо.

Дополнительную информацию смотрите в разделе ″Привилегия, полномочия и авторизация″ книги *Руководство администратора: Реализация*.

Обзор аутентификации и авторизации для баз данных объединения

Так как система баз данных объединения DB2 может получать доступ к информации нескольких систем управления базами данных, для защиты данных могут потребоваться дополнительные шаги.

При планировании подхода к аутентификации учтите, что пользователи могут проходить проверку аутентификации не только на DB2, но и на источниках данных. В системе объединения аутентификация может проводиться на рабочих станциях клиентов DB2, на серверах DB2, на источниках данных (DB2, DB2 for OS/390, другие серверы DRDA, Oracle) или на DB2 (клиент или сервер DB2) в сочетании с источниками данных. Даже в средах DCE могут оказаться необходимыми определенные шаги, если источник данных потребует ID

пользователя и пароль. Дополнительную информацию смотрите в разделе "Обработка аутентификации базы данных объединения" книги *Руководство администратора: Реализация*.

Точно также в DB2 и на источниках данных пользователи должны проходить и проверку авторизации. Каждый источник данных (DB2, Oracle, DB2 for OS/390 и т.п.) поддерживает защиту объектов под своим собственным управлением. При выполнении пользователем операции с псевдонимом он обязан пройти проверку авторизации для таблицы или производной таблицы, к которой обращаются по этому псевдониму.

Глава 3. Системы объединения

Система баз данных объединения или *система объединения* - это система управления базами данных (СУБД), которая обеспечивает для программ и пользователей, выдающих операторы SQL, возможность обращения к нескольким СУБД или к нескольким базам данных в одном операторе. Примером служит объединение между таблицами из двух различных баз данных DB2. Такой тип оператора называется *распределенным требованием*.

Система объединения DB2 Universal Database обеспечивает поддержку распределенных требований по базам данных и СУБД. Можно, например, выполнить операцию UNION между таблицей DB2 и представлением Oracle. К поддерживаемым СУБД относятся DB2, продукты семейства DB2 (например, DB2 for OS/390 и DB2 for AS/400) и Oracle.

Система объединения DB2 обеспечивает *прозрачность положений* для объектов баз данных. Если информация (в таблицах и производных таблицах) перемещается, можно исправить ссылки (так называемые *псевдонимы*), не внося никаких изменений в сами программы, запрашивающие информацию. Кроме того, система объединения DB2 предоставляет возможность *работать* с СУБД, которые не полностью поддерживают диалект SQL DB2 или какие-либо возможности оптимизации. Операции, которые нельзя выполнить в таких СУБД (например, рекурсивный SQL), запускаются под DB2.

Системы объединения DB2 работают *полуавтономным* способом: запросы DB2, содержащие ссылки на объекты Oracle, могут передаваться на Oracle, когда программы Oracle обращаются к тому же серверу. Система объединения DB2 не монополизирует и не ограничивает доступ (помимо ограничений целостности и блокировок) к объектам Oracle и других СУБД.

Система объединения DB2 состоит из экземпляра DB2 UDB, базы данных, которая будет служить *базой данных объединения*, и одного или нескольких *источников данных*. База данных объединения содержит записи каталога, идентифицирующие источники данных и их характеристики. Источник данных включает в себя СУБД и данные. Программы обращаются к базе данных объединения, как и к любой другой базе данных DB2. Среда базы данных объединения показана на рис. 20 на стр. 64.

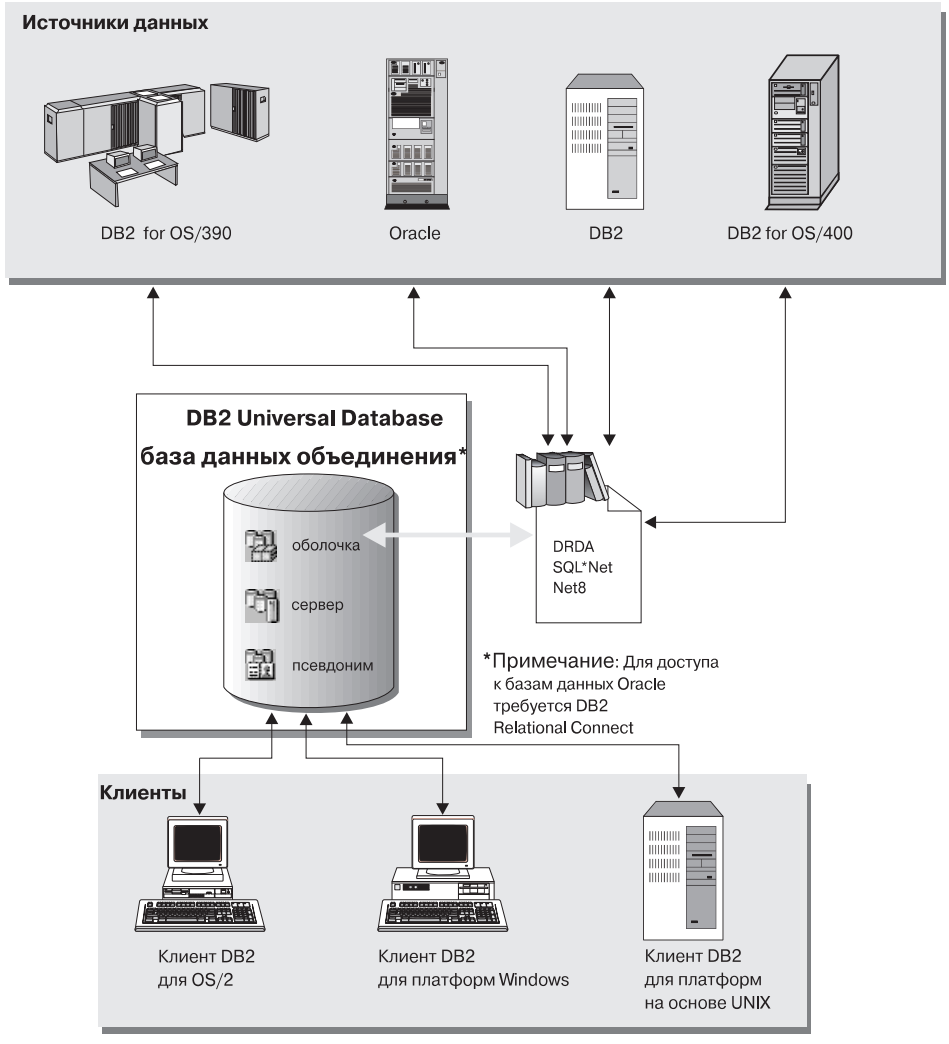


Рисунок 20. Система баз данных объединения

Записи каталога объединения DB2 содержат информацию об объектах источника данных: их имена, содержащуюся в них информацию и условия, на которых их можно использовать. Поскольку этот каталог DB2 хранит информацию об объектах из многих СУБД, он называется *глобальным каталогом*. В этом каталоге хранятся атрибуты объектов. Реальные СУБД, к которым обращаются, модули, используемые для связи с источником данных, и объекты данных СУБД (например, таблицы), к которым потребуется доступ, хранятся вне базы данных. (Одно исключение: база данных объединения может сама являться источником данных для системы объединения.) Объекты

объединения можно создать с помощью Центра управления или операторов SQL DDL. Необходимые объекты базы данных объединения:

Оболочки
Идентифицируют модули (DLL, библиотеки и так далее), используемые для доступа к конкретному классу или категории источника данных.

Серверы
Определяют источники данных. Данные сервера включают имя оболочки, имя сервера, тип сервера, версию сервера, информацию авторизации и опции сервера.

Псевдонимы
Идентификаторы, хранимые в базе данных объединения - ссылки на конкретные объекты источника данных (таблицы, алиасы, производные таблицы). Программы используют псевдонимы в запросах так же, как имена таблиц и производных таблиц.

В зависимости от конкретных потребностей вы можете создавать дополнительные объекты:

- Отображения пользователей - для целей аутентификации
- Отображения типов данных - для настройки отношений между типом источника данных и типом DB2
- Отображения функций - для отображения локальных функций в функции источника данных
- Спецификации индексов - для повышения производительности.

Когда система объединения сконфигурирована, к информации источников данных можно обращаться так, как если бы она находилась в одной большой базе данных. Пользователи и программы посылают запросы на одну базу объединения, которая затем получает данные от систем семейства DB2 и Oracle по необходимости. Пользователи и программы задают в запросах псевдонимы; при помощи этих псевдонимов производится обращение к таблицам и производным таблицам на источниках данных. С точки зрения конечного пользователя псевдонимы подобны алиасам.

На производительность системы объединения влияют многие факторы. Самый важный фактор - обеспечить точность и актуальность информации об источниках данных и объектах, хранимой в глобальном каталоге базы данных объединения. Эта информация используется оптимизатором DB2 и может повлиять на решения об операциях для оценки источников данных. Дополнительную информацию о производительности системы объединения смотрите в книге *Руководство администратора: Производительность*.

Система объединения DB2 работает с некоторыми ограничениями. Распределенные требования ограничиваются операциями только для чтения.

Кроме того, нельзя применять к псевдонимам утилиты (LOAD, REORG, REORGCHK, IMPORT, RUNSTATS и т.п.).

Однако благодаря работе через промежуточный сервер операторы DDL и DML можно передать непосредственно менеджерам баз данных, используя диалект SQL этого источника данных.

Системы объединения допускают параллельные среды. Повышение производительности ограничивается степенью, до которой запрос базы данных объединения может быть семантически разбит на обращения к локальным объектам (таблицам, производным таблицам) и обращения к псевдонимам. Требования данных псевдонимов обрабатываются последовательно; локальные объекты могут обрабатываться параллельно. Например, если дан запрос `SELECT * FROM A, B, C, D`, где A и B - локальные таблицы, а C и D - псевдонимы, ссылающиеся на таблицы на источниках данных Oracle, возможен план объединения, при котором таблицы A и B объединяются с использованием параллельного объединения. Затем результаты объединяются последовательно с псевдонимами C и D.

Включение поддержки системы объединения

Базы данных объединения поддерживаются в DB2 Enterprise Edition (EE) и DB2 Enterprise - Extended Edition (EEE). Чтобы включить поддержку системы объединения:

1. При установке выберите для опции установки *Распределенное объединение для баз данных DB2* значение DB2 EE или EEE.
2. Если в состав системы объединения входят базы данных Oracle, установите DB2 Relational Connect. Дополнительную информацию смотрите в руководстве *Дополнение по установке и настройке*.
3. Задайте для параметра конфигурации менеджера баз данных *federated* значение ″YES″.
4. Создайте оболочки, серверы и псевдонимы (дополнительную информацию смотрите в разделе ″Создание базы данных″ в книге *Руководство администратора: Реализация*).
5. При необходимости создайте дополнительные объекты или задайте опции (дополнительную информацию смотрите в разделе ″Реализация проекта″ в книге *Руководство администратора: Реализация*).

Глава 4. Параллельные системы баз данных

DB2 расширяет возможности менеджера баз данных на параллельную многоузловую среду. *Раздел базы данных* - это часть базы данных, куда входят собственные данные, индексы, файлы конфигурации и журналы транзакций. Раздел базы данных называют иногда узлом или узлом базы данных. (Термин ″узел″ - node - использовался в DB2 Parallel Edition for AIX Версии 1.)

Одноразделная база данных - это база данных с единственным разделом. В этом разделе хранятся все данные базы данных. В этом случае группы узлов (смотрите раздел “Группы узлов” на стр. 11), даже если они заданы, не дают дополнительных возможностей.

Многораздельная база данных - это база данных с несколькими разделами. Таблицы могут находиться в одном или нескольких разделах базы данных. Если таблица находится в группе узлов, состоящей из нескольких разделов, какие-то ее строки хранятся в одном разделе, а другие - в других разделах.

Обычно на каждом физическом узле существует один раздел, а процессоры каждой системы используются менеджером баз данных в каждом разделе базы данных для управления именно его частью общих данных базы данных.

Поскольку данные разделены по разделам базы данных, для удовлетворения информационных требований можно использовать мощность нескольких процессоров на нескольких физических узлах. Требования на получение и изменение данных автоматически расчленяются на подтребования и выполняются параллельно на соответствующих разделах базы данных. Тот факт, что базы данных разбиты на разделы, незаметен для пользователей, вводящих операторы SQL.

Взаимодействие пользователей осуществляется через один раздел базы данных, называемый *узлом координатора*. Координатор запускается на том же разделе базы данных, что и программа, или, в случае удаленной программы, на разделе базы данных, с которой соединяется запущенная программа. Узлом координатора может быть любой раздел базы данных.

Группы узлов и разделение данных

В базе данных можно задавать именованные подмножества, охватывающие один или несколько разделов. Такие подмножества называются *группами узлов*. Подмножества, которые содержат несколько разделов базы данных, называются *многораздельными группами узлов*. Многораздельные группы узлов могут быть определены только для разделов базы данных, принадлежащих одному экземпляру.

На рис. 21 на стр. 69 показан пример базы данных с пятью разделами, в которой:

- Группа узлов охватывает все разделы базы данных кроме одного (Группа узлов 1).
- Группа узлов содержит один раздел базы данных (Группа узлов 2).
- Группа узлов содержит два раздела базы данных (Группа узлов 3).
- Раздел базы данных в Группе узлов 2 используется совместно (и перекрывается) с Группой узлов 1.
- В группе узлов 3 есть один раздел базы данных, который используется совместно (и перекрывается) с Группой узлов 1.

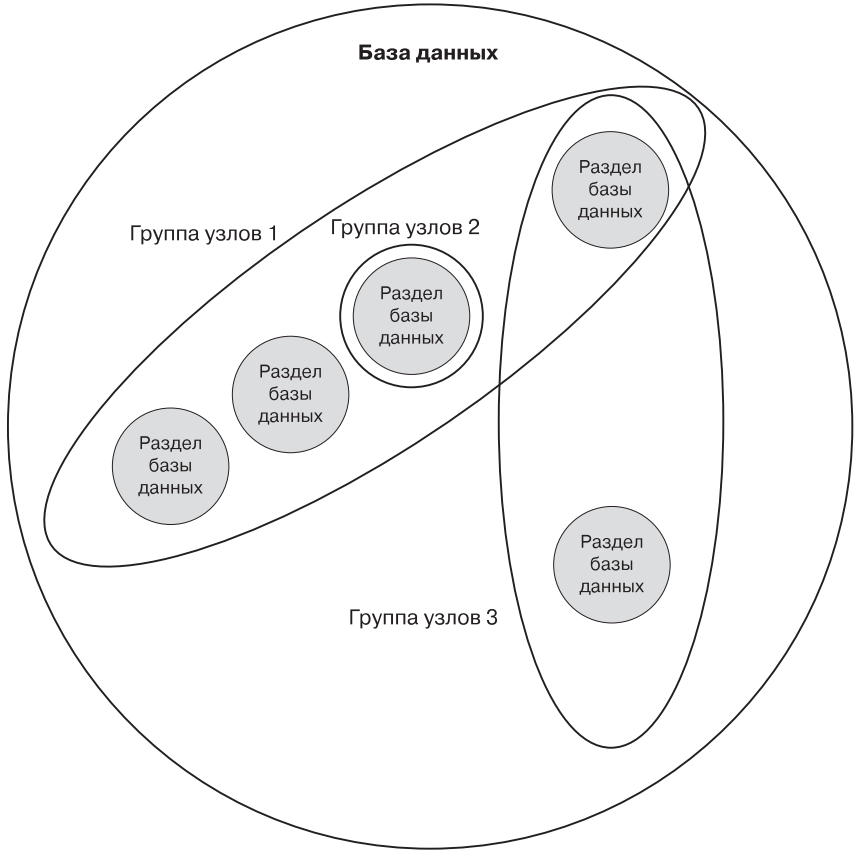


Рисунок 21. Группы узлов в базе данных

Новую группу узлов можно создать при помощи оператора CREATE NODEGROUP. Дополнительную информацию смотрите в справочнике *SQL Reference*. Данные делятся между всеми разделами группы узлов. При использовании многораздельных групп узлов следует учитывать некоторые особенности их разработки. Дополнительную информацию смотрите в разделе “Проектирование групп узлов” на стр. 135.

Типы параллелизма

Компоненты задачи, например запрос к базе данных, могут запускаться параллельно, что существенно улучшает производительность. Характер задачи, конфигурация базы данных и аппаратная среда - все это определяет, как DB2 будет выполнять задачу параллельным способом. Эти особенности взаимосвязаны и должны рассматриваться совместно при разработке базы данных на физическом и логическом уровне. В этом разделе описаны следующие поддерживаемые DB2 типы параллелизма:

- Параллелизм ввода/вывода
- Параллелизм запросов
- Параллелизм утилит

Параллелизм ввода/вывода

Если в табличном пространстве есть несколько контейнеров, менеджер баз данных может использовать *параллельный ввод/вывод*. Параллельный ввод/вывод - это процесс одновременной записи или чтения на нескольких устройствах ввода/вывода; он может значительно улучшить пропускную способность.

Параллелизм ввода/вывода - компонент каждой аппаратной среды, описанной в разделе “Аппаратные среды” на стр. 73. В Табл. 3 на стр. 82 описано, в каких аппаратных средах возможен параллелизм ввода/вывода.

Параллелизм запросов

Параллелизм запросов бывает двух типов: внутренний и внешний.

Внешний параллелизм - это возможность посылки запросов к базе данных несколькими программами одновременно. Каждый запрос выполняется независимо от других, но все они выполняются DB2 одновременно. DB2 всегда поддерживала этот тип параллелизма.

Внутренний параллелизм означает одновременную обработку частей одного запроса с использованием *внутрираздельного параллелизма*, *межраздельного параллелизма* или обоих типов.

Именно в данном смысле термин *параллелизм запросов* используется во всей этой книге.

Внутрираздельный параллелизм

Внутрираздельный параллелизм - это возможность разбивать запрос на несколько частей. (Некоторые утилиты также поддерживают этот тип параллелизма. Смотрите раздел “Параллелизм утилит” на стр. 73.)

Внутрираздельный параллелизм подразделяет операции базы данных, например, создание индекса, загрузка базы данных или запрос SQL, которые обычно рассматриваются как одиночные, на несколько частей, некоторые из которых или все могут быть запущены параллельно *в пределах одного раздела базы данных*.

На рис. 22 на стр. 71 показан запрос, который разбивается на четыре составляющие, запускаемые параллельно; результаты при этом возвращаются быстрее, чем при выполнении запроса последовательным способом. Эти составляющие являются копиями друг друга. Для использования внутрираздельного параллелизма нужно соответствующим образом

сконфигурировать базу данных. Степень параллелизма можно задать самому или позволить это сделать системе. Степень параллелизма - это число составляющих запроса, запускаемых параллельно.

В Табл. 3 на стр. 82 описано, в каких аппаратных средах возможен внутриразделный параллелизм.

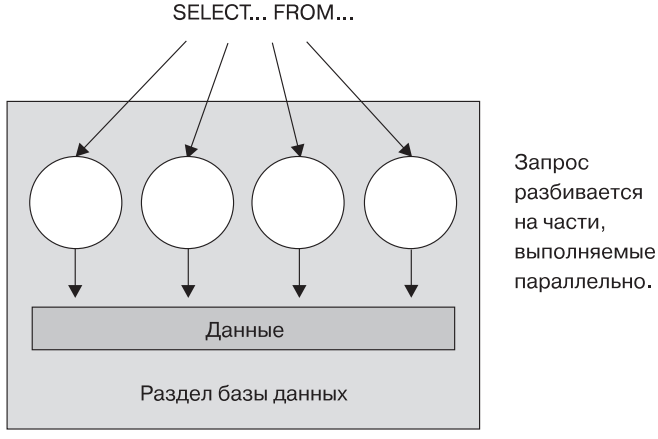


Рисунок 22. Внутриразделный параллелизм

Межраздельный параллелизм

Межраздельный параллелизм - это возможность разбить запрос на несколько частей по нескольким разделам многораздельной базы данных на одном или нескольких компьютерах. Запрос выполняется параллельно. (Некоторые утилиты также поддерживают этот тип параллелизма. Смотрите раздел “Параллелизм утилит” на стр. 73.)

Операции базы данных, например, создание индекса, загрузка базы данных или запрос SQL, которые обычно рассматриваются как одиночные, подразделяются межраздельным параллелизмом на несколько частей, некоторые из которых или все могут быть запущены параллельно *на нескольких разделах многораздельной базы данных на одном или нескольких компьютерах*.

На рис. 23 на стр. 72 показан запрос, который разбивается на четыре составляющие, запускаемые параллельно; результаты при этом возвращаются быстрее, чем при выполнении запроса последовательным способом на одном компьютере.

Степень параллелизма в значительной мере определяется числом создаваемых разделов и тем, как вы задали группы узлов.

В Табл. 3 на стр. 82 описано, в каких аппаратных средах возможен межраздельный параллелизм.

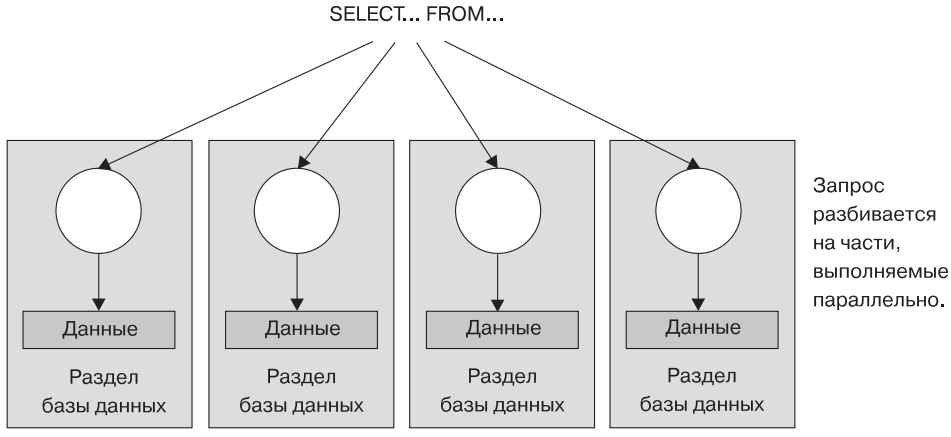


Рисунок 23. Межраздельный параллелизм

Одновременный внутриразделный и межраздельный параллелизм

Внутриразделный и межраздельный параллелизм могут использоваться одновременно. Такое сочетание обеспечивает двукратность параллелизма, в результате чего скорость обработки запросов возрастает еще сильнее:

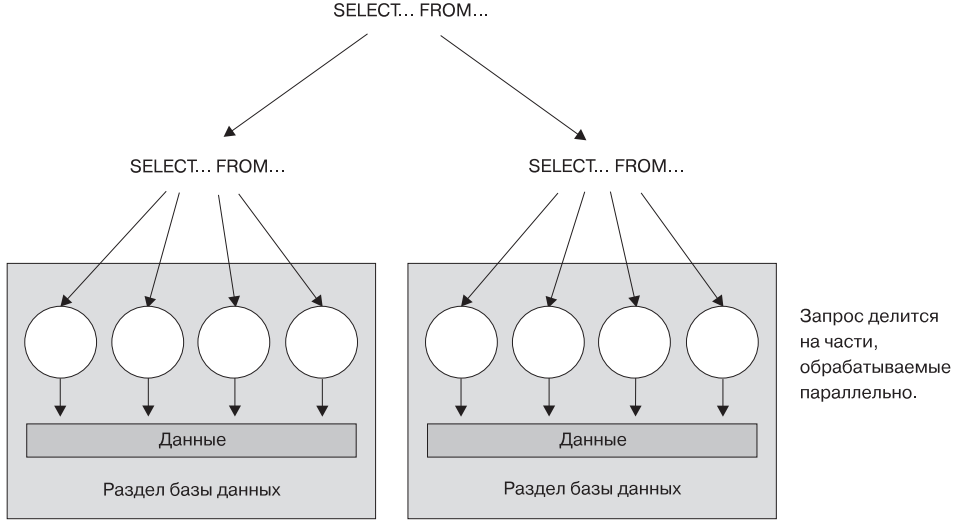


Рисунок 24. Одновременный межраздельный и внутриразделный параллелизм

Параллелизм утилит

Утилиты DB2 могут использовать преимущества внутриразделького параллелизма. Кроме того, они могут использовать межраздельный параллелизм; если заданы разделы многораздельной базы данных, утилиты запускаются в каждом разделе параллельно.

Утилита загрузки может использовать преимущества внутриразделького параллелизма. Загрузка данных - задача с интенсивным использованием процессора. Утилита загрузки использует преимущества нескольких процессоров для таких задач, как синтаксический анализ и форматирование данных. Кроме того, она может использовать серверы параллельного ввода/вывода для параллельной записи данных в контейнеры. Дополнительную информацию о подключении поддержки параллелизма для утилиты загрузки смотрите в книге *Data Movement Utilities Guide and Reference*.

В среде многораздельных баз данных утилита AutoLoader использует преимущества внутриразделького параллелизма, межраздельного параллелизма и параллелизма ввода/вывода с применением параллельных вызовов команды LOAD на каждом разделе базы данных, где находятся таблицы. Дополнительную информацию об утилите AutoLoader смотрите в книге *Data Movement Utilities Guide and Reference*.

Во время создания индекса просмотр и последовательная сортировка данных происходят параллельно. При создании индекса DB2 использует как параллелизм ввода/вывода, так и внутриразделький параллелизм. При выполнении оператора CREATE INDEX это помогает увеличить скорость создания индекса во время перезапуска (если индекс помечен как недопустимый) и во время реорганизации данных.

Резервное копирование и восстановление данных - задачи с использованием интенсивного ввода/вывода. DB2 использует в операциях резервного копирования и восстановления как параллелизм ввода/вывода, так и внутриразделький параллелизм. Параллелизм ввода/вывода в резервном копировании используется для параллельного чтения из нескольких контейнеров табличных пространств и для параллельной асинхронной записи на несколько носителей резервных копий. Дополнительную информацию о подключении поддержки параллелизма для этих утилит смотрите в описании команд BACKUP DATABASE и RESTORE DATABASE в книге *Command Reference* .

Аппаратные среды

В этом разделе описываются следующие аппаратные среды:

- Одноразделькая с одним процессором (однопроцессорная система)
- Одноразделькая с несколькими процессорами (SMP)
- Многораздельные конфигурации

- Многораздельная с одним процессором (MPP)
- Многораздельная с несколькими процессорами (кластер сред SMP)
- Логические разделы базы данных (в DB2 Parallel Edition for AIX Версии 1 они назывались Multiple Logical Nodes или MLN)

Для каждой среды обсуждается емкость и масштабируемость. *Емкость* определяется числом пользователей и программ, которые могут получать доступ к базе данных. В значительной мере она определяется памятью, агентами, блокировками, вводом/выводом и управлением памятью. *Масштабируемость* определяет возможности роста базы данных без изменения рабочих характеристик и времени ответов.

Один раздел с одним процессором

Эта среда состоит из памяти, диска и единственного процессора (смотрите рис. 25 на стр. 75). Она может называться по-разному, например, автономной базой данных, базой данных клиент/сервер, последовательной базой данных, однопроцессорной системой и одноузловой или непараллельной средой.

База данных в этой среде обслуживает потребности отдела или небольшого предприятия, где данными и системными ресурсами (в том числе единственным процессором) управляет один менеджер баз данных.

В Табл. 3 на стр. 82 приведены типы параллелизма, наиболее подходящие для использования преимуществ этой аппаратной конфигурации.

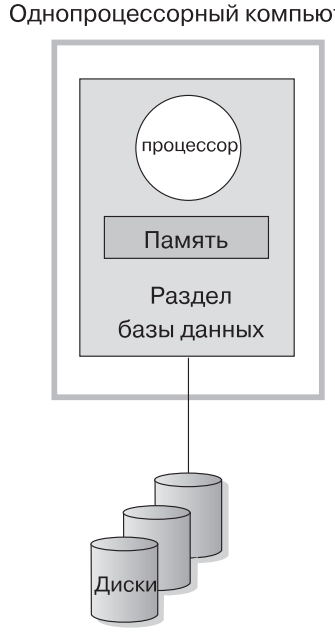


Рисунок 25. Один раздел с одним процессором

Емкость и масштабируемость

В этой среде можно добавлять дополнительные диски. Наличие одного или нескольких серверов ввода/вывода для каждого диска допускает несколько одновременных операций ввода/вывода. Кроме того, в эту среду можно добавить дополнительное место на жестком диске.

Однопроцессорная система ограничивается тем дисковым пространством, с которым может справиться процессор. Однако с увеличением рабочей нагрузки единственный процессор может оказаться не в состоянии обработать запросы пользователей за приемлемое время, как бы вы ни наращивали другие компоненты, например, памяти или диска. При достижении порога емкости или масштабируемости можно рассмотреть переход к одноразделной среде с несколькими процессорами.

Один раздел с несколькими процессорами

Эта среда обычно состоит из нескольких процессоров одинаковой мощности на одном компьютере (смотрите рис. 26 на стр. 76) и называется *симметричной мультипроцессорной системой (SMP)*. При этом такие ресурсы, как дисковое пространство и память, *используются совместно*.

Если есть несколько процессоров, различные операции базы данных могут быть выполнены быстрее. Чтобы увеличить скорость обработки, DB2 может также

поделить обработку одного запроса между несколькими доступными процессорами. Преимущество нескольких процессоров может использоваться и другими операциями, например при загрузке данных, резервном копировании и восстановлении табличных пространств, а также при создании индексов для существующих данных.

В Табл. 3 на стр. 82 приведены типы параллелизма, наиболее подходящие для использования преимуществ этой аппаратной конфигурации.

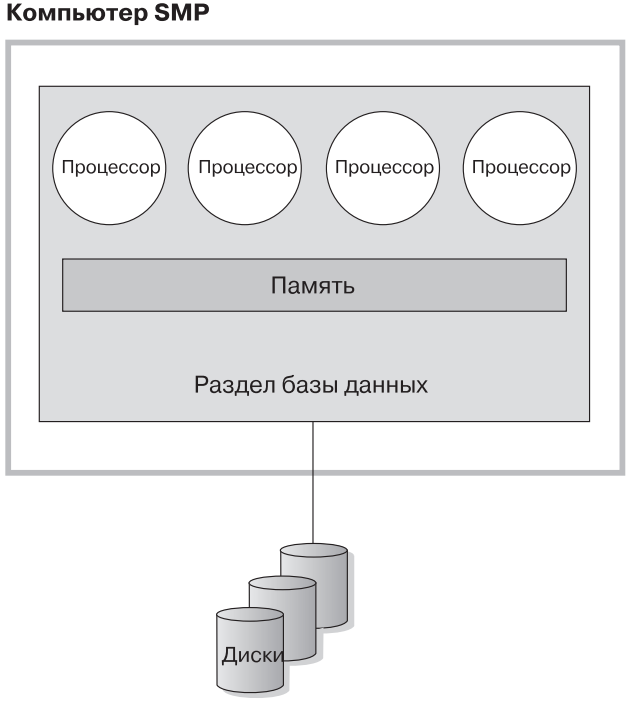


Рисунок 26. Симметричная мультипроцессорная система с одноразделной базой данных

Емкость и масштабируемость

В этой среде можно добавлять дополнительные процессоры. Однако, поскольку к одним и тем же данным могут пытаться обратиться разные процессоры, для этой среды с ростом числа операций могут появиться ограничения. Эффективное совместное использование всех данных базы данных достигается через совместно используемую память и совместно используемые диски.

Емкость ввода/вывода раздела базы данных, связанного с процессором, можно увеличить, увеличив число дисков. Специально для обработки запросов ввода/вывода можно установить серверы ввода/вывода. Наличие одного или нескольких серверов ввода/вывода для каждого диска допускает несколько одновременных операций ввода/вывода.

При достижении порога емкости или масштабируемости можно рассмотреть переход к среде с несколькими разделами.

Многораздельные конфигурации

Базу данных можно разделить на несколько разделов, каждый на своем компьютере. Несколько компьютеров с несколькими разделами базы данных можно сгруппировать вместе. В этом разделе описываются следующие конфигурации разделов:

- Многораздельная в системах с одним процессором
- Многораздельная в системах с несколькими процессорами
- Логические разделы базы данных

Несколько однопроцессорных разделов

В этой среде существует несколько разделов базы данных. Каждый раздел располагается на своем компьютере и у него есть свой процессор, память и диски (смотрите рис. 27 на стр. 78). Все компьютеры соединяются средствами связи. Эта среда может называться по-разному, например, кластером, кластером однопроцессорных систем, средой широкомасштабной параллельной обработки (MPP) и конфигурацией с архитектурой ″ничто не используется совместно″. Последнее название точно отражает распределение ресурсов в этой среде. В отличие от среды SMP, в среде MPP память и диски не используются совместно. Среда MPP устраняет ограничения, которые ставит совместное использование памяти и дисков.

Среда многораздельных баз данных позволяет базе данных оставаться логическим целым, несмотря на то, что она физически разделена на несколько разделов. Факт, что данные разбиты на разделы, остается незаметен для большинства пользователей. Работа может быть поделена между менеджерами баз данных; каждый менеджер баз данных в каждом разделе работает со своей частью базы данных.

В Табл. 3 на стр. 82 приведены типы параллелизма, наиболее подходящие для использования преимуществ этой аппаратной конфигурации.

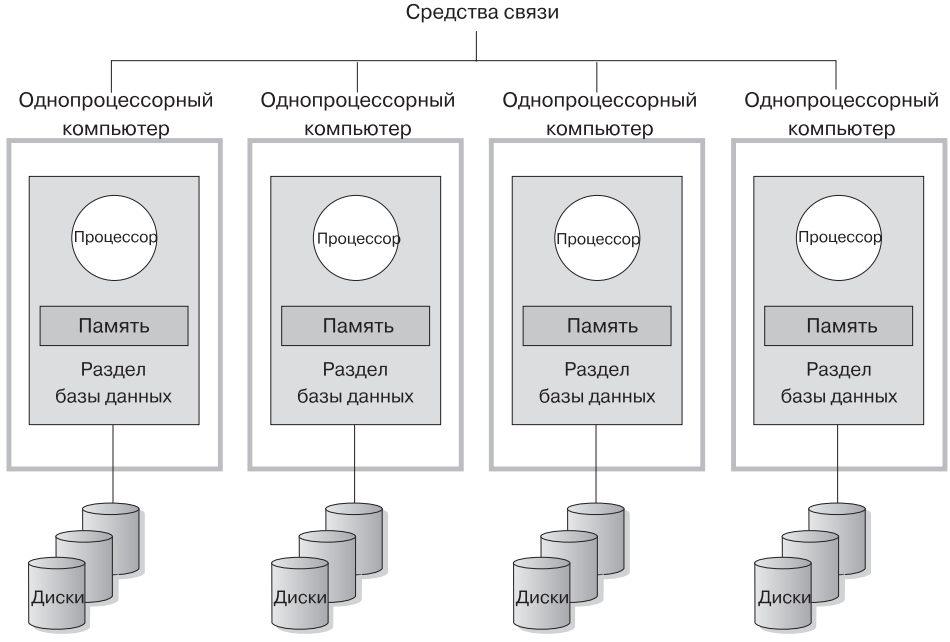


Рисунок 27. Система широкомасштабной параллельной обработки

Емкость и масштабируемость: В этой среде в конфигурацию можно добавлять дополнительные разделы (узлы) базы данных. На некоторых платформах, например RS/6000 SP, можно использовать до 512 узлов. Однако могут существовать практические пределы управления большим числом компьютеров и экземпляров.

При достижении порога емкости или масштабируемости можно рассмотреть переход к системе, где у каждого раздела будет несколько процессоров.

Несколько многопроцессорных разделов

Альтернатива конфигурации, в которой каждый раздел имеет единственный процессор - это конфигурация, где раздел использует несколько процессоров. Такую среду называют *кластером SMP* (смотрите рис. 28 на стр. 79).

Такая среда сочетает преимущества SMP и параллелизма MPP. Это значит, что запрос может обрабатываться на одном разделе несколькими процессорами. Кроме того, запрос может выполняться параллельно на нескольких разделах.

В Табл. 3 на стр. 82 приведены типы параллелизма, наиболее подходящие для использования преимуществ этой аппаратной конфигурации.

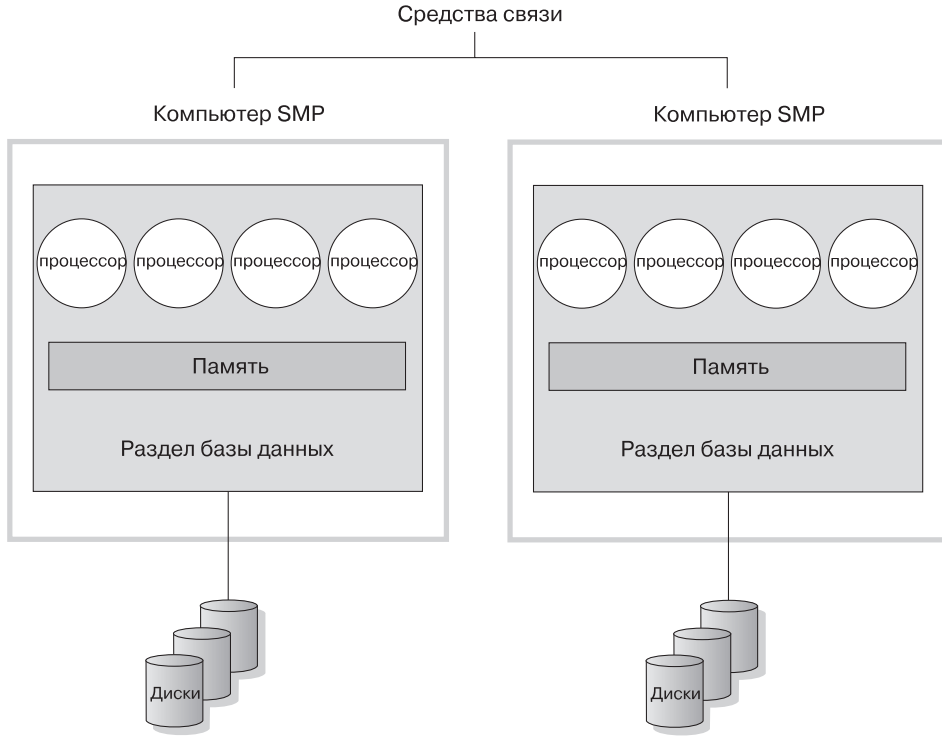


Рисунок 28. Кластер сред SMP

Емкость и масштабируемость: В этой среде можно добавлять дополнительные разделы базы данных.

Логические разделы базы данных

Логический раздел базы данных отличается от физического раздела тем, что он управляет не всем компьютером. Хотя ресурсы компьютера общие, разделы базы данных не используют их совместно. Процессоры используются совместно, а диски и память разделены.

Логические разделы базы данных обеспечивают масштабируемость. Несколько менеджеров баз данных, работающие в нескольких логических разделах, могут полнее использовать доступные ресурсы, чем один менеджер баз данных. На рис. 29 на стр. 80 показано, что на компьютере SMP можно добиться масштабируемости, добавив дополнительные разделы; это характерно для компьютеров с несколькими процессорами. Разбив базу данных на разделы, можно будет управлять каждым разделом и восстанавливать его отдельно от остальных.

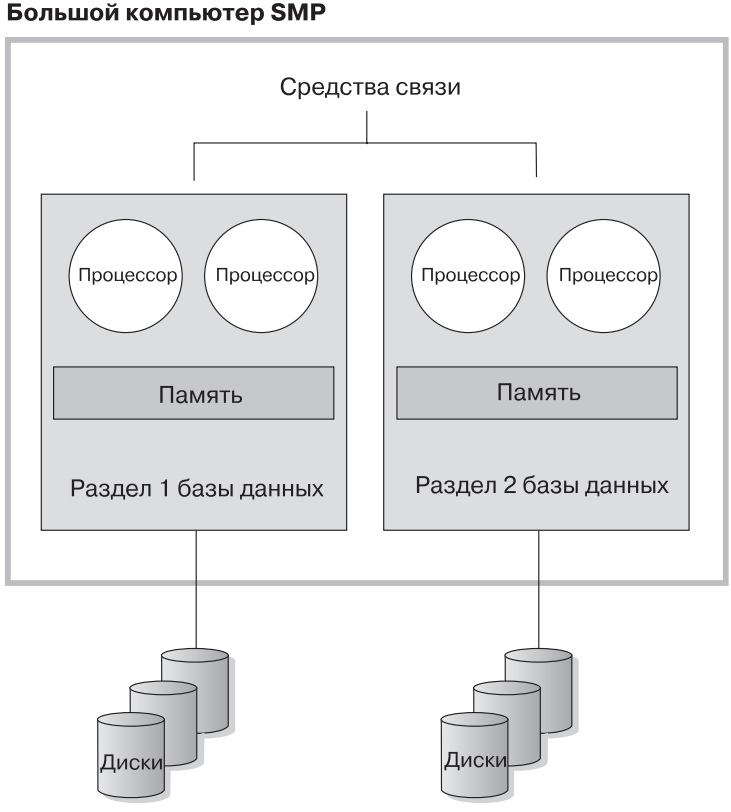


Рисунок 29. Многораздельная база данных, симметричная мультипроцессорная система

На рис. 30 на стр. 81 показано, как распространить конфигурацию, показанную на рис. 29, на несколько компьютеров, чтобы увеличить мощность обработки.

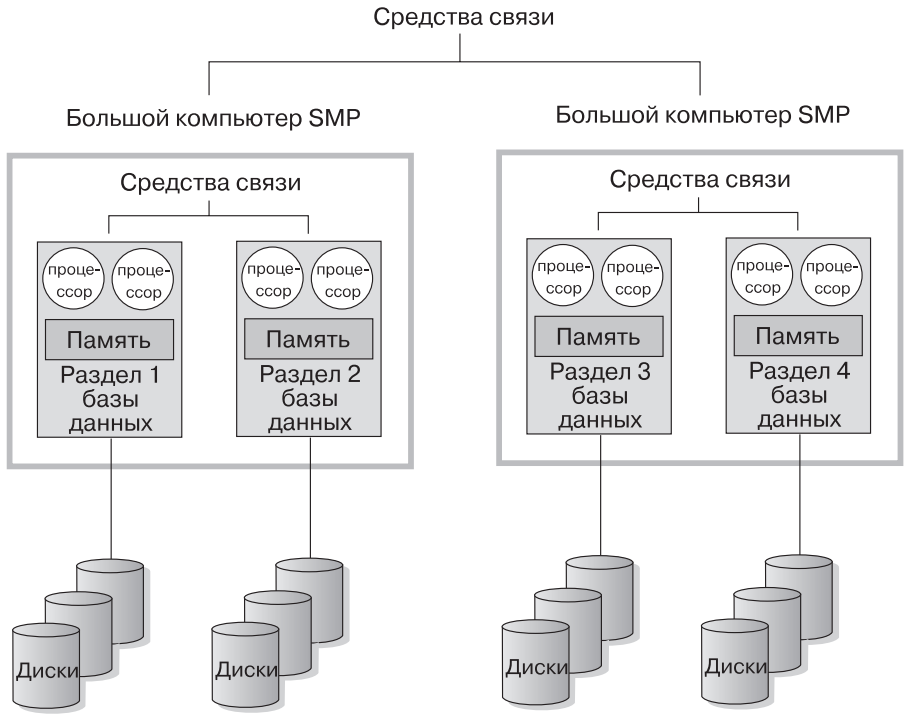


Рисунок 30. Многораздельная база данных, симметричные мультипроцессорные системы, кластеризованные совместно

В Табл. 3 на стр. 82 приведены типы параллелизма, наиболее подходящие для использования преимуществ этой аппаратной среды.

Примечание: Возможность сконфигурировать несколько разделов на одном компьютере (независимо от числа процессоров) дает большую гибкость при разработке конфигураций с высокой доступностью к данным и стратегий восстановления после сбоев. При сбое компьютера раздел баз данных может быть автоматически перемещен и перезапущен на другом компьютере, где уже работает другой раздел этой базы данных. Дополнительную информацию смотрите в разделе “Глава 11. Проектирование высокой доступности” на стр. 219.

Сводка по вариантам параллелизма для каждой аппаратной среды

В следующей таблице представлена сводная информация о типах параллелизма, подходящих для использования преимуществ различных аппаратных сред.

Таблица 3. Типы параллелизма, возможные в каждой аппаратной среде

Аппаратная среда	**Параллелизм ввода/вывода**	**Внутренний параллелизм**	
		Внутрираздельн. параллелизм	**Межраздельный параллелизм**
Один раздел, один процессор	Да	Нет(1)	Нет
Один раздел, несколько процессоров (SMP)	Да	Да	Нет
Несколько разделов, один процессор (MPP)	Да	Нет(1)	Да
Несколько разделов, несколько процессоров (кластер сред SMP)	Да	Да	Да
Логические разделы базы данных	Да	Да	Да
Примечание: (1) Может оказаться полезным установить для степени параллелизма (с помощью одного из параметров конфигурации) значение больше единицы даже в однопроцессорной системе, особенно если процессор не полностью используется выполняемыми запросами (например, если есть ограничения со стороны ввода/вывода).			

Глава 5. О хранилищах данных

DB2 Universal Database содержит Центр хранилищ данных, компонент, который автоматизирует работу хранилищ. Центр хранилищ данных можно использовать для определения данных, помещаемых в хранилище. Затем Центр хранилищ можно применить для автоматического регулярного обновления данных в хранилище.

В этом разделе описываются работа и задачи хранилища данных. Более подробную информацию о хранилищах данных и информацию об использовании Центра хранилищ данных можно найти в книге *Data Warehouse Center Administration Guide* и в справке Центра хранилищ данных.

Что такое хранилище данных?

Системы, содержащие *рабочие данные* – данные ежедневных транзакций вашей деятельности – могут дать ценную информацию для аналитиков. Например, аналитики могут использовать информацию о продажах продуктов по регионам и по временам года для поиска аномалий или планирования будущих продаж.

Однако если аналитики обращаются к рабочим данным непосредственно, возникает несколько проблем:

- У них может не быть необходимого опыта для построения запросов к рабочей базе данных. Например, для запросов к базам данных IMS нужна прикладная программа, которая использует особый тип языка работы с данными. Обычно программисты, обладающие нужными знаниями для построения запросов к рабочей базе данных, полный день заняты на сопровождении этой базы и программ работы с ней.
- Для многих рабочих баз данных, например, для базы банка, критичной является производительность. Такая система не может обслуживать пользователей, делающих промежуточные запросы.
- Формат рабочих данных обычно не вполне удобен для использования аналитиками. Например, сводные данные по продуктам, регионам и временам года для анализа будут намного более полезны, чем необработанные данные.

Эти проблемы решают хранилища данных. В *хранилище данных* вы можете поместить *информационные данные* – данные, извлеченные из рабочих данных и преобразованные для принятия решений конечными пользователями. Например, средствами хранилища данных можно скопировать данные продаж из рабочей базы данных, выполнить вычисления для составления по этой информации сводных данных и записать полученные сводные данные в другую базу данных,

отдельную от рабочей. Конечные пользователи могут строить запросы к этой отдельной базе данных (*хранилищу*), не затрагивая рабочих баз данных.

В следующем разделе описываются объекты (тематические области, источники хранилища, потребители хранилища, узлы агентов, шаги и процессы), которые можно использовать при создании и поддержке хранилища данных.

Тематические области

Тематическая область идентифицирует и группирует процессы, связанные с логической областью деятельности. Например, при построении хранилища данных по маркетингу и продажам можно определить тематическую область Продажи и тематическую область Маркетинг. Затем можно добавить процессы, связанные с продажами, в тематическую область Продажи. Подобным образом определения, связанные с данными маркетинга, можно добавить в тематическую область Маркетинг.

Источники хранилища

Источники хранилища данных идентифицируют таблицы и файлы, которые будут обеспечивать данными хранилище. Центр хранилищ данных использует спецификации в источниках хранилища для обращения к данным и для получения эти данных. Источником может быть практически любой реляционный или нереляционный источник (таблица, производная таблица или файл), который подключен к вашему хранилищу.

Потребители хранилища

Потребители хранилища - это таблицы или файлы баз данных, содержащие данные, преобразованные для удобства использования их конечными пользователями. Потребители хранилища могут, как и источники, давать данные для шагов Центра хранилищ данных.

Агенты хранилища и узлы агентов

Агенты Центра хранилищ данных управляют потоком данных между источниками данных и потребителями данных. Агенты доступны в операционных системах Windows NT, AIX, OS/2, OS/390, OS/400 и SUN Solaris. Для связи с другими базами данных агенты используют драйверы Open Database Connectivity (ODBC) или DB2 CLI.

Передача данных между источниками и потребителями хранилища может обслуживаться несколькими агентами. Число используемых агентов определяется существующей конфигурацией связи и планируемым объемом данных, передаваемых через хранилище. Если для нескольких процессов, выполняемых одновременно, требуется один и тот же агент, можно сгенерировать дополнительные экземпляры этого агента.

Агенты могут быть локальными или удаленными. *Локальный агент хранилища* устанавливается на том же компьютере, что и сервер хранилища. *Удаленный агент хранилища* устанавливается на другом компьютере, откуда доступна связь с сервером хранилища.

Узел агента - логическое имя рабочей станции, на которой установлена программа агента. Имя узла агента не надо путать с именем хоста TCP/IP. У одного физического компьютера может быть только одно имя хоста TCP/IP. Однако на одном компьютере можно определить несколько узлов агентов. Логическое имя служит для идентификации каждого узла агента.

Узел агента по умолчанию, называемый Default VW AgentSite, - это локальный агент в Windows NT, который Центр хранилищ данных определяет во время установки управляющей базы данных хранилища.

Шаги и процессы

Шаг - это логический объект в Центре хранилищ данных, который определяет:

- Структуру выходной таблицы или файла
- Механизм (SQL или программу) для заполнения выходной таблицы или файла
- Расписание, по которому заполняется выходная таблица или файл.

Шаги перемещают и преобразуют данные с помощью операторов SQL или вызовов программ. При запуске шага выполняется передача данных между источником хранилища и потребителем хранилища и все необходимые преобразования этих данных.

Процесс состоит из серий шагов, выполняющих задачи по преобразованию и перемещению данных. В общем случае, процесс заполняет потребитель хранилища в базе данных хранилища, извлекая данные из одного или нескольких источников, которыми могут быть таблицы или файлы базы данных. Однако можно определить процесс для запуска программ, который не задает ни источников, ни потребителей.

Шаг можно запускать по требованию или запланировать его запуск на заданное время. Можно спланировать единовременный запуск шага или запускать его периодически, например по пятницам. Можно также спланировать запуск шагов в последовательности так, чтобы по завершении одного шага начинал выполняться другой. Можно спланировать запуск шагов в зависимости от завершения (успешного или неудачного) какого-либо другого шага. Если составляется расписание процесса, первый шаг в процессе запускается во время, указанное в расписании.

При выполнении шаг или процесс может сохранять данные:

- Заменяя все данные в потребителе хранилища новыми.

- Добавляя новые данные к существующим.
- Добавляя отдельный вариант данных.

Например, требуется, чтобы Центр хранилищ данных выполнил следующие задачи:

1. Извлек данные из различных баз данных.
2. Преобразовал данные в единый формат.
3. Записал данные в таблицу в хранилище данных.

В этом случае нужно создать процесс, содержащий отдельные шаги. Каждый шаг будет выполнять отдельную задачу, например извлекать данные из баз данных или преобразовывать их в нужный формат. Затем нужно воспользоваться другим шагом для заполнения таблицы назначения, содержащей преобразованные данные.

В следующих разделах описываются разные типы шагов, которые вы найдете в Центре хранилищ данных. Дополнительную информацию о шагах смотрите в книге *Data Warehouse Center Administration Guide*.

Шаги SQL

Шаг SQL использует оператор SQL SELECT для извлечения данных из источника хранилища и генерирует оператор INSERT для вставки данных в таблицу назначения хранилища.

Шаги программ

Программные шаги бывают различных типов: программы DB2 for AS/400, DB2 for OS/390, DB2 for UDB, Visual Warehouse 5.2 DB2, сервера OLAP, программы работы с файлами и шаги репликации. Эти шаги запускают определенные программы и утилиты.

Преобразователи

Преобразователи - это хранимые процедуры и пользовательские функции, задающие специальные статистические преобразования или преобразования хранилища данных, которые можно использовать для преобразования данных. Преобразователи можно использовать для очистки, инвертирования и поворота данных, генерирования первичных ключей и таблиц периодов, а также для вычисления различных статистических показателей.

В шаге преобразователя задается один из статистических преобразователей или преобразователей хранилища. При запуске процесса шаг преобразователя записывает данные в один или несколько потребителей хранилища.

Шаги пользовательских программ

Шаг пользовательской программы - это логический объект в Центре хранилищ данных, представляющий программу, которую должен запускать Центр хранилищ данных. Шаг пользовательской программы может запускаться агентом хранилища:

- Во время заполнения потребителя хранилища.
- После заполнения потребителя хранилища.
- Самостоятельно.

Например, можно написать пользовательскую программу, которая будет выполнять следующие процессы:

1. Экспорт данных из таблицы.
2. Обработка данных.
3. Запись данных в промежуточный выходной ресурс или в потребитель хранилища.

Задачи хранилищ

При создании хранилища данных решаются следующие задачи:

- Определение тематической области, идентифицирующей и группирующей процессы, которые будут использоваться в хранилище.
- Исследование данных источника (или рабочих данных) и определение источников хранилища.
- Создание базы данных для использования в качестве хранилища и определение потребителей хранилища.
- Задание способов перемещения и преобразования данных источника в формат для базы данных хранилища с помощью определения процесса.
- Проверка определенных вами шагов и составление расписания их автоматического запуска.
- Управление хранилищем с использованием задаваемой защиты и мониторинга базы данных.
- Создание каталога данных в хранилище, если у вас есть пакет DB2 Warehouse Manager. Каталог данных - это база данных, содержащая деловые метаданные. Метаданные помогают пользователям идентифицировать и находить данные и информацию, доступную для них в организации. Конечные пользователи хранилища могут проводить поиск в этом каталоге, чтобы определять, к каким таблицам направлять запросы.
- Определение модели схемы типа звезда для данных в хранилище. Схема типа звезда - это особая структура, которая состоит из нескольких таблиц ассоциаций (описывающих различные аспекты деятельности предприятия) и одной таблицы фактов (содержащей соответствующие фактические материалы). Например, если ваша фирма производит прохладительные напитки, некоторые таблицы ассоциаций могут описывать продукты, рынки и

сроки. Таблица фактов содержит информацию транзакций о продуктах, заказываемых в каждом регионе в зависимости от времени года.

- Объединение таблицы фактов и таблиц ассоциаций для объединения подробных сведений из таблиц ассоциаций с информацией о заказах. Например, можно объединить таблицу ассоциаций продуктов с таблицей фактов, чтобы добавить к ней информацию о том, как каждый продукт включался в пакеты заказов.

Об этих и других задачах можно узнать подробнее, воспользовавшись учебником *Business Intelligence Tutorial*, просмотрев краткий обзор *DB2 Universal Database Quick Tour* или прочитав книгу *Data Warehouse Center Administration Guide*.

Глава 6. О модуле Spatial Extender

В этом разделе кратко описан модуль Spatial Extender, объяснены его задачи и рассказано о данных, которые он обрабатывает. Подробную информацию об использовании Spatial Extender смотрите в книге *Spatial Extender User's Guide and Reference*.

Назначение Spatial Extender

Spatial Extender используется для создания *географической информационной системы* (GIS): комплекса объектов, данных и программ, позволяющих генерировать и анализировать пространственные данные о географических объектах. К *географическим объектам* относятся естественные и искусственные объекты, образующие земную поверхность. Они составляют природную (реки, леса, горы и пустыни) и культурную (города, населенные пункты, предприятия, ориентиры на местности и т.п.) среду.

К *пространственным данным* относятся:

- Положение географических объектов по отношению к окрестностям (например, точки в городе, где расположены больницы и клиники, или близость населенных пунктов к сейсмическим зонам)
- Пути связи между географическими объектами (например, информация, что определенная речная система находится в некотором районе или что мосты в данном районе пересекают ее притоки)
- Измерения, относящиеся к одному или нескольким географическим объектам (например, расстояние от здания предприятия до границ его участка или длина периметра заповедника)

Пространственные данные, как сами по себе, так и в сочетании с традиционным выводом базы данных, могут помочь в разработке проектов и принятии деловых и политических решений. Предположим, что управляющему по социальному обеспечению округа нужно проверить, что получатели социальной помощи и кандидаты на ее получение действительно проживают на территории обслуживания этого округа. Spatial Extender может извлечь эти сведения из местоположения обслуживаемой территории и адресов кандидатов и получателей.

Или предположим, что владелец сети ресторанов желает распространить свой бизнес на близлежащие города. Чтобы определить, где открыть новые рестораны, ему нужно получить ответы на такие вопросы как, например: Где в этих городах концентрируется тип людей, которые будут часто посещать мои рестораны? Где находятся главные магистрали? Где самые низкие показатели

преступности? Где находятся рестораны моих конкурентов? Чтобы ответить на такие вопросы, Spatial Extender может дать наглядное представление пространственных данных, а лежащая в основе система управления базой данных может сгенерировать пометки и текст для объяснения вывода.

Данные о географических объектах

В этом разделе представлен обзор данных, которые вы генерируете, храните и обрабатываете для получения пространственной информации. Рассматриваемые темы:

- Как данные описывают географические объекты
- Характер пространственных данных
- Способы получения пространственных данных

Как данные представляют географические объекты

В Spatial Extender географический объект может быть представлен строкой в таблице или производной таблице или частью такой строки. Например, рассмотрим следующие географические объекты: офисные и жилые здания. На рис. 31 каждая строка таблицы BRANCHES описывает филиал банка. Другой вариант - каждая строка таблицы CUSTOMERS в целом описывает клиента банка, но часть этой строки (а именно столбцы с адресами клиентов) можно рассматривать как представление места жительства клиента.
В этих таблицах содержатся данные, идентифицирующие и описывающие

Таблица филиалов BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141

Таблица клиентов CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141	A	A

Рисунок 31. Строка таблицы, описывающая географический объект; строка таблицы, данные об адресе в которой описывают географический объект. Строка данных в таблице BRANCHES описывает филиал банка. Данные об адресах в таблице CUSTOMERS описывают место жительства клиента.

филиалы банков и клиентов. Такие данные называются *данными атрибутов*.

Подмножество данных атрибутов (значения, описывающие филиалы банков и адреса клиентов) могут быть преобразованы в поля, содержащие пространственные данные. Например, на рисунке указан следующий адрес одного из филиалов банка: 92467 Airzone Blvd., San Jose CA 95141. Адрес клиента - 9 Concourt Circle, San Jose CA 95141. Spatial Extender может

преобразовать эти адреса в значения, указывающие положение филиала банка и дома клиента относительно окрестностей. На рис. 32 показаны таблицы BRANCHES и CUSTOMERS с новыми столбцами, содержащими такие значения.

Таблица филиалов BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	

Таблица клиентов CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141		A	A

Рисунок 32. Таблицы с добавленными пространственными столбцами. В каждой таблице столбец LOCATION будет содержать координаты, соответствующие этим адресам.

Если адреса и подобные идентификаторы используются в качестве отправной точки для пространственной информации, они называются *исходными данными*. Так как полученные из них значения поставляют пространственную информацию, эти значения называются *пространственными данными*. Следующий раздел рассказывает о пространственных данных и знакомит со связанными с ними типами данных.

Характер пространственных данных

Многие пространственные данные составляются из координат. *Координата* - это число, обозначающее положение относительно начала отсчета. Например, широта - это координата, обозначающая положение относительно экватора. Долгота - координата, обозначающая положение относительно гринвичского меридиана. Таким образом, положение Йелоустонского национального парка обозначается его широтой (44,45 градусов к северу от экватора) и долготой (110,40 градусов к западу от гринвичского меридиана).

Широта, долгота, их начала отсчета и другие относящиеся к ним параметры вместе называются *системой координат*. Существуют системы координат на основе иных значений, отличающихся от широты и долготы. В этих системах координат есть собственные измерения положения, начала отсчета и дополнительные отличительные параметры.

Простейший элемент пространственных данных состоит из двух координат, определяющих положение одиночной географической точки. Термин *элемент данных* относится к значению или значениям, занимающим ячейку в реляционной таблице. Другой элемент данных может описываться несколькими координатами, которые определяют линейный путь, например дорогу или реку.

Третий вид состоит из координат, определяющих периметр области, например, границы земельного участка или поймы.

Каждый элемент этих пространственных данных является примером одного из типов пространственных данных. Тип данных для двух координат, обозначающих положение - ST_Point, тип данных для координат, обозначающих линейные пути - ST_LineString, и тип данных для координат, обозначающих периметры - ST_Polygon. Эти типы вместе с другими типами пространственных данных являются структурированными типами, принадлежащими к отдельной иерархии.

Откуда берутся пространственные данные

Пространственные данные можно получить:

- Из данных атрибутов
- Из других пространственных данных
- Посредством импорта

Использование данных атрибутов в качестве исходных данных

Получение данных из данных атрибутов (например, из адресов) называется *геокодированием*. На рис. 32 на стр. 91 показаны два столбца (один в таблице BRANCHES, а второй - в таблице CUSTOMERS), предназначенные для пространственных данных. Представим, что модуль Spatial Extender выполнил геокодирование адресов в этих таблицах и поместил результаты (координаты, соответствующие адресам) в эти столбцы. Этот результат показан на рис. 33.

Таблица филиалов BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	1653 3094

Таблица клиентов CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141	953 1527	A	A

Рисунок 33. Таблицы с пространственными данными, полученными из исходных данных. Столбец LOCATION в таблице CUSTOMERS содержит координаты, полученные геокодировщиком из адреса в столбцах ADDRESS, CITY, STATE и ZIP. Аналогично столбец LOCATION в таблице BRANCHES содержит координаты, полученные геокодировщиком из адреса в столбцах ADDRESS, CITY, STATE и ZIP данной таблицы.

Для геокодирования данных атрибутов и помещения полученных пространственных данных в столбцы Spatial Extender использует функцию, которая называется *геокодировщик*.

Использование в качестве исходных данных других пространственных данных

Пространственные данные можно сгенерировать не только из данных атрибутов, но также и из других пространственных данных. Предположим, что банку, филиалы которого определены в таблице BRANCHES, нужно знать, сколько клиентов находятся в пределах пяти миль от каждого филиала. Прежде чем банк сможет получить эту информацию из базы данных, он должен дать определение зоны, лежащей внутри пятимильного радиуса вокруг каждого филиала. Такое определение может создать функция Spatial Extender ST_Buffer. Используя в качестве входных данных координаты каждого филиала, эта функция может сгенерировать координаты, разграничивающие периметры нужных зон. На рис. 34 показана таблица BRANCHES с информацией, поставляемой функцией ST_Buffer.

Таблица филиалов BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	SALES_AREA
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	1653 3094	1002 2001, 1192 3564, 2502 3415, 1915 3394, 1002 2001

Рисунок 34. Таблица, содержащая новые пространственные данные, полученные из существующих пространственных данных. Координаты в столбце SALES_AREA были получены функцией ST_Buffer из координат в столбце LOCATION.

Помимо функции ST_Buffer модуль Spatial Extender обеспечивает несколько других функций, которые могут получать новые пространственные данные из существующих пространственных данных.

Импорт пространственных данных

Третий способ получения пространственных данных - их импорт из файлов одного из форматов, поддерживаемых Spatial Extender. Эти файлы содержат данные, которые обычно применяются для карт: данные о переписи, о затопляемых поймах, сейсмоопасных районах и т.п. Воспользовавшись такими данными в сочетании с вашими пространственными данными, можно расширить доступную вам географическую информацию. Например, если отделу по социальной работе требуется определить опасности, представляющие угрозу для проживающего населения, можно воспользоваться функцией ST_Buffer для определения зоны данного округа, а затем импортировать данные по затопляемым поймам и сейсмоопасным районам, чтобы посмотреть, какие из этих проблемных областей пересекаются с данной зоной.

Часть 3. Проектирование баз данных

Глава 7. Логическое проектирование базы данных

В этом разделе описываются следующие этапы проектирования базы данных:

- “Решите, какие данные записывать в базу данных”
- “Определите таблицы для каждого типа отношений” на стр. 99
- “Дайте определения столбцов для всех таблиц” на стр. 101
- “Выберите один или несколько столбцов в качестве первичного ключа” на стр. 104
- “Обеспечьте, чтобы совпадающие значения представляли один объект” на стр. 107
- “Рассмотрите возможность нормализации таблиц” на стр. 108
- “Планирование введения ограничений” на стр. 114
- “Другие особенности проектирования базы данных” на стр. 121.

Цель при проектировании базы данных - создать представление вашей среды, которое можно будет легко понять и использовать как основу для расширения. Кроме того, хотелось бы, чтобы структура базы данных помогала поддерживать согласованность и целостность данных. Этого можно достичь, создавая структуру, которая сократит дублирование данных и устранит аномалии, которые могут возникать при изменении базы данных.

Приведенные действия - это часть процесса *логического* проектирования базы данных. Проектирование базы данных - это нелинейный процесс; скорее всего, при разработке структуры базы данных некоторые шаги придется повторять.

Физическая реализация структуры базы данных описана в разделах “Глава 8. Физическая структура базы данных” на стр. 123 и ″Реализация вашего проекта″ в книге *Руководство администратора: Реализация*.

Решите, какие данные записывать в базу данных

Первый шаг в разработке структуры базы данных - это определение типов данных, которые будут храниться в таблицах базы данных. База данных содержит информацию об *объектах* в некоторой организации или на предприятии и отношениях между ними. В реляционной базе данных *объекты* представляются в виде *таблиц*.

Объект - это человек, предмет или понятие, о которых вы хотите хранить информацию. Примеры объектов в таблице примеров - это сотрудники, отделы и проекты. (Описание базы данных примера смотрите в справочнике *SQL Reference*.)

В таблице сотрудников в примере у объекта ″сотрудник″ есть *атрибуты* (свойства), такие как номер сотрудника, имя, отдел, где он работает, и размер заработной платы. Эти свойства отражают *столбцы* EMPNO, FIRSTNME, LASTNAME, WORKDEPT и SALARY.

Экземпляр объекта ″сотрудник″ состоит из значений во всех столбцах, соответствующих одному сотруднику. У каждого сотрудника есть уникальный номер (EMPNO), который можно использовать для идентификации экземпляра объекта ″сотрудник″. Каждая строка в таблице представляет экземпляр объекта или отношения. Например, в следующей таблице значения в первой строке описывают сотрудника Haas.

Таблица 4. Экземпляры объекта сотрудник и их атрибуты

EMPNO	FIRSTNME	LASTNAME	WORKDEPT	JOB
000010	Christine	Haas	A00	Президент
000020	Michael	Thompson	B01	Начальник
000120	Sean	O’Connell	A00	Клерк
000130	Dolores	Quintana	C01	Аналитик
000030	Sally	Kwan	C01	Начальник
000140	Heather	Nicholls	C01	Аналитик
000170	Masatoshi	Yoshimura	D11	Проектировщик

Растет потребность в поддержке нетрадиционных применений баз данных, например, мультимедийных. Можно рассматривать атрибуты, которые являются мультимедийными объектами, такими как документы, видеозаписи или мультимедийные данные, изображения и голос.

Каждый столбец строки в таблице некоторым образом связан со всеми остальными столбцами этой строки. Вот некоторые отношения, фигурирующие в таблицах примера:

- Сотрудники распределены по отделам
  - Долорес Кинтана работает в отделе C01
- Сотрудники работают на определенных должностях
  - Долорес работает аналитиком
- Отделы подотчетны другим отделам
  - Отдел C01 подотчетен отделу A00
  - Отдел B01 подотчетен отделу A00
- Сотрудники работают над проектами
  - Долорес и Хитер заняты в проекте IF1000
- Сотрудники руководят отделами
  - Салли руководит отделом C01.

″Сотрудник″ и ″отдел″ - это объекты; ″Салли Хван″ - это часть экземпляра объекта ″сотрудник″, а ″C01″ - это часть экземпляра объекта ″отдел″. Такое же отношение применяется к этим столбцам во всех строках таблицы. Например, одна строка таблицы выражает отношение, что Салли Хван руководит отделом C01; другая - отношение, что Син О'Коннелл - клерк в отделе A00.

Ответ на вопрос ″Какую информацию поместить в таблицу?″ зависит от выражаемых отношений, требуемой гибкости и скорости получения данных.

Кроме отношений между объектами вашей организации, вам надо определить информацию другого типа, например логические правила, которые надо применять к этим данным.

Определите таблицы для каждого типа отношений

В базе данных может быть определено несколько типов отношений. Рассмотрим возможные отношения между сотрудниками и отделами. Сотрудник может работать только в одном отделе - это *однозначное* отношение для сотрудника. Но в одном отделе может быть много сотрудников - это *многозначное* отношение для отдела. Отношение между отделами и сотрудниками - это отношение типа *один-с-многими*. В этом разделе рассмотрены следующие типы отношений:

- "Отношения один-с-многими и многие-с-одним"
- "Связи Многие-с-многими" на стр. 100
- "Отношение один-с-одним" на стр. 101

Отношения один-с-многими и многие-с-одним

Чтобы определить таблицы для каждого отношения один-с-многими и многие-с-одним:

1. Соберите все отношения, для которых в качестве ″многих″ выступают одинаковые объекты.
2. Определите одну таблицу для всех отношений этой группы.

В приведенном ниже примере в качестве ″многих″ для первого и второго отношения выступают сотрудники, поэтому мы определяем таблицу сотрудников - EMPLOYEE.

Таблица 5. Отношение Многие-с-одним

Объект	Отношение	Объект
Сотрудники	работают в	отделах
Сотрудники	работают на	должностях
Отделы	подотчетны	(управляющим) отделам

Для третьей связи в качестве ″многих″ выступают отделы, поэтому мы определяем таблицу отделов - DEPARTMENT.

Ниже показано, как эти отношения отражаются в таблицах:

Таблица EMPLOYEE:

EMPNO	WORKDEPT	JOB
000010	A00	Президент
000020	B01	Начальник
000120	A00	Клерк
000130	C01	Аналитик
000030	C01	Начальник
000140	C01	Аналитик
000170	D11	Проектировщик

Таблица DEPARTMENT:

DEPTNO	ADMRDEPT
C01	A00
D01	A00
D11	D01

Связи Многие-с-многими

Многозначное для обеих сторон отношение - это отношение многие-со-многими. Один сотрудник может быть занят в нескольких проектах, и над одним проектом может работать несколько сотрудников. На оба вопроса: ″Над чем работает Долорес Кинтана?″ и ″Кто работает над проектом IF1000?″ можно дать несколько ответов. Отношение многие-со-многими может быть выражена таблицей со столбцами для каждого объекта (″сотрудников″ и ″проектов″), как показано в следующем примере.

Следующий пример показывает, как отношение многие-с-многими (сотрудник может работать над несколькими проектами, и над одним проектом может работать несколько сотрудников) выражается в таблице:

Таблица деятельности сотрудников (EMP_ACT):

EMPNO	PROJNO
000030	IF1000
000030	IF2000
000130	IF1000
000140	IF2000
000250	AD3112

Отношение один-с-одним

Отношения один-с-одним однозначны в обоих направлениях. Начальник управляет только одним отделом, а у отдела есть только один начальник. На каждый из вопросов: ″Кто начальник отдела C01?″ и ″Каким отделом руководит Салли Хван?″ можно дать однозначный ответ. Это отношение можно отразить в любой из таблиц DEPARTMENT или EMPLOYEE. Поскольку у каждого отдела есть начальник, но не все сотрудники - начальники, самый логичный вариант - добавить начальников в таблицу DEPARTMENT, как показано в следующем примере.

В следующем примере показано, как отношение один-с-одним отражается в таблице:

Таблица DEPARTMENT:

DEPTNO	MGRNO
A00	000010
B01	000020
D11	000060

Дайте определения столбцов для всех таблиц

Чтобы определить столбец в реляционной таблице:

1. Выберите имя для столбца.

   У каждого столбца должно быть уникальное в своей таблице имя. Выбор имен для столбцов подробно описан в разделе “Приложение B. Правила именования” на стр. 373.

2. Укажите, какой тип данных будет храниться в столбце.

   Тип данных и *длина* указывают тип данных и максимальную длину, допустимые для столбца. Можно выбрать типы данных из предоставляемых

менеджером баз данных или же создать свои собственные пользовательские типы. Информацию о встроенных и пользовательских типах DB2 смотрите в справочнике *SQL Reference*.

Вот примеры категорий типов данных: числовые, символьные строки, двухбайтные (или графические) символьные строки, дата-время и двоичные строки.

Типы данных *большой объект* (LOB) поддерживают объекты мультимедиа, такие как документы, видеозаписи, изображения и голос. Для реализации этих объектов используются следующие типы данных:

- Строка *двоичный большой объект* (BLOB). Примеры двоичных больших объектов - это фотографии сотрудников, голосовые записи и видеозаписи.
- Строка *символьный большой объект* (CLOB), в которой последовательность символов может состоять из одно-, многобайтных символов или их сочетания. Пример символьного большого объекта - резюме сотрудника.
- Строка *двухбайтный символьный большой объект* (DBCLOB), в которой последовательность состоит из двухбайтных символов. Пример DBCLOB - это резюме на японском языке.

Чтобы лучше понять, как поддерживаются большие объекты, посмотрите справочник *SQL Reference*.

Пользовательский тип - это тип, производный от существующего. Вам может понадобиться определить типы, производные от существующего типа и имеющие с ним общие характеристики, но тем не менее несовместимые с ними.

Структурированный тип - это пользовательский тип, структура которого определена в базе данных. Он содержит набор именованных *атрибутов*, у каждого из которых есть тип. Структурированный тип может быть определен как *подтип* другого структурированного типа, называемого его *надтипом*. Подтип наследует все атрибуты своего надтипа, но для него можно определить и дополнительные атрибуты. Набор структурированных типов, относящихся к одному надтипу, называется *иерархией типов*, а надтип, для которого нет надтипа, называется *корневым типом* иерархии типов.

Структурированный тип можно использовать как тип таблицы или производной таблицы. Имена и типы атрибутов структурированного типа вместе с идентификатором объекта становятся именами и типами столбцов этой *типизированной таблицы* или *типизированной производной таблицы*. Строки типизированной таблицы или производной таблицы можно рассматривать как объекты структурированного типа.

Структурированный тип нельзя использовать как тип данных столбца таблицы или производной таблицы. Нельзя также присвоить целый объект структурированного типа переменной хоста в прикладной программе.

Ссылочный тип - это сопровождающий тип для структурированного типа. Как и особый тип, ссылочный тип - это скалярный тип, имеющий то же представление, что и один из встроенных типов данных. Это общее представление - одно для всех типов в иерархии типов. Представление ссылочного типа определено, если создан корневой тип иерархии типов. При использовании ссылочного типа структурированный тип указывается как параметр типа. Этот параметр называется *типом назначения* ссылки.

Назначение ссылки - это всегда строка в типизированной таблице или производной таблице. Когда используется ссылочный тип, у него может быть определена *область действия*. Область действия определяет таблицу (называемую *таблицей назначения*) или производную таблицу (называемую *производной таблицей назначения*) которая содержит строку назначения для ссылки. Таблица или производная таблица назначения должны иметь тот же тип, что и тип назначения ссылочного типа. Экземпляр объекта ссылающегося типа с заданной областью действия однозначно определяет строку в типизированной таблице или производной таблице, называемой его *строкой назначения*.

Пользовательскую функцию можно использовать в различных случаях, в том числе для вызова процедур сравнения и преобразования значений пользовательских типов. Пользовательские функции расширяют и дополняют возможности, предоставляемые встроенными функциями; их можно использовать всюду, где используются встроенные функции. Есть два типа пользовательских функций:

- Внешняя функция, которая написана на одном из языков программирования
- Функция с источником, используемая для вызова других пользовательских функций

Рассмотрим, например, два числовых типа данных - европейский и американский размеры обуви. Оба типа выражают размер обуви, но они несовместимы, так как системы измерения различны и сравнивать значения в разных системах нельзя. Можно вызывать пользовательскую функцию для преобразования одного размера в другой.

Чтобы лучше понять, что такое пользовательские типы, структурированные типы, ссылочные типы и пользовательские функции, посмотрите справочник *SQL Reference*.

3. Укажите, для каких столбцов могут понадобиться значения по умолчанию.

   В некоторых столбцах осмысленные значения могут стоять не во всех строках, потому что:

   - Значение столбца неприменимо к этой строке.

Например, в столбце, содержащем отчество сотрудника, может не стоять значение, если у сотрудника нет отчества.

- Значение применимо, но пока неизвестно.

  Например, столбец MGRNO может не содержать допустимый номер начальника, потому что предыдущий начальник отдела был переведен, а новый - еще не назначен.

В обоих случаях вы можете выбрать, допустить использование значения NULL (специальное значение, указывающее, что значение столбца неизвестно или неприменимо), или использовать по умолчанию другое значение, назначаемое менеджером баз данных или программой.

Значение NULL и значения по умолчанию подробно описаны в справочнике *SQL Reference*.

Выберите один или несколько столбцов в качестве первичного ключа

Ключ - это набор столбцов, которые могут использоваться для идентификации конкретной строки или строк и для доступа к ним. Ключ указывается в описании таблицы, индекса или реляционного ограничения. Один столбец может входить в несколько ключей.

Ключ уникальности - это ключ, на который накладывается ограничение: никакие два его значения не совпадают. Столбец ключа уникальности не может содержать значений NULL. Например, столбец номеров сотрудников может быть определен как ключ уникальности, потому что каждое значение в этом столбце определяет только одного сотрудника. У двух разных сотрудников не может быть одинаковых номеров.

Механизм, используемый для реализации ключа уникальности, называется *индексом уникальности*. Индекс уникальности таблицы - это столбец или упорядоченный набор столбцов, каждое значение которого идентифицирует (функционально определяет) единственную строку. Индекс уникальности может содержать значения NULL.

Первичный ключ - это один из ключей уникальности таблицы, который выбран в качестве ключа первостепенной важности. У таблицы может быть только один первичный ключ.

Для первичного ключа автоматически создается *первичный индекс*. Первичный индекс используется менеджером баз данных для эффективного доступа к строкам таблицы и позволяет менеджеру баз данных поддерживать уникальность первичного ключа. (Для эффективного доступа к данным при обработке запросов можно также определить индексы по столбцам других ключей.)

Если у таблицы нет ″естественного″ ключа уникальности или если для различения строк используется последовательность их появления в базе, может быть полезно использовать отметки времени. (Смотрите также раздел “Определите столбцы идентификации” на стр. 106.)

Первичные ключи для некоторых из таблиц примера:

Таблица	**Ключевой столбец**
Таблица сотрудников	EMPNO
Таблица отделов	DEPTNO
Таблица проектов	PROJNO

В следующем примере показана часть таблицы PROJECT, включающая ее столбец первичного ключа.

Таблица 6. Первичный ключ таблицы PROJECT

PROJNO (Первичный ключ)	**PROJNAME**	**DEPTNO**
MA2100	Автоматизация сварочной линии	D01
MA2110	Программирование сварочной линии	D11

Если в каждом столбце таблицы есть повторяющиеся значения, определить первичный ключ, состоящий только из одного столбца, невозможно. Ключ из нескольких столбцов называется *составным ключом*. Комбинация значений столбцов должна определять единственный объект. Если определить составной ключ сложно, можно создать новый столбец с уникальными значениями.

В следующем примере показан первичный ключ, состоящий из нескольких столбцов (составной ключ):

Таблица 7. Составной первичный ключ для таблицы EMP_ACT

EMPNO (Первичный ключ)	**PROJNO (Первичный ключ)**	**ACTNO (Первичный ключ)**	**EMPTIME**	**EMSTDATE (Первичный ключ)**
000250	AD3112	60	1,0	01.01.1982
000250	AD3112	60	0,5	01.02.1982
000250	AD3112	70	0,5	01.02.1982

Выберите ключевые столбцы

Для определения столбцов, которые разумно выбрать ключевыми, найдите наименьшее число столбцов, которое однозначно определяет объект. Может

найтись несколько подходящих ключей. В Табл. 2 на стр. 26 можно выбирать различные ключи. Каждый из столбцов EMPNO, PHONENO и LASTNAME однозначно определяет сотрудника.

Критерии для выбора первичного ключа - это определенность, уникальность и постоянство:

- Определенность означает, что для каждой строки всегда существует значение первичного ключа.
- Уникальность означает, что значения ключа различны для разных строк.
- Постоянство означает, что значения первичного ключа никогда не изменяются.

Из трех ключей-кандидатов в примере только EMPNO отвечает всем этим критериям. У сотрудника может не быть телефонного номера, когда он начинает работать в компании. Сотрудник может сменить фамилию, и, хотя фамилии случайно могут оказаться уникальными, это не гарантировано. Столбец номеров сотрудников - это наилучший вариант первичного ключа. Сотруднику только однажды присваивается уникальный номер, и этот номер, как правило, не изменяется за время, пока сотрудник остается в компании. Так как у каждого сотрудника должен быть номер, значения в столбце номеров сотрудников являются определенными.

Определите столбцы идентификации

Столбец идентификации дает DB2 возможность автоматически генерировать уникальное числовое значение для каждой строки таблицы. У таблицы может быть только один столбец, определенный с атрибутом идентификации. Примеры столбцов идентификации: порядковый номер, номер сотрудника, номер акции и номер страхового случая.

Значения для столбца идентификации могут генерироваться всегда или по умолчанию.

- DB2 гарантирует уникальность столбца идентификации, определенного как *генерируемый всегда*. Его значения всегда генерируются DB2; программам не позволяется задавать значение явно.
- Если определить столбец идентификации как *генерируемый по умолчанию*, программы смогут явно задавать его значение. Если же значение не указывается, DB2 генерирует его, но в этом случае уникальность не гарантируется. DB2 гарантирует уникальность только среди сгенерированных ей значений. Генерация по умолчанию удобна для распространения данных, при которых содержимое существующей таблицы копируется, или при выгрузке и перезагрузке таблицы.

Столбцы идентификации идеально подходят для генерации уникальных значений первичных ключей. Программы могут использовать столбцы идентификации, чтобы избежать проблем с производительностью и

одновременностью, которые могут возникнуть, если программа генерирует свой собственный счетчик уникальности вне базы данных. Например, обычная реализация на уровне программы - это поддерживать однострочную таблицу, содержащую счетчик. Каждая транзакция блокирует эту таблицу, увеличивает номер, а потом выполняет принятие; это гарантирует, что только одна транзакция за раз может увеличить счетчик. Если же поддерживать счетчик с помощью столбца идентификации, можно достичь гораздо более высокого уровня одновременности, так как счетчик не будет блокироваться транзакциями. Одна не принятая транзакция, увеличившая счетчик, не помешает последующим транзакциям тоже увеличить счетчик.

Счетчик для столбца идентификации увеличивается (или уменьшается) независимо от транзакции. Если некоторая транзакция увеличивает счетчик идентификации два раза, между двумя сгенерированными числами может оказаться промежуток, так как могут найтись другие транзакции, одновременно увеличивающие тот же счетчик идентификации (то есть вставляющие строки в ту же таблицу). Если программа хочет получить последовательные числа из диапазона, она должна получить монопольную блокировку для таблицы, у которой есть столбец идентификации. Принимая такое решение, надо учесть получающуюся потерю одновременности. Более того, может так получиться, что в числах в столбце идентификации появятся промежутки, если для транзакции, сгенерировавшей значение для столбца идентификации, был выполнен откат или значения в базе данных попали в кэш, а база была отключена прежде, чем все они были присвоены.

Последовательные числа, генерируемые столбцом идентификации, обладают следующими дополнительными свойствами:

- Значения могут быть любого точного числового типа данных с масштабом 0, то есть: SMALLINT, INTEGER, BIGINT или DECIMAL с масштабом 0. (Числа с плавающей точкой одинарной и двойной точности считаются приближенными числовыми типами данных.)
- Последовательные значения могут отличаться на любое указанное целочисленное приращение. Приращение по умолчанию - 1.
- Значение счетчика для столбца идентификации восстановимо. Если происходит ошибка, значение счетчика восстанавливается по журналам, таким образом уникальность значений остается гарантированной.
- Значения столбца идентификации могут кэшироваться для лучшей производительности.

Обеспечьте, чтобы совпадающие значения представляли один объект

Возможны несколько таблиц, описывающих атрибуты одного набора объектов. Например, в таблице EMPLOYEE есть номер отдела, в котором работает сотрудник, а в таблице DEPARTMENT - начальник, руководящий отделом с данным номером. Чтобы получить оба набора атрибутов одновременно, можно

объединить эти две таблицы по совпадающему столбцу, как показано в следующем примере. Значения в WORKDEPT и DEPTNO соответствуют одному и тому же объекту, что дает *способ объединения* таблиц DEPARTMENT и EMPLOYEE.

Таблица DEPARTMENT:

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
D21	Административный отдел	000070	D01

Таблица EMPLOYEE:

EMPNO	FIRSTNAME	LASTNAME	WORKDEPT	JOB
000250	Daniel	Smith	D21	Клерк

Если вы получаете информацию об объекте из нескольких таблиц, надо, чтобы совпадающие значения представляли один объект. Имена столбцов объединения могут быть разными (как WORKDEPT и DEPTNO в предыдущем примере) или же одинаковыми (как у столбцов DEPTNO в таблицах отделов и проектов).

Рассмотрите возможность нормализации таблиц

Нормализация помогает устранить избыточность и рассогласованность в данных таблиц. Это процесс сведения таблиц к набору столбцов, в котором все неключевые столбцы зависят от столбца первичного ключа. Если этого не сделать, данные могут стать несогласованными при обновлениях.

В этом разделе коротко рассмотрены правила для первой, второй, третьей и четвертой нормальных форм. Пятая нормальная форма таблицы, которая упоминается во многих книгах по проектированию баз данных, здесь не описана.

Форма **Описание**

Первая
Во всех позициях строк и столбцов стоит по *одному* значению, нигде нет набора значений (смотрите раздел “Первая нормальная форма” на стр. 109).

Вторая
Каждый столбец, не входящий в ключ, зависит от ключа (смотрите раздел “Вторая нормальная форма” на стр. 109).

Третья Каждый неключевой столбец независим от других неключевых столбцов; он зависит только от ключа (смотрите раздел “Третья нормальная форма” на стр. 111).

Четвертая

Ни в одной строке не содержится нескольких независимых многозначных фактов об одном объекте (смотрите раздел “Четвертая нормальная форма” на стр. 113).

Первая нормальная форма

Таблица находится в *первой нормальной форме*, если в каждой ячейке есть только одно значение, нигде нет набора значений. Таблица в первой нормальной форме необязательно соответствует критериям более высоких нормальных форм.

Например, следующая таблица нарушает правило для первой нормальной формы, так как столбец WAREHOUSE содержит несколько значений для одного экземпляра PART.

Таблица 8. Таблица, нарушающая первую нормальную форму

PART (Первичный ключ)	**WAREHOUSE**
P0010	Склад A, Склад B, Склад C
P0020	Склад B, Склад D

В следующем примере те же данные организованы в таблицу в первой нормальной форме.

Таблица 9. Таблица, соответствующая первой нормальной форме

PART (Первичный ключ)	**WAREHOUSE (Первичный ключ)**	**QUANTITY**
P0010	Склад A	400
P0010	Склад B	543
P0010	Склад C	329
P0020	Склад B	200
P0020	Склад D	278

Вторая нормальная форма

Таблица находится во *второй нормальной форме*, если каждый столбец, не входящий в ключ, зависит от *всего* ключа.

Вторая нормальная форма нарушается, если неключевой столбец зависит от *части* составного ключа, как в следующем примере:

Таблица 10. Таблица, нарушающая вторую нормальную форму

PART (Первичный ключ)	WAREHOUSE (Первичный ключ)	QUANTITY	WAREHOUSE_ADDRESS
P0010	Склад A	400	1608 New Field Road
P0010	Склад B	543	4141 Greenway Drive
P0010	Склад C	329	171 Pine Lane
P0020	Склад B	200	4141 Greenway Drive
P0020	Склад D	278	800 Massey Street

Здесь первичный ключ состоит из столбцов PART и WAREHOUSE. Таблица нарушает правила второй нормальной формы, так как столбец WAREHOUSE_ADDRESS зависит только от значения WAREHOUSE.

Недостатки такой структуры:

- Адрес склада повторяется во всех записях для детали, хранящейся на этом складе.
- Если адрес склада изменяется, нужно обновлять строки для всех деталей, хранящихся на этом складе.
- Из-за этой избыточности данные могут стать несогласованными, то есть в разных записях будут указаны различные адреса для одного склада.
- Если в какой-то момент на одном из складов не будет деталей, можно потерять его адрес, поскольку его не будет ни в одной из строк.

Чтобы исправить эти недостатки, можно разделить таблицу на следующие две таблицы:

Таблица 11. Таблица PART_STOCK, соответствующая второй нормальной форме

PART (Первичный ключ)	WAREHOUSE (Первичный ключ)	QUANTITY
P0010	Склад A	400
P0010	Склад B	543
P0010	Склад C	329
P0020	Склад B	200
P0020	Склад D	278

Таблица 12. Таблица WAREHOUSE, соответствует второй нормальной форме

WAREHOUSE (Первичный ключ)	WAREHOUSE_ADDRESS
Склад A	1608 New Field Road

Таблица 12. Таблица WAREHOUSE, соответствует второй нормальной форме (продолжение)

WAREHOUSE (Первичный ключ)	WAREHOUSE_ADDRESS
Склад B	4141 Greenway Drive
Склад C	171 Pine Lane
Склад D	800 Massey Street

Поддержка двух таблиц во второй нормальной форме влечет некоторое ухудшение производительности. Программам, создающим отчеты о положении деталей, придется объединять обе таблицы для извлечения нужной информации.

Чтобы лучше понять вопросы, связанные с производительностью, посмотрите раздел "Настройка производительности программы" в книге *Руководство администратора: Производительность*.

Третья нормальная форма

Таблица находится в третьей нормальной форме, если каждый неключевой столбец независим от других неключевых столбцов и зависит только от ключа.

Первая таблица следующего примера содержит столбцы EMPNO и WORKDEPT. Допустим, добавляется столбец DEPTNAME (смотрите Табл. 14 на стр. 112). Новый столбец зависит от WORKDEPT, а первичный ключ - это EMPNO. Получившаяся таблица будет нарушать третью нормальную форму. Если мы изменим DEPTNAME для одного из сотрудников, например, для Джона Паркера, это не изменит названий отделов для других сотрудников этого отдела. Но теперь у отдела E11 будет два разных названия. Получающаяся несогласованность показана в измененной версии таблицы.

Таблица 13. Ненормализованная таблица EMPLOYEE_DEPARTMENT до изменения

EMPNO (Первичный ключ)	FIRSTNAME	LASTNAME	WORKDEPT	DEPTNAME
000290	John	Parker	E11	Операционный отдел
000320	Ramlal	Mehta	E21	Служба программной поддержки
000310	Maude	Setright	E11	Операционный отдел

Таблица 14. Ненормализованная таблица EMPLOYEE_DEPARTMENT после изменения. Информация в таблице стала несогласованной.

EMPNO (Первичный ключ)	FIRSTNAME	LASTNAME	WORKDEPT	DEPTNAME
000290	John	Parker	E11	Отдел управления установкой
000320	Ramlal	Mehta	E21	Служба поддержки программ
000310	Maude	Setright	E11	Операционный отдел

Эту таблицу можно нормализовать, создав новую таблицу со столбцами для WORKDEPT и DEPTNAME. Тогда обновление типа изменения названия отдела станет гораздо проще - понадобится изменять только новую таблицу.

Однако станет сложнее написать запрос SQL, возвращающий имя сотрудника и название его отдела, потому что потребуется объединять эти две таблицы. К тому же его выполнение, скорее всего, потребует больше времени, чем выполнение запроса к одной таблице. Потребуется дополнительное место для хранения, так как столбец WORKDEPT должен быть в обеих таблицах.

В результате нормализации будут определены следующие таблицы:

Таблица 15. Таблица EMPLOYEE после нормализации таблицы EMPLOYEE_DEPARTMENT

EMPNO (Первичный ключ)	FIRSTNAME	LASTNAME	WORKDEPT
000290	John	Parker	E11
000320	Ramlal	Mehta	E21
000310	Maude	Setright	E11

Таблица 16. Таблица DEPARTMENT после нормализации таблицы EMPLOYEE_DEPARTMENT

DEPTNO (Первичный ключ)	DEPTNAME
E11	Операционный отдел
E21	Служба поддержки программ

Четвертая нормальная форма

Таблица находится в четвертой нормальной форме, если ни в одной строке не содержится нескольких независимых многозначных фактов об объекте.

Рассмотрим такие объекты: сотрудники, сферы деятельности и языки. Один сотрудник может иметь несколько сфер деятельности и знать несколько языков. В этом случае есть две взаимосвязи: между сотрудниками и сферами деятельности, и между сотрудниками и языками. Таблица не находится в четвертой нормальной форме, если она отражает сразу обе связи, как в следующем примере:

Таблица 17. Таблица, нарушающая четвертую нормальную форму

EMPNO (Первичный ключ)	**SKILL (Первичный ключ)**	**LANGUAGE (Первичный ключ)**
000130	Моделирование данных	Английский
000130	Проектирование базы данных	Английский
000130	Проектирование программ	Английский
000130	Моделирование данных	Испанский
000130	Проектирование базы данных	Испанский
000130	Проектирование программ	Испанский

Вместо этого такие отношения надо представить в двух таблицах:

Таблица 18. Таблица EMPLOYEE_SKILL, соответствующая четвертой нормальной форме

EMPNO (Первичный ключ)	**SKILL (Первичный ключ)**
000130	Моделирование данных
000130	Проектирование базы данных
000130	Проектирование программ

Таблица 19. Таблица EMPLOYEE_LANGUAGE, соответствующая четвертой нормальной форме

EMPNO (Первичный ключ)	**LANGUAGE (Первичный ключ)**
000130	Английский
000130	Испанский

Однако, если атрибуты взаимосвязаны (то есть сотрудник знает некоторый язык только для некоторой сферы деятельности), таблицу *не* надо разделять.

Хорошая стратегия при проектировании базы данных - собрать все данные в таблицы в четвертой нормальной форме, а потом посмотреть, дает ли результат приемлемый уровень производительности. Если производительность недостаточна, вы можете реорганизовать данные в таблицы третьей нормальной формы и опять оценить производительность.

Планирование введения ограничений

Ограничение - это правило, которому подчиняется менеджер баз данных. В этом разделе рассмотрено четыре типа работы с ограничениями:

Ограничения уникальности Проверка уникальности значений ключа в таблице. Любые изменения, вносимые в столбец первичного ключа, проверяются на соблюдение уникальности.

Реляционная целостность Накладывает реляционные ограничения на операции вставки, изменения и удаления. В этом состоянии базы данных допустимы любые значения любых внешних ключей.

Проверочные ограничения таблицы Проверяет, не нарушают ли измененные данные условий, указанных при создании или переопределении таблицы.

Триггеры Определяет набор действий, которые надо выполнить при вызове операций изменения, удаления или вставки для данной таблицы.

Ограничения уникальности

Ограничение уникальности - это правило, которое гарантирует уникальность значений ключа внутри таблицы. Каждый ключевой столбец в ограничении уникальности должен быть определен как NOT NULL. Ограничения уникальности определяются в операторах CREATE TABLE или ALTER TABLE с помощью условий PRIMARY KEY или UNIQUE.

У таблицы может быть сколько угодно ограничений уникальности, но только одно ограничение уникальности можно определить как первичный ключ для таблицы. Более того, таблица не может иметь несколько ограничений уникальности на одном и том же наборе столбцов.

Если определено ограничение уникальности, менеджер баз данных создает (если надо) индекс уникальности и назначает его первичным индексом или системным индексом уникальности. Ограничение реализуется через индекс уникальности. Когда ограничение уникальности установлено, проверка уникальности при изменении многих строк откладывается до конца изменения.

Ограничение уникальности также можно использовать как родительский ключ в реляционном ограничении.

Реляционная целостность

Менеджер баз данных поддерживает реляционную целостность с помощью *реляционных ограничений*, которые требуют, чтобы каждое значение данного атрибута или столбца таблицы существовало также в какой-то другой таблице или другом столбце. Например, реляционное ограничение может требовать, чтобы каждый сотрудник из таблицы EMPLOYEE работал в одном из отделов, который есть в таблице DEPARTMENT. Сотрудник не может работать в несуществующем отделе.

Можно включить реляционные ограничения в базу данных, чтобы гарантировать поддержание реляционной целостности и дать оптимизатору возможность использовать информацию об этих специальных отношениях для более эффективной обработки запросов. При планировании реляционной целостности учтите все отношения между таблицами базы данных. Обозначить отношение можно, определив первичный ключ и реляционные ограничения.

Рассмотрим следующие связанные таблицы:

Таблица 20. Таблица DEPARTMENT

DEPTNO (Первичный ключ)	DEPTNAME	MGRNO
A00	Spiffy Computer Service Div.	000010
B01	Плановый отдел	000020
C01	Информационный центр	000030
D11	Производственный отдел	000060

Таблица 21. Таблица EMPLOYEE

EMPNO (Первичный ключ)	FIRSTNAME	LASTNAME	WORKDEPT (Внешний ключ)	PHONENO
000010	Christine	Haas	A00	3978
000030	Sally	Kwan	C01	4738
000060	Irving	Stern	D11	6423
000120	Sean	O'Connell	A00	2167
000140	Heather	Nicholls	C01	1793
000170	Masatoshi	Yoshimura	D11	2890

Многие понятия, полезные для понимания реляционной целостности, будут обсуждаться на примере этих таблиц.

Ключ уникальности - это столбец или набор столбцов, в котором значения не повторяются в различных строках. Вы можете определить один из ключей уникальности как первичный ключ таблицы. Ключ уникальности называется *родительским ключом*, если на него ссылается внешний ключ.

Первичный ключ - это ключ уникальности, указанный в определении таблицы. У каждой таблицы может быть только один первичный ключ. У приведенных выше таблиц DEPARTMENT и EMPLOYEE первичными ключами являются столбцы DEPTNO и EMPNO соответственно.

Внешний ключ - это столбец или набор столбцов таблицы, ссылающийся на ключ уникальности или первичный ключ той же самой или другой таблицы. Внешний ключ используется для образования отношений с ключом уникальности или первичным ключом, чтобы поддерживать реляционную целостность между таблицами. Столбец WORKDEPT в таблице EMPLOYEE - это внешний ключ, так как он ссылается на первичный ключ - столбец DEPTNO - в таблице DEPARTMENT.

Составной ключ - это ключ, состоящий из нескольких столбцов. И первичный, и внешний ключи могут быть составными. Например, если отделы однозначно идентифицируются комбинацией из номера подразделения и номера отдела, понадобится два столбца, чтобы создать ключ для таблицы DEPARTMENT.

Родительский ключ - это первичный ключ или ключ уникальности реляционного ограничения. *Ограничение первичного ключа* - это родительский ключ реляционного ограничения по умолчанию, если набор столбцов родительского ключа не указан.

Родительская таблица - это таблица, содержащая родительский ключ, связанный хотя бы с одним внешним ключом той же самой или другой таблицы. Таблица может быть родительской в произвольном числе связей. Например, таблица DEPARTMENT с первичным ключом DEPTNO является родительской для таблицы EMPLOYEE с внешним ключом WORKDEPT.

Родительская строка - это строка родительской таблицы, значение родительского ключа которой совпадает хотя бы с одним значением внешнего ключа зависимой таблицы. Строка родительской таблицы необязательно является родительской строкой. Четвертая строка (D11) таблицы DEPARTMENT является родительской строкой для третьей и шестой строк таблицы EMPLOYEE. Вторая строка (B01) таблицы DEPARTMENT не является родительской ни для какой строки.

Зависимая таблица - это таблица, содержащая один или более внешних ключей. Зависимая таблица сама может быть родительской таблицей. Таблица может

быть зависимой в произвольном числе связей. Таблица EMPLOYEE содержит внешний ключ WORKDEPT и является зависимой от таблицы DEPARTMENT с первичным ключом DEPTNO.

Зависимая строка - это строка зависимой таблицы, имеющая непустое значение внешнего ключа, совпадающее со значением родительского ключа. Значение внешнего ключа представляет собой ссылку из зависимой строки на родительскую строку. Так как внешний ключ может принимать значения NULL, строка зависимой таблицы необязательно является зависимой строкой.

Таблицу называют *потомком* другой таблицы, если это зависимая таблица или если она потомок зависимой таблицы. Таблица-потомок содержит внешний ключ, по которому можно найти родительский ключ некоторой таблицы.

Реляционный цикл - это путь, связывающий таблицу саму с собой. Таблица, непосредственно связанная сама с собой, называется *автореферентной*. Если бы в таблице EMPLOYEE был столбец MGRID, содержащий EMPNO начальника для каждого сотрудника, она была бы автореферентной. Столбец MGRID был бы внешним ключом для таблицы EMPLOYEE.

Автореферентная таблица выступает одновременно как родительская и зависимая в одной взаимосвязи. Автореферентная строка - это строка, которая сама для себя и родительская, и зависимая. Ограничение, имеющее место в этой ситуации, называется автореферентным. Например, если значение внешнего ключа в строке автореферентной таблицы совпадает со значением ключа уникальности в этой строке, строка является автореферентной.

Реляционное ограничение - это требование, чтобы непустые значения указанного внешнего ключа допускались, только если они также встречаются среди значений ключа уникальности указанной таблицы. Назначение реляционных ограничений - это гарантия поддержания связей в базе данных и соблюдения правил ввода данных.

Последствия для операций SQL

Введение реляционных ограничений приводит к особым последствиям для некоторых операций SQL, зависящим от того, является таблица родительской или зависимой. В этом разделе описано, как поддержание реляционной целостности влияет на операции SQL INSERT, DELETE, UPDATE и DROP.

DB2 *не* накладывает реляционные ограничения на системы автоматически. Следовательно, если вы хотите наложить реляционные ограничения на систему, в ваших программах должна быть реализована необходимая логика.

Далее описаны следующие темы:

- “Правила INSERT” на стр. 118
- “Правила DELETE” на стр. 118

- “Правила UPDATE” на стр. 119.

Правила INSERT: Вставить строку в родительскую таблицу можно в любой момент, не предпринимая никаких действий в отношении зависимой таблицы. Однако в зависимую таблицу нельзя вставить строку, если в родительской таблице нет строки со значением родительского ключа, равным значению внешнего ключа вставляемой строки, кроме случая, когда значение внешнего ключа пусто (NULL). Значение составного ключа считается пустым, если все его компоненты пусты (равны NULL).

Это правило накладывается при задании внешнего ключа.

При попытке вставить строку в таблицу с реляционными ограничениями операция INSERT не будет выполнена, если какое-либо непустое значение внешнего ключа не встречается в родительском ключе. Если при попытке вставить несколько строк операция INSERT не удалась для одной строки, не будет вставлена ни одна строка.

Правила DELETE: Когда из родительской таблицы удаляется строка, DB2 проверяет, нет ли в зависимых таблицах зависимых строк с соответствующими значениями внешнего ключа. Если нашлись зависимые строки, возможно несколько действий. Какое действие будет выполнено, определяет заданное вами при создании зависимой таблицы правило *удаления*.

Правила удаления для зависимой таблицы (таблицы, содержащей внешний ключ) при удалении первичного ключа:

RESTRICT — Не допускает удаления строк из родительской таблицы, если у них есть зависимые строки. Если надо удалить как родительские, так и зависимые строки, сначала удалите зависимые строки. Удаление родительских строк первыми нарушает реляционное ограничение и поэтому не допускается.

NO ACTION — Наличие родительской строки для каждого потомка проверяется после применения всех реляционных ограничений.

CASCADE — Означает, что удаление строки из родительской таблицы влечет автоматическое удаление всех связанных с ней строк зависимой таблицы. Это правило полезно, если строка зависимой таблицы не представляет интереса при отсутствии строки родительской таблицы.

Удаление родительской строки влечет автоматическое удаление зависимых строк,

	ссылающихся на первичный ключ. Нет необходимости удалять сначала зависимые строки. Если у некоторых из зависимых строк есть свои зависимые строки, применяется правило удаления для этих отношений. DB2 управляет каскадным удалением.
SET NULL	Внешнему ключу зависимых строк при удаление строки из родительской таблицы присваивается значение NULL. Другие части строки не изменяются.

Если при создании таблицы правила удаления не определены явно, применяется правило NO ACTION.

Любая таблица, которая может быть вовлечена в операцию удаления, называется связанной по удалению. Для отношения связи по удалению есть следующие ограничения.

- Таблица не может быть связана сама с собой по удалению через реляционный цикл из нескольких таблиц.
- Если таблица связана по удалению с другой таблицей через несколько отношений зависимости, у этих отношений должны быть одинаковые правила удаления - либо CASCADE, либо NO ACTION.
- Если автореферентная таблица зависит от другой таблицы в отношении CASCADE, правило удаления автореферентного отношения должно быть тоже CASCADE.

Удалять строки из зависимой таблицы можно в любой момент, не предпринимая никаких действий для родительской таблицы. В отношении отдел - сотрудник, например, увольнение сотрудника и удаление его строки из таблицы сотрудников никак не повлияет на таблицу отделов. (В обратном отношении - сотрудник - отдел - ID начальника отдела является внешним ключом, ссылающимся на родительский ключ таблицы сотрудников. В таком случае увольнение начальника *влияет* на таблицу отделов.)

Правила UPDATE: DB2 не допускает изменения ключа уникальности родительской строки. Когда внешний ключ в зависимой таблице изменяется и новое значение непусто, оно должно соответствовать какому-нибудь значению родительского ключа родительской таблицы данной связи. Если операция UPDATE нарушает какие-либо реляционные ограничения, возникает ошибка и никакие строки не изменяются.

Когда изменяется значение в столбце родительского ключа:

- При правиле изменения RESTRICT изменение не производится, если какая-либо строка зависимой таблицы соответствует исходному значению ключа.

- Если после выполнения оператора UPDATE (но до применения триггеров after) хотя бы у одной строки зависимой таблицы не будет соответствующего родительского ключа, при правиле изменения NO ACTION изменение не выполняется.

Чтобы изменить значение родительского ключа в родительской строке, сначала нужно удалить отношения со всеми дочерними строками в зависимых таблицах одним из следующих способов:

- Стерев дочерние строки; или,
- Изменив внешние ключи в зависимых таблицах, чтобы они содержали другие допустимые значения.

Если зависимостей от значения ключа для строки нет, строка больше не является родительской в реляционном отношении и ее можно изменить.

Если часть внешнего ключа изменяется и ни одна часть этого внешнего ключа не пуста, новое значение внешнего ключа должно встречаться и как значение ключа уникальности родительской таблицы. Если от данного ключа уникальности не зависит никакой внешний ключ, то есть строка, содержащая этот ключ уникальности, *не* является родительской, часть ключа уникальности можно изменить. Однако в этом случае для изменения можно выбрать только одну строку, так как вы работаете с ключом уникальности и повторение строк недопустимо.

Проверочные ограничения таблицы

Логические правила, отражаемые в вашей структуре, можно реализовывать с помощью проверочных ограничений таблицы. *Проверочные ограничения таблицы* указывают условия, которые применяются к каждой строке таблицы. Эти ограничения автоматически активируются, когда к таблице применяется оператор изменения или вставки. Проверочные ограничения определяются в операторах CREATE TABLE или ALTER TABLE.

Проверочное ограничение таблицы можно использовать для проверки достоверности. Например, значения номера отдела должны лежать в диапазоне от 10 до 100; название должности сотрудника может быть только ″Продавец″, ″Начальник″ или ″Клерк″; сотрудник, работающий в компании более 8 лет, должен зарабатывать не меньше 40500 долларов.

Информацию о влиянии проверочных ограничений на команды IMPORT и LOAD можно найти в руководстве *Data Movement Utilities Guide and Reference*.

Триггеры

Триггер - это определенный набор действий, которые выполняются при каждом выполнении операций удаления, вставки или изменения над данной таблицей. Можно определить триггеры для соблюдения логических правил. Также триггеры можно использовать для автоматического обновления данных сводки

или аудита. Так как триггеры хранятся в базе данных, вам не придется программировать эти действия в каждой прикладной программе. Триггер программируется один раз и хранится в базе данных; он автоматически вызывается DB2, если это требуется, когда программа использует базу данных. Это гарантирует, что относящиеся к данным логические правила всегда будут выполнены. Если логическое правило поменялось, придется изменять только триггер.

Оператор SQL с триггером может вызывать пользовательские функции (UDF). Это позволяет вызывать при активации триггера операции, не описываемые SQL. Например, можно в качестве оповещения отправить электронную почту. Подробную информацию о триггерах смотрите в разделе ″Создание триггера″ книги *Руководство администратора: Реализация* и в руководстве *Application Development Guide*.

Другие особенности проектирования базы данных

При проектировании базы данных важно понять, к каким таблицам у пользователей должен быть доступ. Доступ к таблицам предоставляется или отзывается с помощью полномочий. Наивысший уровень прав доступа дают полномочия управления системой (SYSADM). Пользователь с полномочиями SYSADM может предоставлять другие полномочия, в том числе полномочия администратора базы данных (DBADM).

Есть другие вопросы, которые, возможно, вы захотите отразить в вашей структуре, такие как *действия аудита*, *хронологические данные*, *сводные таблицы*, *защита*, *контроль типов данных* и *возможность параллельной обработки*.

В целях *аудита* может понадобиться записывать все изменения, вносимые в базу данных за определенный период. Например, можно обновлять таблицу аудита каждый раз, когда меняется заработная плата сотрудника. Изменения в эту таблицу можно вносить автоматически, если определен соответствующий триггер. Аудит можно также осуществлять с помощью возможности аудита DB2. Дополнительную информацию смотрите в разделе ″Аудит действий DB2″ в книге *Руководство администратора: Реализация*.

Для повышения производительности может понадобиться обращаться только к некоторому объему данных, поддерживая при этом базовые *хронологические* данные. В структуру надо включить требования по поддержанию хронологических данных, такие как число месяцев или лет, в течение которых данные должны быть еще доступны, прежде чем их можно будет удалить.

Можно также использовать информацию *сводок*. Например, у вас может быть таблица, содержащая в себе всю информацию о ваших сотрудниках. Однако вам бы хотелось, чтобы эта информация была разбита по отдельным таблицам для отделов или подразделений. В этом случае будут полезны сводные таблицы для

каждого отдела или подразделения, основанные на данных исходной таблицы. Подробную информацию о сводных таблицах смотрите в разделе ″Создание сводных таблиц″ книги *Руководство администратора: Реализация*.

Средства *защиты* тоже надо задавать в вашей структуре. Например, можно обеспечить доступ пользователя к определенным типам данных через таблицы защиты. Можно определить уровни доступа к различным типам данных и задать, кто имеет доступ к этим данным. Конфиденциальные данные, такие как данные о сотруднике и его зарплате, будут иметь строгие ограничения защиты. Дополнительную информацию о защите и авторизациях смотрите в разделе ″Управление доступом к базе данных″ книги *Руководство администратора: Реализация*.

Можно создавать таблицы, с которыми связан некоторый *структурированный тип*. Имея такие типизированные таблицы, можно создавать иерархическую структуру с определенными отношениями между этими таблицами, называемую *иерархией типов*. Иерархия типов состоит из одного корневого типа, надтипов и подтипов.

Представление *ссылочного типа* определено, если создан корневой тип иерархии типов. Назначение ссылки - это строка в типизированной таблице или производной таблице.

Дополнительную информацию о реализации структуры с типизированными строками и таблицами смотрите в разделе ″Реализация вашего проекта″ в руководстве *Руководство администратора: Реализация*. Информацию о перемещении данных между типизированными таблицами, находящимися в иерархической структуре, смотрите в руководстве *Data Movement Utilities Guide and Reference*.

По мере роста вашего бизнеса вам могут понадобиться дополнительные емкость и производительность, обеспечиваемые DB2 Enterprise - Extended Edition. В этой среде ваша база данных распределена по нескольким компьютерам или системам, каждая из которых ответственна за хранение и выдачу некоторой порции всей базы данных. Все разделы (или узлы) при обработке операций утилит или SQL работают параллельно.

Вопросы, связанные с параллельными операциями, рассматриваются на протяжении всей этой книги.

Глава 8. Физическая структура базы данных

После создания логической структуры базы данных (“Глава 7. Логическое проектирование базы данных” на стр. 97) встают вопросы о физической среде, где будут находиться база данных и таблицы. Нужно понять, какие файлы будут созданы для поддержки базы данных и управления ей, определить, сколько места потребуется для хранения данных, а также решить, как использовать табличные пространства, необходимые для хранения данных.

Рассматриваются следующие темы:

Каталоги баз данных

При создании базы данных DB2 создает отдельный каталог для хранения управляющих файлов (например, файлов заголовков журналов) и для контейнеров, выделяемых табличным пространствам по умолчанию. Связанные с базой данных объекты не всегда хранятся в каталоге базы данных; они могут храниться в различных местах, в частности, на различных устройствах.

База данных создается в экземпляре, определяемом переменной среды DB2INSTANCE, или в экземпляре, к которому вы явно подключились при помощи команды ATTACH. Введение в использование экземпляров смотрите в разделе ″Использование нескольких экземпляров менеджера баз данных″ книги *Руководство администратора: Реализация*.

В системах на основе UNIX используется следующая схема именования:

```
заданный_путь/$DB2INSTANCE/NODEnnnn/SQL00001
```

В операционных системах OS/2 и Windows используется следующая схема именования:

```
D:\$DB2INSTANCE\NODEnnnn\SQL00001
```

где

- `заданный_путь` - необязательное заданное пользователем место установки экземпляра.

- NODEnnnn - идентификатор узла в среде распределенных баз данных. Первый узел - NODE0000.
- "D:" - буква диска, задающая том, на котором расположен корневой каталог.

SQL00001 содержит объекты, связанные с первой созданной базой данных; последующим базам данных даются последовательные номера: SQL00002 и так далее.

Эти подкаталоги создаются в каталоге с именем, совпадающим с именем экземпляра менеджера баз данных, к которому вы были подключены при создании базы данных. (В OS/2 и операционных системах Windows эти подкаталоги создаются в корневом каталоге тома, обозначенного буквой диска.) Эти подкаталоги экземпляров и баз данных создаются в пути, указанном в команде CREATE DATABASE, и менеджер баз данных управляет ими автоматически. В зависимости от платформы каждый экземпляр может принадлежать владельцу экземпляра, имеющему полномочия системного администратора (SYSADM) для базы данных этого экземпляра.

Чтобы избежать возможных проблем, не создавайте каталогов с той же схемой именования и не производите никаких действий с каталогами, созданными менеджером баз данных.

Файлы баз данных

С базой данных связаны следующие файлы:

Имя файла	Описание
SQLDBCON	В этом файле хранятся параметры и флаги настройки базы данных. Информацию об изменении параметров конфигурации базы данных смотрите в книге *Руководство администратора: Производительность*.
SQLOGCTL.LFH	Этот файл помогает отслеживать все файлы журнала базы данных и управлять ими.
Syyyyyyy.LOG	Файлы журнала базы данных, пронумерованные от 0000000 до 9999999. Количество этих файлов определяется параметрами конфигурации базы данных *logprimary* и *logsecond*. Размер отдельных файлов определяется параметром конфигурации базы данных *logfilsiz*. При циклической регистрации файлы с теми же номерами используются повторно. При архивной регистрации номера файлов увеличиваются по мере того, как журналы архивируются и выделяются новые журналы. При достижении 9999999 нумерация снова начинается с нуля.

По умолчанию эти файлы журнала хранятся в каталоге SQLOGDIR. SQLOGDIR находится в подкаталоге SQL*nnnnn*.

SQLINSLK Этот файл помогает обеспечить использование базы данных только одним экземпляром менеджера баз данных.

SQLTMPLK Этот файл помогает обеспечить использование базы данных только одним экземпляром менеджера баз данных.

SQLSPCS.1 В этом файле содержится определение и текущее состояние всех табличных пространств в базе данных.

SQLSPCS.2 Этот файл - резервная копия `SQLSPCS.1`. При отсутствии обоих этих файлов доступ к базе данных невозможен.

SQLBP.1 В этом файле содержится определение всех пулов буферов, используемых в базе данных.

SQLBP.2 Этот файл - резервная копия `SQLBP.1`. При отсутствии обоих этих файлов доступ к базе данных невозможен.

DB2RHIST.ASC

Это файл хронологии базы данных. В нем хранится хронология управляющих операций, производившихся над базой данных, например, резервного копирования и восстановления из резервной копии.

DB2RHIST.BAK

Этот файл - резервная копия `DB2RHIST.ASC`.

Примечания:

1. *Не* изменяйте эти файлы непосредственно. Обращаться к ним можно только через документированные API и инструменты, использующие эти API, включая процессор командной строки и Центр управления DB2.
2. Не перемещайте эти файлы.
3. Не удаляйте эти файлы.
4. Единственный поддерживаемый способ резервного копирования базы данных или табличного пространства - через API **sqlubkp** (Резервное копирование базы данных), а также через использующие этот API процессор командной строки и Центр управления DB2.

Оценка пространства, требуемого для таблиц

Оценка размера объектов баз данных не может быть точной. Вычисление размера усложняют излишки, связанные с фрагментацией диска, свободное место и использование столбцов переменной длины, так как для типов столбцов и длин строк имеется множество возможностей. После предварительной оценки размера базы данных создайте проверочную базу данных и заполните ее данными, приближенными к реальным.

В Центре управления есть несколько утилит, которые помогают оценить требования размера различных объектов баз данных:

- Вы можете выбрать объект и воспользоваться утилитой оценки размера. Эта утилита сообщает текущий размер существующего объекта, например, таблицы. Затем вы можете изменить объект, и утилита вычислит для него новые приблизительные значения. Эта утилита поможет приближенно вычислить требования памяти, учитывая предстоящий рост данных. Она дает не просто оценку размера объекта. Она предоставляет также возможный диапазон размеров объекта: наименьший размер, основанный на текущих значениях, и наибольший возможный размер.
- Связи между объектами можно посмотреть в диалоговом окне ″Показать связанные″.
- Вы можете выделить любой объект базы данных в текущем экземпляре и выбрать ″Генерировать DDL″. Эта функция при помощи утилиты **db2look** генерирует операторы определения данных для базы данных. Информацию об этой утилите смотрите в руководстве *Command Reference*.

В обоих случаях доступна кнопка ″Показать SQL″ или ″Показать команды″. Вы можете сохранить полученные операторы SQL или команды в файлы сценариев для последующего использования. Во всех этих утилитах доступна электронная справка.

Учитывайте эти утилиты при планировании физических требований баз данных.

При оценке размера базы данных следует также учитывать:

- “Таблицы системного каталога” на стр. 127
- “Данные пользовательских таблиц” на стр. 127
- “Данные длинных полей” на стр. 129
- “Данные больших объектов” на стр. 129
- “Индексное пространство” на стр. 130

Здесь не рассматриваются требования к дисковому пространству, которые предъявляют:

- Локальный файл каталога баз данных
- Системный файл каталога баз данных
- Операционная система для управления файлами, включая:
  - размер файлового блока
  - управляющее пространство каталога

Таблицы системного каталога

При создании базы данных создаются таблицы системного каталога. Системные таблицы растут по мере того, как в базу данных добавляются объекты и привилегии базы данных. Первоначально они занимают на диске приблизительно 3,5 Мбайта.

Размер пространства, выделяемого для таблиц каталога, зависит от типа табличного пространства и размера экстента табличного пространства, содержащего таблицы каталога. Например, если используется табличное пространство DMS с размером экстента 32, для табличного пространства каталога будет выделено 20 Мбайт. Дополнительную информацию смотрите в разделе “Проектирование и выбор табличных пространств” на стр. 143.

Примечание: Для баз данных с несколькими разделами таблицы каталога находятся только на разделе, из которого была вызвана команда CREATE DATABASE. Место для таблиц каталога требуется только для этого раздела.

Данные пользовательских таблиц

По умолчанию табличные данные хранятся на страницах по 4 Кбайта. Каждая страница (независимо от ее размера) содержит 76 дополнительных байт для менеджера баз данных. Для хранения пользовательских данных (или строк) останется 4020 байт, хотя никакая строка на странице размером 4 Кбайта не может превышать 4005 байт в длину. Строка *не может* занимать несколько страниц. При использовании страниц размером 4 Кбайта допустимо не более 500 столбцов.

Страницы табличных данных *не* содержат данных для столбцов типов LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB и DBCLOB. В строках в странице табличных данных содержится дескриптор для этих столбцов. (Информацию об оценке требований размера для табличных объектов, содержащих данные этих типов, смотрите в разделах “Данные длинных полей” на стр. 129 и “Данные больших объектов” на стр. 129.)

Обычно строки вставляются в таблицу в порядке ″первый подходящий″. В файле (при помощи карты свободного места) ищется первый отрезок свободного места, достаточно большой, чтобы вместить новую строку. При изменении строки она изменяется на старом месте, если только на странице достаточно для этого пространства. Если это не так, то на старом месте строки создается запись, указывающая на новое местоположение измененной строки в табличном файле.

При вызове оператора ALTER TABLE APPEND ON данные всегда добавляются в конец, и информация о свободном месте на страницах данных не сохраняется. Дополнительную информацию об этом операторе смотрите в справочнике *SQL Reference*.

Количество страниц по 4 Кбайта для каждой пользовательской таблицы в базе данных можно вычислить так:

```
   число_записей_на_страницу = ОКРУГЛИТЬ В МЕНЬШУЮ СТОРОНУ(4020/(средний
размер строки + 10))
```

и

```
   количество_страниц = (количество_записей/записей_на_страницу) * 1,1
```

где средний размер строки - сумма средних размеров столбцов (информацию о размере отдельных столбцов смотрите в руководстве *SQL Reference*, оператор CREATE TABLE), а множитель ″1,1″ нужен для учета дополнительного места.

Примечание: Эта формула дает только приблизительную оценку. Если длина записи переменна, точность оценки уменьшается из-за фрагментации и записей переполнения.

Вы можете также создавать пулы буферов или табличные пространства с размером страницы 8 кб, 16 кб или 32 кб. Размер страницы у всех таблиц, созданных в табличном пространстве, совпадает с размером страницы табличного пространства. Отдельный объект таблицы или индекса может достигать размера 512 Гб (при размере страницы 32 кб). При использовании страниц размером 8 кб, 16 кб или 32 кб допустимо не более 1012 столбцов. При использовании страниц размером 4 кб максимальное количество столбцов - 500. Максимальные длины строк также зависят от размера страницы:

- При размере страницы 4 кб длина строки может достигать 4005 байт.
- При размере страницы 8 кб длина строки может достигать 8101 байта.
- При размере страницы 16 кб длина строки может достигать 16293 байт.
- При размере страницы 32 кб длина строки может достигать 32677 байт.

Увеличение размера страницы ведет к уменьшению количества уровней во всех индексах. При работе с программами OLTP (online transaction processing - оперативная обработка транзакций), выполняющими нерегулярное чтение и запись строк, лучше использовать меньший размер страницы, так как это позволяет не тратить пространство буферов на ненужные строки. При работе с программами DSS (decision support system - система поддержки решений), обращающимися сразу к большому числу последовательных строк, лучше использовать больший размер страницы, так как это уменьшает количество требований ввода/вывода, необходимых для считывания заданного числа строк. Исключение - ситуация, когда размер строки меньше размера страницы, деленного на 255. В этом случае на каждой странице появляется неиспользуемое место. (Это происходит потому, что максимальное число строк на странице - 255.) Чтобы уменьшить потери места, возможно, лучше использовать меньший размер страницы.

Резервную копию нельзя восстановить с другим размером страницы.

Нельзя импортировать файлы данных IXF, содержащие больше 755 столбцов. Дополнительную информацию об импорте данных в таблицы и о файлах данных IXF смотрите в руководстве *Data Movement Utilities Guide and Reference*.

Объявленные временные таблицы можно создавать только в табличных пространствах специального типа ″пользовательское временное″. Пользовательское временное табличное пространство не создается по умолчанию. Временные таблицы не могут содержать данные типа LONG. Таблицы автоматически отбрасываются, когда программа отключается от базы данных, и при оценке требуемого ими места нужно это учитывать.

Данные длинных полей

Данные длинных полей хранятся в отдельном табличном объекте, структура которого отличается от других типов данных (смотрите разделы “Данные пользовательских таблиц” на стр. 127 и “Данные больших объектов”).

Данные хранятся в областях по 32 Кбайта, разбитых на сегменты, размеры которых равны степеням двойки, умноженным на 512 байт. (Таким образом, эти сегменты могут содержать 512 байт, 1024 байта, 2048 байт и так далее, вплоть до 32768 байт.)

Типы данных длинных полей (LONG VARCHAR или LONG VARGRAPHIC) хранятся так, чтобы упростить повторное использование свободного места. Информация о выделенном и свободном месте хранится на страницах размещения по 4 Кбайта, периодически вставляемых между страницами с информацией объекта.

Количество неиспользуемого места в объекте зависит от размера данных длинных полей и от того, является ли этот размер относительно постоянным для всех появлений длинных данных. Для вхождений данных длиннее 255 байт это неиспользуемое место может достигать 50 процентов размера данных длинных полей.

Если символьные данные меньше размера страницы и помещаются в запись вместе с остальными данными, вместо типов данных LONG VARCHAR или LONG VARGRAPHIC следует использовать CHAR, GRAPHIC, VARCHAR или VARGRAPHIC.

Данные больших объектов

Данные больших объектов хранятся в двух отдельных табличных объектах, структура которых отличается от остальных типов данных (смотрите разделы “Данные пользовательских таблиц” на стр. 127 и “Данные длинных полей”).

Для оценки места, требуемого данными большого объекта, нужно учитывать два табличных объекта, используемых для хранения данных этого типа:

- **Объекты данных большого объекта**

Данные хранятся в областях по 64 Мбайта, разбитых на сегменты, размеры которых равны степеням двойки, умноженным на 1024 байта. (Таким образом, эти сегменты могут содержать 1024 байта, 2048 байт, 4096 байт и так далее, вплоть до 64 Мбайт.)

Чтобы уменьшить место, используемое на диске данными большого объекта, надо указать опцию COMPACT в условии *lob-options* операторов CREATE TABLE и ALTER TABLE. Опция COMPACT минимизирует необходимое дисковое пространство, позволяя разбивать данные больших объектов на меньшие сегменты. При этом сами данные не сжимаются, просто для них используется минимальный объем пространства, до ближайшей границы 1 Кбайт. Опция COMPACT может уменьшить производительность при добавлении данных в конец значений больших объектов.

Объем свободного места, содержащийся в объектах данных большого объекта, зависит от частоты изменений и удалений, а также от размера вставляемых значений большого объекта.

- **Объекты размещения больших объектов**

  Информация о выделенном и свободном месте хранится на страницах размещения по 4 Кбайта отдельно от самих данных. Количество этих страниц зависит от объема данных, выделенных для данных больших объектов, включая неиспользуемое место. Дополнительный расход пространства вычисляется так: одна 4-Кбайтная страница на каждые 64 Гбайта и одна 4-Кбайтная страница на каждые 8 Мбайт.

Если символьные данные меньше размера страницы и помещаются в запись вместе с остальными данными, вместо типов данных BLOB, CLOB и DBCLOB следует использовать CHAR, GRAPHIC, VARCHAR или VARGRAPHIC.

Индексное пространство

Место, необходимое для каждого индекса, можно приблизительно вычислить как:

```
(средний размер ключа индекса + 8) * количество строк * 2
```

где:

- ″Средний размер ключа индекса″ - это число байт в каждом столбце из ключа индекса. Информацию о том, как вычислить число байт для столбцов с различными типами данных, смотрите в описании оператора CREATE TABLE в справочнике *SQL Reference*. (При оценке среднего размера столбца для столбцов VARCHAR и VARGRAPHIC используйте текущий средний размер данных плюс один байт. Не используйте максимальный объявленный размер.)
- Множитель ″2″ отражает дополнительный расход места, например, на промежуточные страницы и на свободное место.

Примечание: Для всех столбцов, в которых допустимы пустые значения, добавьте лишний байт для индикатора пустых значений.

При создании индекса необходимо временное пространство. Максимальный объем временного пространства при создании индекса можно приблизительно вычислить как:

```
(средний размер ключа индекса + 8) * количество строк * 3,2
```

где множитель ″3,2″ отражает дополнительный расход места на индекс и пространство для сортировки при создании индекса.

Примечание: Если индекс не является индексом уникальности, для хранения повторных значений ключа используется только четыре байта. Указанная выше оценка не учитывает дублей. Приведенная формула может дать слишком большое значение объема пространства, необходимого для хранения индекса.

Для оценки количества конечных страниц можно использовать следующие две формулы (вторая дает более точную оценку). Точность этих оценок в большой степени зависит от того, насколько точно средние значения отражают действительные данные.

Примечание: Для SMS минимальное необходимое место - 12 Кбайт. Для DMS минимальное необходимое место - один экстент.

- Грубая оценка среднего числа ключей на конечную страницу:

  $$\frac{(0{,}9 * (U - (M*2))) * (D + 1)}{K + 6 + (4 * D)}$$

  где:
  - U, доступное место на странице, приблизительно равно размеру страницы минус 100. Для страницы размером 4096 U равно 3996.
  - M = U / (8 + *минимальныйРазмерКлюча*)
  - D = среднее число дубликатов на значение ключа
  - K = *среднийРазмерКлюча*

  Помните, что к значениям *минимальныйРазмерКлюча* и *среднийРазмерКлюча* надо добавить по одному байту на каждую часть ключа, допускающую пустые значения, и по одному байту для длины на каждую часть ключа с переменной длиной.

  При наличии включаемых столбцов их тоже надо учитывать в значениях *минимальныйРазмерКлюча* и *среднийРазмерКлюча*.

  Если при создании индекса указывалось количество свободного места, отличающееся от значения по умолчанию (10%), `0,9` надо заменить на значение (100 - процент_свободного_места)/100.

- Более точная оценка среднего числа ключей на конечную страницу:

  ```
  L = число конечных страниц = X / (среднее число ключей на конечную страницу)
  ```

  где X - общее число строк в таблице.

  Исходный размер индекса оценивается примерно так:

  ```
  (L + 2L/(среднее число ключей на конечную страницу)) * размер_страницы
  ```

  Для табличных пространств DMS сложите размеры всех индексов для данной таблицы и округлите результат в большую сторону до ближайшего числа, кратного размеру экстента табличного пространства, в котором находится индекс.

  Необходимо выделить дополнительное место для роста индекса в результате операций INSERT/UPDATE, которые могут повлечь за собой разбиение страниц.

  Приведенная ниже формула позволяет получить более точную оценку первоначального размера индекса, а также оценку количества уровней в этом индексе. (Это особенно важно, если в определении индекса есть включаемые столбцы.) Среднее число ключей на промежуточную страницу:

  ```
  (0,9 * (U - (M*2))) * (D + 1)
  ----------------------------
        K + 12 + (8 * D)
  ```

  где:

  – U - доступное место на странице, приблизительно равно размеру страницы минус 100. Для страницы размером 4096 U равно 3996.
  – D - среднее число дублей на значение ключа на промежуточных страницах (их будет гораздо меньше, чем на конечных страницах, и для упрощения вычислений можно подставить сюда 0).
  – M = U / (8 + *минимальныйРазмерКлюча* для промежуточных страниц)
  – K = *среднийРазмерКлюча* для промежуточных страниц

  Значения *минимальныйРазмерКлюча* и *среднийРазмерКлюча* для промежуточных страниц должны совпадать со значениями для конечных страниц, кроме случаев, когда есть включаемые столбцы. На промежуточных страницах включаемые столбцы не хранятся.

  Не следует заменять `0,9` на `(100 - процентов_свободного_места)/100`, если это значение меньше, чем `0,9`, так как при создании индекса на промежуточных страницах оставляется максимум 10 процентов свободного места.

  Число промежуточных страниц можно оценить следующим образом:

```
if L > 1 then {P++; Z++}
While (Y > 1)
{
   P = P + Y
   Y = Y / N
  Z++
}
```

где:

- P - число страниц (изначально 0).
- L - число конечных страниц.
- N - число ключей для каждой промежуточной страницы.
- Y = L / N
- Z - количество уровней в дереве индекса (изначально 1).

Общее число страниц равно:

```
T = (L + P + 2) * 1,0002
```

Дополнительные 0,02 процента отражают дополнительное место, включая страницы с картами пространства.

Необходимый для создания индекса объем места оценивается следующим образом:

```
T * размер_страницы
```

Дополнительные требования к пространству

Необходимо также следующее дополнительное пространство:

- “Пространство для файлов журнала”
- “Временное рабочее пространство” на стр. 134

Пространство для файлов журнала

Объем пространства (в байтах), необходимого для файлов журнала, колеблется от

```
(число_первичн*(размер_файла_журн+2)*4096)+8192
```

до:

```
((число_первичн+число_вторичн)*(размер_файла_журн+2)*4096)+8192
```

где:

- *число_первичных* - число первичных файлов журнала, определенное в файле конфигурации базы данных
- *число_вторичных* - число вторичных файлов журнала, определенное в файле конфигурации базы данных

- *размер_файла_журнала* - число страниц в каждом файле журнала, определенное в файле конфигурации базы данных
- 2 - число страниц заголовка, требуемых для каждого файла журнала
- 4096 - число байт на одной странице
- 8192 - размер (в байтах) файла управления журналом.

Дополнительную информацию об этих параметрах конфигурации смотрите в книге *Руководство администратора: Производительность*.

Примечание: Общее пространство активного журнала не должно превышать 32 Гбайт.

Верхний предел пространства файлов журнала зависит от фактического числа вторичных файлов журнала, необходимых для работы менеджера баз данных. Этот предел может никогда не потребоваться или требоваться только в отдельные периоды повышенной активности.

Если база данных допускает восстановление с повтором транзакций, необходимо учитывать дополнительные требования к пространству журнала:

- Если включен параметр конфигурации *logretain*, файлы журнала будут архивироваться в каталоге журнала. Место на диске в какой-то момент заполнится, если файлы журнала не перемещать в другое место.
- Если включен параметр конфигурации *userexit*, архивные файлы журнала перемещаются в другое место обработчиком пользователя. Дополнительное пространство журнала все равно требуется для:
  - Оперативных архивных журналов, ожидающих перемещения обработчиком пользователя
  - Новых файлов журнала, форматируемых для будущего использования.

Временное рабочее пространство

Некоторым операторам SQL для работы необходимы временные таблицы (например, рабочий файл для операций сортировки, которые нельзя провести в памяти). Для этих временных таблиц необходимо дисковое пространство; объем требуемого пространства зависит от запросов и размера возвращаемых таблиц, и его нельзя оценить даже приблизительно.

Чтобы отследить, какой объем рабочего пространства используется при обычной работе, можно использовать системный монитор баз данных и API запросов табличного пространства.

Проектирование групп узлов

Группа узлов - это именованное множество из одного или нескольких узлов, определенных как принадлежащие некоторой базе данных. Все разделы базы данных, являющиеся частью конфигурации системы базы данных, должны быть заранее определены в *файле конфигурации разделов*, который называется `db2nodes.cfg`. Группа узлов может содержать от одного раздела базы данных до всего множества разделов базы данных, определенных для системы базы данных.

Новая группа узлов создается при помощи оператора CREATE NODEGROUP; ее можно изменить при помощи оператора ALTER NODEGROUP. В группе узлов можно добавить или отбросить один или несколько разделов базы данных. До изменения группы узлов разделы базы данных должны быть определены в файле `db2nodes.cfg`. Внутри групп узлов находятся табличные пространства. Внутри табличных пространств находятся таблицы.

При создании или изменении группы узлов с ней связывается *карта разделения*. Карта разделения совместно с *ключом разделения* и алгоритмом хеширования используется менеджером баз данных для определения, на каком разделе базы данных в группе узлов будет храниться заданная строка данных. Дополнительную информацию о картах разделения смотрите в разделе “Карты разделения” на стр. 137. Дополнительную информацию о ключах разделения смотрите в разделе “Ключи разделения” на стр. 139.

В одноразделной базе данных не требуется ни ключа разделения, ни карты разделения. При использовании одноразделной базы данных нет необходимости проектировать группы узлов. *Раздел базы данных* - это часть базы данных, заполненная пользовательскими данными, индексами, файлами конфигурации и журналами транзакций. Менеджер баз данных использует группы узлов, созданные по умолчанию при создании базы данных. IBMCATGROUP - группа узлов по умолчанию для табличного пространства, содержащего системные каталоги. IBMTEMPGROUP - группа узлов по умолчанию для системных временных табличных пространств. IBMDEFAULTGROUP - группа узлов по умолчанию для табличных пространств, содержащих пользовательские таблицы, которые вы решите туда поместить. Пользовательское временное табличное пространство для объявленной временной таблицы может находиться в IBMDEFAULTGROUP или в любой группе узлов, созданной пользователем, но не в IBMTEMPGROUP.

Если вы используете многораздельную группу узлов, учтите следующие особенности проектирования:

- В многораздельной группе узлов индекс уникальности можно создать, только если он является надмножеством ключа разделения.

- В зависимости от числа разделов в базе данных может быть одна или несколько одноразделных групп узлов и одна или несколько многоразделных групп узлов.
- Каждому разделу базы данных должен быть назначен уникальный номер раздела. Один раздел базы данных может находиться в одной или в нескольких группах узлов.
- Для быстрого восстановления раздела базы данных, содержащего таблицы системного каталога, не размещайте на том же разделе базы данных пользовательские таблицы. Этого можно достичь, поместив пользовательские таблицы в группы узлов, в которые не включен раздел базы данных в группе узлов IBMCATGROUP.

Небольшие таблицы следует помещать в одноразделные группы узлов, если только вам не требуется их *совместное размещение* с большей таблицей. Совместное размещение - это размещение строк из разных таблиц, содержащих связанные данные, в одном разделе базы данных. Совместно размещенные таблицы позволяют DB2 использовать более эффективные стратегии объединения. Совместно размещенные таблицы могут располагаться в одноразделной группе узлов. Таблицы считаются совместно размещенными, если они находятся в многораздельной группе узлов, в их ключах разделения одинаковое число столбцов, а типы данных соответствующих столбцов совместимы с разделами. Строки в совместно размещенных таблицах с одинаковым значением ключа разделения помещаются в один и тот же раздел базы данных. Таблицы могут находиться в разных табличных пространствах одной группы узлов; они все равно будут считаться совместно размещенными.

Следует избегать растягивания таблиц среднего размера на слишком большое количество разделов базы данных. Например, таблица размером 100 Мбайт может лучше работать в группе узлов из 16 разделов, чем в группе узлов из 32 разделов.

Группы узлов можно использовать для отделения таблиц сетевой обработки транзакций (OLTP) от таблиц поддержки решений (DSS), чтобы избежать снижения производительности транзакций OLTP.

Особенности проектирования групп узлов

Необходимость в разделении базы данных зависит от ее логической структуры и объема обрабатываемых данных. В этом разделе описываются следующие темы, связанные с разделением базы данных:

- “Разделение данных” на стр. 137
- “Карты разделения” на стр. 137
- “Ключи разделения” на стр. 139
- “Совместное размещение таблиц” на стр. 141
- “Совместимость с разделами” на стр. 141

- “Реплицируемые сводные таблицы” на стр. 142

Разделение данных

DB2 поддерживает модель многораздельной памяти, позволяющую хранить данные на нескольких разделах базы данных. Это означает, что данные физически хранятся на нескольких разделах базы данных, но доступ к ним осуществляется так, как если бы они находились в одном месте. Программам и пользователям, обращающимся к данным многораздельной базы данных, не нужно знать физического местоположения этих данных.

Хотя эти данные физически разделены, при использовании и управлении они рассматриваются, как логическое целое. Пользователи могут выбирать способ разделения данных, объявляя ключи разделения. Выбрав табличное пространство и соответствующую группу узлов, в которых должны храниться табличные данные, пользователи могут также определять, по скольким разделам базы данных и как могут быть распределены эти данные. Кроме того, для задания соответствия между значениями ключа разделения и разделами базы данных, которое определяет размещение и получение каждой строки данных, используется изменяемая карта разделения с алгоритмом хеширования. Так можно распределить нагрузку по многораздельной базе данных для больших таблиц, позволяя хранить меньшие таблицы на одном или нескольких разделах базы данных. На каждом разделе базы данных есть локальные индексы для хранящихся на нем данных, что увеличивает производительность при доступе к локальным данным.

Вы не обязаны разделять все таблицы между всеми разделами в базе данных. DB2 поддерживает *частичное рассеяние* - это значит, что таблицы и их табличные пространства можно распределить по подмножеству разделов базы данных в системе (то есть по группе узлов).

Второй возможный вариант для размещения таблиц на каждом разделе базы данных - использование таблиц сводок и последующая их репликация. Вы можете создать таблицу сводок, содержащую всю необходимую информацию, а затем реплицировать ее на все узлы. Дополнительную информацию смотрите в разделе “Реплицируемые сводные таблицы” на стр. 142.

Карты разделения

В среде многораздельных баз данных у менеджера баз данных должен быть способ определения, на каком разделе базы данных находятся какие строки таблицы. Менеджер баз данных должен знать, где найти требуемые данные, и находит их при помощи карты, которая называется *картой разделения*.

Карта разделения - это создаваемый внутри программы массив, содержащий 4096 элементов для многораздельных групп узлов или один элемент для одноразделных групп узлов. Для одноразделной группы узлов в карте разделения есть только один элемент, содержащий номер раздела базы данных,

в котором хранятся все строки таблицы базы данных. Для многораздельных групп узлов циклически задаются номера разделов группы узлов. Как план города разделяется сеткой на части, так и менеджер баз данных при помощи *ключа разделения* определяет место (раздел базы данных), где хранятся данные.

Допустим, например, что у вас есть база данных с четырьмя разделами, пронумерованными от 0 до 3. Карта разделения группы узлов IBMDEFAULTGROUP этой базы данных будет иметь вид:

```
0 1 2 3 0 1 2 ...
```

Если создать в этой базе данных группу узлов, использующую разделы 1 и 2, карта разделения для этой группы узлов будет иметь вид:

```
1 2 1 2 1 2 1 ...
```

Если ключ разделения для загружаемой в базу данных таблицы - целое число с возможными значениями от 1 до 500000, ключ разделения хешируется в номер раздела от 0 до 4095. Это число используется в качестве указателя внутри карты разделения для выбора раздела базы данных для этой строки.

На рис. 35 показана строка со значением ключа разделения (c1, c2, c3), отображаемая в раздел 2, который, в свою очередь, указывает на раздел базы данных n5.

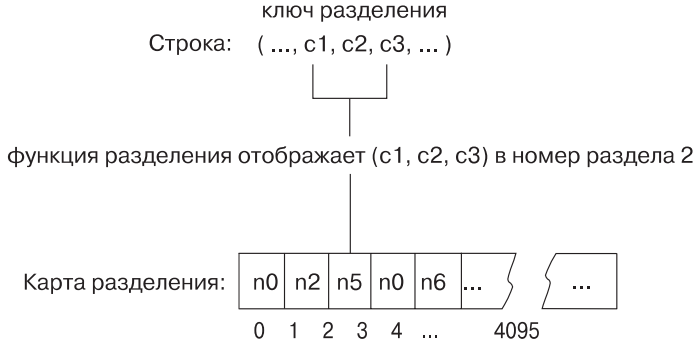


Рисунок 35. Распределение данных при помощи карты разделения

Карта разделения - это гибкий способ управления местом хранения данных в многораздельной базе данных. Если в будущем вам потребуется изменить в базе данных распределение данных по разделам базы данных, воспользуйтесь утилитой перераспределения данных. Эта утилита позволяет перебалансировать распределение данных или внести в него асимметрию. Дополнительную информацию об этой утилите смотрите в разделе ″Перераспределение данных между разделами базы данных″ книги *Руководство администратора: Производительность*.

Чтобы получить для просмотра копию карты разделения, можно воспользоваться API получения информации о разделении таблицы (**sqlugtpi**). Дополнительную информацию об этом API смотрите в руководстве *Administrative API Reference*.

Ключи разделения

Ключ разделения - это столбец (или группа столбцов), используемый для определения раздела, в котором хранится заданная строка данных. Ключ разделения для таблицы задается оператором CREATE TABLE. Если ключ разделения не задан для таблицы в табличном пространстве, которое распределено по нескольким разделам базы данных в группе узлов, он создается по умолчанию из первого столбца первичного ключа. Если первичный ключ не задан вообще, в качестве ключа разделения по умолчанию принимается столбец первого недлинного поля, определенного в данной таблице. (К *длинным* типам относятся все длинные типы данных и все типы данных больших объектов). Если вы создаете таблицу в табличном пространстве, связанном с одноразделной группой узлов, и вам нужен ключ разделения, вы должны задать его в явном виде. Он не будет создан по умолчанию.

Если ни один столбец не отвечает требованиям для ключа разделения по умолчанию, таблица создается без ключа разделения. Таблицы без ключа разделения допускаются только в одноразделных группах узлов. Ключи разделения можно добавить или отбросить позднее при помощи оператора ALTER TABLE. Изменение ключа разделения допустимо только для таблицы, табличное пространство которой связано с одноразделной группой узлов.

Важно выбрать правильный ключ разделения. При этом необходимо учесть:

- Способ доступа к таблицам
- Характер нагрузки запросов
- Стратегии объединения, используемые системой базы данных.

Если вы не собираетесь использовать совместное размещение, лучший ключ разделения таблицы - ключ, равномерно распределяющий данные по всем разделам базы данных в группе узлов. Совместное размещение таблиц определяется ключом разделения для каждой таблицы в табличном пространстве, связанном с группой узлов. Таблицы считаются совместно размещенными, если:

- Таблицы расположены в табличных пространствах, находящихся в одной группе узлов
- В ключах разделения всех таблиц участвует одно и то же число столбцов
- Типы данных соответствующих столбцов совместимы с разделами.

Строки совместно размещенных таблиц с одинаковыми значениями ключа разделения расположены на одном и том же разделе. Дополнительную информацию о совместимости с разделами смотрите в разделе “Совместимость

с разделами” на стр. 141. Дополнительную информацию о совместном размещении таблиц смотрите в разделе “Совместное размещение таблиц” на стр. 141.

Неподходящий ключ разделения может вызвать неравномерное распределение данных. В качестве ключа разделения не следует выбирать столбцы с неравномерно распределенными данными и столбцы с малым количеством различных значений. Различных значений должно быть достаточно, чтобы обеспечить равномерное распределение строк между всеми разделами базы данных в группе узлов. Стоимость применения хеш-алгоритма разделения пропорциональна размеру ключа разделения. В ключ разделения может входить не больше 16 столбцов, но меньшее число столбцов улучшает производительность. В ключ разделения не следует включать столбцы без необходимости.

При определении ключей разделения необходимо учитывать следующие особенности:

- Не поддерживается создание многораздельных таблиц, содержащих только длинные типы данных (LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB и DBCLOB).
- Определение ключа разделения нельзя изменить.
- В строке таблицы нельзя изменить значение столбца, входящего в ключ разделения.
- Значения столбца, входящего в ключ разделения, можно только удалять и вставлять.
- В ключ разделения должны входить столбцы, по которым чаще выполняется объединение.
- Ключ разделения должен состоять из столбцов, часто используемых в условии GROUP BY.
- В любой уникальный ключ или первичный ключ должны входить все столбцы, входящие в ключ разделения.
- В среде сетевой обработки транзакций (OLTP) все столбцы, входящие в ключ разделения, должны участвовать в транзакции с использованием условия равенства (=) константам или переменным хоста. Допустим, например, что у вас есть номер сотрудника *emp_no*, часто используемый в транзакциях типа:

  ```
  UPDATE emp_table SET ... WHERE
  emp_no = переменная-хоста
  ```

  В этом случае столбец EMP_NO будет для EMP_TABLE удачным ключом разделения.

Хеш-разделение - это метод, которым определяется место каждой строки в многораздельной таблице. Он работает так:

1. Хеш-алгоритм, применяемый к значению ключа разделения, выдает номер раздела от нуля до 4095.
2. При создании группы узлов создается карта разделения. Карта разделения циклически заполняется последовательным перебором номеров разделов. Дополнительную информацию о картах разделения смотрите в разделе “Карты разделения” на стр. 137.
3. Полученный номер раздела используется в качестве указателя внутри карты разделения. Число в этой позиции карты разделения - это номер раздела базы данных, на котором хранится требуемая строка.

Совместное размещение таблиц

Вы можете обнаружить, что в ответ на определенные запросы часто выдаются данные из нескольких таблиц. В этом случае нужно, чтобы связанные данные из таких таблиц были расположены как можно ближе друг к другу. В среде, где база данных физически разделена между несколькими разделами базы данных, должен быть способ держать связанные части разделенных таблиц как можно ближе друг к другу. Такая возможность называется *совместным размещением таблиц*.

Таблицы совместно размещены, если они хранятся в одной и той же группе узлов и их ключи разделения совместимы. Если несколько таблиц помещены в одну группу узлов, для них используется общая карта разделения. Таблицы могут находиться в разных табличных пространствах, но они должны быть связаны с одной группой узлов. Типы данных связанных столбцов во всех ключах разделения должны быть *совместимы с разделами*. Информацию о совместимости с разделами смотрите в разделе “Совместимость с разделами”.

При доступе к нескольким таблицам для объединения или подзапроса DB2 может распознать, что объединяемые данные находятся на одном и том же разделе базы данных. Если это происходит, DB2 может провести объединение или подзапрос на том разделе базы данных, где хранятся данные, вместо того чтобы перемещать данные между разделами базы данных. Эта возможность проводить объединения и подзапросы на том же разделе базы данных дает значительное улучшение производительности. Дополнительную информацию смотрите в разделе ″Совместно размещенные объединения″ книги *Руководство администратора: Производительность*.

Совместимость с разделами

Базовые типы данных связанных столбцов ключей разделения сравниваются, после чего могут быть объявлены *совместимыми по разделу*. Для совместимых по разделу типов две переменных, по одной для каждого типа, равные по значению, отображаются одним и тем же алгоритмом разделения в один и тот же номер раздела.

Совместимость по разделу обладает следующими свойствами:

- Базовый тип данных совместим с пользовательскими типами, производными от него.
- Для типов данных DATE, TIME и TIMESTAMP используются внутренние форматы. Они несовместимы друг с другом, и ни один из них не совместим с CHAR.
- На совместимость по разделу не влияют определения столбцов NOT NULL или FOR BIT DATA.
- Значения NULL совместимых типов данных обрабатываются одинаково; для несовместимых типов данных это не обязательно так.
- Для проверки на совместимость по разделу используются базовые типы данных пользовательских типов.
- Десятичные числа с одним значением в ключе разделения обрабатываются одинаково, даже если их масштаб и точность различаются.
- Хеш-алгоритм не учитывает конечные пробелы в строках символов (CHAR, VARCHAR, GRAPHIC и VARGRAPHIC).
- BIGINT, SMALLINT и INTEGER - совместимые типы данных.
- REAL и FLOAT - совместимые типы данных.
- CHAR и VARCHAR разной длины - совместимые типы данных.
- GRAPHIC и VARGRAPHIC - совместимые типы данных.
- Совместимость по разделу неприменима к типам данных LONG VARCHAR, LONG VARGRAPHIC, CLOB, DBCLOB и BLOB, поскольку они не поддерживаются в качестве ключей разделения.

Реплицируемые сводные таблицы

Сводная таблица - это таблица, определенная запросом, который используется также для определения данных в таблице. Сводные таблицы можно использовать для улучшения производительности запросов. Если DB2 определяет, что часть запроса можно упростить с использованием сводной таблицы, менеджер баз данных может переписать запрос с использованием сводной таблицы.

В среде многораздельных баз данных сводные таблицы можно реплицировать. *Реплицируемые сводные таблицы* можно использовать для улучшения производительности запросов. Реплицируемая сводная таблица основана на таблице, которая может быть создана в одноразделной группе узлов, но вы хотите реплицировать ее по всем разделам базы данных в этой группе узлов. Чтобы создать реплицируемую сводную таблицу, укажите в операторе CREATE TABLE ключевое слово REPLICATED. Ключевое слово REPLICATED можно задавать только для сводной таблицы, определенной с опцией REFRESH DEFERRED.

Подробную информацию о сводных таблицах смотрите в разделе ″Создание сводных таблиц″ книги *Руководство администратора: Реализация*.

При помощи реплицируемых сводных таблиц можно добиться совместного размещения таблиц, для которых обычно это невозможно. Реплицируемые сводные таблицы в особенности удобны для объединений, содержащих большую таблицу фактов и маленькие таблицы ассоциаций. Чтобы минимизировать необходимую дополнительную память, а также влияние необходимости обновления каждой реплики, реплицируемые таблицы должны быть маленькими и редко обновляемыми.

Примечание: Возможно, имеет смысл реплицирование редко обновляемых больших таблиц: одноразовые расходы на репликацию возмещаются выигрышем в быстродействии, который дает совместное размещение.

Указав соответствующий предикат в условии подвыборки, используемом для определения реплицируемой таблицы, можно реплицировать выбранные столбцы, выбранные строки или и то, и другое.

Дополнительную информацию о реплицируемых сводных таблицах смотрите в описании оператора CREATE TABLE в справочнике *SQL Reference*. Дополнительную информацию о совместно размещенных объединениях смотрите в разделе "Совместные объединения" книги *Руководство администратора: Реализация*.

Проектирование и выбор табличных пространств

Табличное пространство - это модель хранения, соответствующая промежуточному уровню между базой данных и таблицами, хранящимися в этой базе данных. Табличные пространства находятся в группах узлов. Они позволяют напрямую назначать контейнерам местонахождение базы данных и табличных данных. (Контейнером может быть каталог, устройство или файл.) Это может улучшить производительность и целостность, а также сделать конфигурацию более гибкой.

Дополнительную информацию о создании и изменении табличного пространства смотрите в разделе "Создание табличного пространства" и в разделе "Изменение табличного пространства" книги *Руководство администратора: Реализация*.

Поскольку табличные пространства размещаются в группах узлов, выбранное для хранения таблицы табличное пространство определяет, как данные для этой таблицы распределяются по разделам базы данных в данной группе узлов. Одно табличное пространство может содержать несколько контейнеров. Можно создавать несколько контейнеров (от одного или нескольких табличных пространств) на одном и том же физическом диске. Для улучшения производительности каждый контейнер должен находиться на отдельном диске.

На рис. 36 показано, как таблицы и табличные пространства в базе данных связаны с контейнерами, относящимися к этой базе данных.

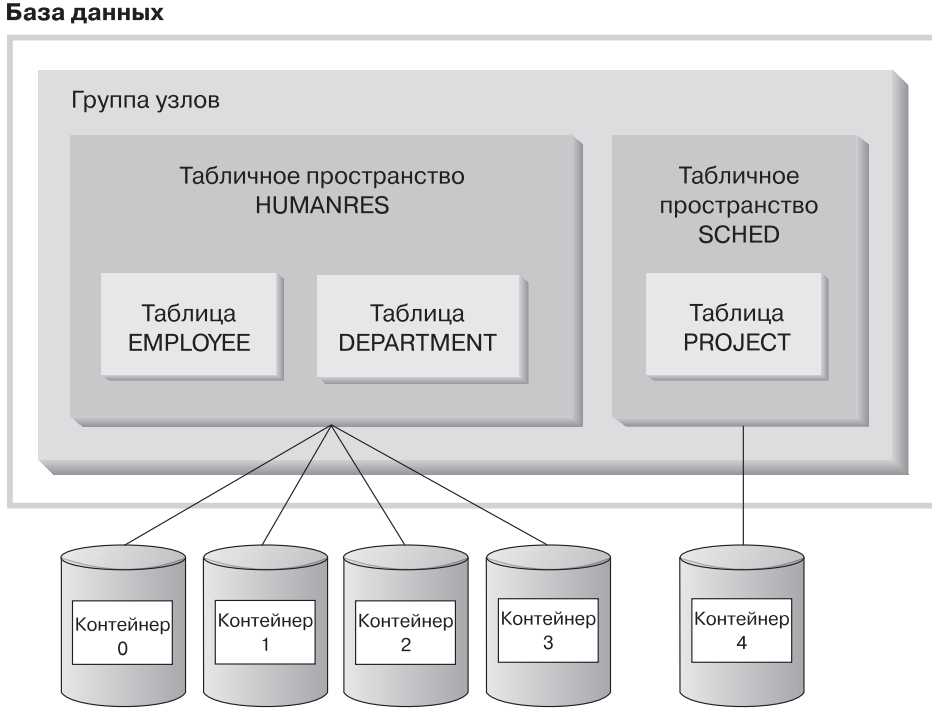


Рисунок 36. Табличные пространства и таблицы в базе данных

Таблицы EMPLOYEE и DEPARTMENT находятся в табличном пространстве HUMANRES, которое занимает контейнеры 0, 1, 2 и 3. Таблица PROJECT находится в табличном пространстве SCHED в контейнере 4. В этом примере каждый контейнер находится на отдельном диске.

Менеджер баз данных пытается уравнять загрузку контейнеров данными. В результате для хранения данных используются все контейнеры. Количество страниц, которое менеджер баз данных записывает в один контейнер, прежде чем начать работать с другим, называется *размером экстента*. Менеджер баз данных не всегда начинает сохранять табличные данные с первого контейнера.

На рис. 37 на стр. 145 показано табличное пространство HUMANRES с размером экстента, равным двум страницам по 4 Кбайта, и четыре контейнера, в каждом из которых выделено небольшое количество экстентов. В таблицах DEPARTMENT и EMPLOYEE по семь страниц; обе они занимают все четыре контейнера.

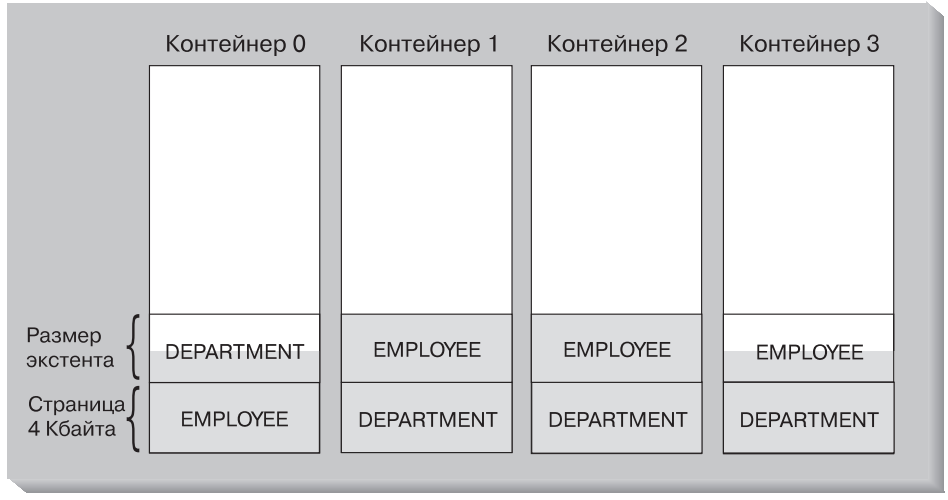


Рисунок 37. Контейнеры и экстенты

В базе данных должно быть по меньшей мере три табличных пространства:

- Одно *табличное пространство каталога*, в котором содержатся все таблицы системного каталога для этой базы данных. Это табличное пространство называется SYSCATSPACE, и его нельзя отбросить. IBMCATGROUP - группа узлов по умолчанию для этого табличного пространства.
- Одно или несколько *пользовательских табличных пространств*, содержащих все пользовательские таблицы. По умолчанию создается одно табличное пространство - USERSPACE1. IBMDEFAULTGROUP - группа узлов по умолчанию для этого табличного пространства.

  При создании таблицы необходимо указывать имя табличного пространства, иначе результаты могут отличаться от ожидаемых. Если не указать имени табличного пространства, таблица будет размещена в соответствии со следующими правилами: Если существуют табличные пространства, созданные пользователем, из них выбирается табличное пространство с наименьшим размером страницы, достаточно большим для этой таблицы. В противном случае будет использовано USERSPACE1, если его размер страницы достаточно велик для этой таблицы. Если нет пользовательских табличных пространств с достаточным размером страницы, таблица не создается.

  Размер страницы таблицы определяется либо размером строки, либо количеством столбцов. Максимально допустимая длина строки зависит от размера страницы табличного пространства, в котором создается таблица. Возможные значения размера страницы - 4 Кбайта (по умолчанию), 8 Кбайт, 16 Кбайт и 32 Кбайта. Можно использовать табличное пространство с одним размером страницы для основной таблицы и табличное пространство с

другим размером страницы для длинных данных и данных больших объектов. (Напомним, что SMS не поддерживает таблиц, занимающих несколько табличных пространств, а DMS поддерживает.) Если количество столбцов или размер строки превышает пределы для размера страницы табличного пространства, возвращается ошибка (SQLSTATE 42997).

- Одно или несколько *временных табличных пространств*, содержащих временные таблицы. Временные табличные пространства делятся на *системные временные табличные пространства* и *пользовательские временные табличные пространства*. В базе данных должно быть по меньшей мере одно системное временное табличное пространство; по умолчанию при создании базы данных создается одно системное временное табличное пространство под именем TEMPSPACE1. IBMTEMPGROUP - группа узлов по умолчанию для этого табличного пространства. Пользовательские временные табличные пространства при создании базы данных по умолчанию *не* создаются.

  Если база данных использует несколько временных табличных пространств, временные объекты циклически располагаются по временным табличным пространствам.

  Если выполняются запросы по таблицам в табличных пространствах, определенных с размером страницы, превышающим 4 Кбайта по умолчанию (например, ORDER BY с 1012 столбцами), некоторые из них могут завершиться неудачно. Это произойдет, если не определено ни одного временного табличного пространства с большим размером страницы. Может потребоваться создать временное табличное пространство с большим размером страницы (8 Кбайт, 16 Кбайт или 32 Кбайта). Любой оператор DML (Data Manipulation Language - язык управления данными) может неудачно завершиться, если нет временного рабочего пространства с размером страницы, совпадающим с наибольшим размером страницы в пользовательском табличном пространстве.

  Нужно определить одно временное табличное пространство SMS с размером страницы, совпадающим с размером страницы большинства пользовательских табличных пространств. Этого должно быть достаточно для типичных состояний среды и рабочей нагрузки. Смотрите также раздел “Рекомендации для временных табличных пространств” на стр. 159.

В среде многораздельной базы данных узел каталога будет содержать все три табличных пространства по умолчанию, а остальные разделы базы данных будут содержать только TEMPSPACE1 и USERSPACE1.

Есть два типа табличных пространств, каждое из которых можно использовать в базе данных:

- “SMS - Пространство, управляемое системой” на стр. 147: Пространство хранения управляется менеджером файлов операционной системы.
- “Табличное пространство, управляемое базой данных” на стр. 151: Пространство хранения управляется менеджером баз данных.

SMS - Пространство, управляемое системой

В табличном пространстве SMS (System Managed Space - пространство, управляемое системой) пространство, в котором хранится таблица, выделяется и управляется менеджером файловой системы операционной системы. Модель хранения обычно состоит из множества файлов, представляющих табличные объекты, которые хранятся в пространстве файловой системы. Пользователь выбирает местоположение файлов, DB2 управляет их именами, а файловая система отвечает за управление ими. Управляя объемом данных, записываемых в каждый файл, менеджер баз данных равномерно распределяет данные по контейнерам табличного пространства. По умолчанию используется табличное пространство SMS.

С каждой таблицей связан по меньшей мере один физический файл SMS. Список этих файлов и описание их содержимого смотрите в разделе “Физические файлы SMS” на стр. 149.

В табличном пространстве SMS при увеличении объекта соответствующий файл увеличивается по одной странице за раз. Если вам нужно улучшить производительность вставки, возможно, следует разрешить многостраничное размещение файлов. Это позволяет системе размещать или увеличивать файл больше, чем по одной странице за раз. Чтобы разрешить многостраничное размещение файлов, запустите **db2empfa**. В среде многораздельной базы данных эту утилиту нужно запустить на всех разделах базы данных. После того как многостраничное размещение файлов включено, выключить его невозможно. Дополнительную информацию о **db2empfa** смотрите в руководстве *Command Reference*.

Табличные пространства SMS нужно определять в явном виде при помощи опции MANAGED BY SYSTEM команды CREATE DATABASE или оператора CREATE TABLESPACE. При проектировании табличных пространств SMS следует учитывать два ключевых фактора:

- Контейнеры для табличного пространства.

  Нужно задать число контейнеров, которые будут использоваться для этого табличного пространства. Очень важно задать все нужные контейнеры, так как после создания табличного пространства SMS контейнеры нельзя ни добавлять, ни удалять. В среде многораздельной базы данных, если в группу узлов для табличного пространства SMS добавляется новый раздел, добавить в новый раздел контейнеры можно при помощи оператора ALTER TABLESPACE.

  Каждый используемый для табличного пространства SMS контейнер задает полное или относительное имя каталога. Каждый из этих каталогов может находиться в другой файловой системе (или на другом физическом диске). Максимальный размер табличного пространства можно оценить так:

  ```
  число контейнеров * (максимальный размер файловой системы,
     поддерживаемый операционной системой)
  ```

Эта формула предполагает, что каждому контейнеру назначена отдельная файловая система и что в каждой файловой системе свободно максимальное количество места. Практически это не обязательно так, и максимальный размер табличного пространства может быть гораздо меньше.

Примечание: Контейнеры нужно определять с осторожностью. Если на них уже существуют файлы или каталоги, возвращается ошибка (SQL0298N).

- Размер экстента для табличного пространства.

  Размер экстента можно задать только при создании табличного пространства. Поскольку размер экстента нельзя изменить позже, важно выбрать для него подходящее значение. Дополнительную информацию смотрите в разделе “Выбор размера экстента” на стр. 158.

  Если при создании табличного пространства не указать размера экстента, менеджер баз данных создаст табличное пространство с размером экстента по умолчанию, заданным параметром конфигурации базы данных *dft_extent_sz* (дополнительную информацию об этом параметре смотрите в книге *Руководство администратора: Производительность*). Значение этого параметра конфигурации вначале задается на основе информации, введенной при создании базы данных. Если в команде CREATE DATABASE не указан параметр DFT_EXTENT_SZ, размер экстента будет по умолчанию равен 32.

Чтобы выбрать для табличного пространства подходящие значения количества контейнеров и размера экстента, необходимо учитывать:

- Ограничения, накладываемые операционной системой на размер логической файловой системы.

  Например, у некоторых операционных систем есть ограничение в 2 Гбайта. Тогда, если вам нужен табличный объект размером 64 Гбайта, в системе такого типа вам потребуется по меньшей мере 32 контейнера.

  При создании табличного пространства можно задать контейнеры, расположенные на других файловых системах, увеличив таким образом объем данных, которые можно хранить в базе данных.

- Как менеджер баз данных управляет файлами данных и контейнерами, связанными с табличным пространством.

  Первый файл табличных данных (`SQL00001.DAT`) создается в первом контейнере, указанном для табличного пространства, и этот файл может расти, пока не достигнет размера экстента. Когда он достигает этого размера, менеджер баз данных начинает записывать данные в `SQL00001.DAT` в следующем контейнере. Этот процесс продолжается, пока файлы `SQL00001.DAT` не появятся во всех контейнерах, после чего менеджер баз данных вернется к первому контейнеру. Этот процесс (называющийся *чередованием*) продолжает применяться к каталогам контейнеров, пока какой-нибудь контейнер не заполнится (SQL0289N) или операционная система не откажется выделять дополнительное место (ошибка переполнения диска).

Чередование используется также для файлов индексов (SQL*nnnnn*.INX), длинных полей (SQL*nnnnn*.LF) и LOB (SQL*nnnnn*.LB и SQL*nnnnn*.LBA).

Примечание: Табличное пространство SMS заполнено, когда заполнен хотя бы один из его контейнеров. Следовательно, важно выделять для всех контейнеров одинаковое место.

Чтобы данные распределялись по контейнерам более равномерно, менеджер баз данных определяет, с какого контейнера начать, взяв остаток от деления идентификатора таблицы (в последнем примере - 1) на количество контейнеров. Контейнеры нумеруются последовательно, начиная с 0.

Дополнительную информацию о файлах, используемых в табличном пространстве SMS, смотрите в разделе “Физические файлы SMS”.

Физические файлы SMS

В контейнере каталога табличного пространства SMS находятся следующие файлы:

Имя файла	Описание
SQLTAG.NAM	В каждом подкаталоге контейнера есть этот файл; они используются менеджером баз данных при соединении с базой данных для проверки ее целостности.
SQLxxxxx.DAT	Табличный файл. Здесь хранятся все строки таблицы, за исключением данных типов LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB и DBCLOB.
SQLxxxxx.LF	Файл, содержащий данные типов LONG VARCHAR и LONG VARGRAPHIC (называемые также ″данными длинных полей″). Этот файл создается, только если в таблице есть столбцы LONG VARCHAR или LONG VARGRAPHIC.
SQLxxxxx.LB	Файлы, содержащие данные типов BLOB, CLOB и DBCLOB (называемые также ″данными больших объектов″). Эти файлы создаются, только если в таблице есть столбцы BLOB, CLOB или DBCLOB.
SQLxxxxx.LBA	Файлы, содержащие информацию о размещении и свободном месте в файлах SQL*xxxxx*.LB.
SQLxxxxx.INX	Файл индекса для таблицы. Все индексы для соответствующей таблицы хранятся в этом файле. Он создается, только если определены какие-нибудь индексы.

Примечание: При отбрасывании индекса занимаемое им пространство физически не освобождается из файла индекса (.INX), пока файл индекса не будет удален. Файл индекса удаляется при отбрасывании индексов для данной таблицы (и принятии изменений) или при реорганизации таблицы. Если файл индекса не удаляется, после принятия отбрасывания индекса пространство будет помечено в качестве свободного и затем повторно использовано для будущих создаваемых индексов или для работы с индексами.

SQLxxxxx.DTR

Временный файл данных для реорганизации файла DAT. При реорганизации таблицы утилита reorg (посредством команды REORG TABLE) создает таблицу в одном из системных временных табличных пространств. Можно задать использование для этих временных табличных пространств вместо контейнеров, используемых для пользовательских таблиц, других контейнеров.

SQLxxxxx.LFR

Временный файл данных для реорганизации файла LF. При реорганизации таблицы утилита reorg (посредством команды REORG TABLE) создает таблицу в одном из системных временных табличных пространств. Можно задать использование для этих временных табличных пространств вместо контейнеров, используемых для пользовательских таблиц, других контейнеров.

SQLxxxxx.RLB

Временный файл данных для реорганизации файла LB. При реорганизации таблицы утилита reorg (посредством команды REORG TABLE) создает таблицу в одном из системных временных табличных пространств. Можно задать использование для этих временных табличных пространств вместо контейнеров, используемых для пользовательских таблиц, других контейнеров.

SQLxxxxx.RBA

Временный файл данных для реорганизации файла LBA. При реорганизации таблицы утилита reorg (посредством команды REORG TABLE) создает таблицу в одном из системных временных табличных пространств. Можно задать использование для этих временных табличных пространств вместо контейнеров, используемых для пользовательских таблиц, других контейнеров.

Примечания:

1. *Не* изменяйте эти файлы непосредственно. Обращаться к ним можно только через документированные API и инструменты, использующие эти API, включая процессор командной строки и Центр управления DB2.
2. Не перемещайте эти файлы.
3. Не удаляйте эти файлы.
4. Единственный поддерживаемый способ резервного копирования базы данных или табличного пространства - через API **sqlubkp** (Резервное копирование базы данных), а также через использующие этот API процессор командной строки и Центр управления DB2.

Табличное пространство, управляемое базой данных

В табличном пространстве DMS (Database Managed Space - пространство, управляемое базой данных) пространство хранения управляется менеджером баз данных. Эта модель хранения состоит из ограниченного числа устройств, пространство на которых управляется DB2. Администратор решает, какие устройства использовать, и DB2 управляет пространством на этих устройствах. В сущности, такое табличное пространство - реализация файловой системы специального назначения, разработанной для наилучшего соответствия нуждам менеджера баз данных. В определение табличного пространства входит список принадлежащих этому табличному пространству устройств или файлов, в которых можно хранить данные.

Табличное пространство DMS, содержащее пользовательские таблицы, можно определить, как:

- *Обычное* табличное пространство для хранения нормальных данных таблиц и индексов
- *Длинное* табличное пространство для хранения данных длинных полей и больших объектов.

При проектировании табличных пространств и контейнеров DMS необходимо учитывать следующее:

- Менеджер баз данных использует чередование для равномерного распределения данных по всем контейнерам.
- Максимальный размер обычных табличных пространств - 64 Гбайта для страниц по 4 Кбайта; 128 Гбайт для страниц по 8 Кбайт; 256 Гбайт для страниц по 16 Кбайт; и 512 Гбайт для страниц по 32 Кбайта. Максимальный размер длинных табличных пространств - 2 Тбайта.
- В отличие от табличных пространств SMS, контейнеры, из которых состоит табличное пространство DMS, могут быть разного размера, хотя это и не рекомендуется, так как это вызывает неравномерность при чередовании между контейнерами и плохо влияет на производительность. Если контейнер заполнен, табличные пространства DMS используют свободное пространство других контейнеров.

- Поскольку место выделяется заранее, его должно быть достаточно на момент создания табличного пространства. При использовании контейнеров-устройств для определения контейнера необходимо, чтобы существовало достаточно вместительное устройство. На каждом устройстве можно определить только один контейнер. Чтобы избежать потерь пространства, размер устройства должен быть равен размеру контейнера. Например, если на устройстве выделено 5000 страниц, а при определении контейнера выделяется 3000 страниц, 2000 страниц этого устройства нельзя будет использовать.
- В каждом контейнере зарезервирована одна дополнительная страница, а остальные таблицы будут использоваться по одному экстенту за раз. Используются только целые экстенты, так что для наилучшего управления пространством можно при выделении контейнера определять подходящий размер при помощи следующей формулы:

  ```
  (размер_экстента * n) + 1
  ```

  где *размер_экстента* - размер каждого экстента в этом табличном пространстве, а *n* - число экстентов, которое вы хотите хранить в этом контейнере.
- В табличном пространстве резервируются три дополнительных экстента.
- Для хранения любых данных пользовательских таблиц требуются по меньшей мере два экстента. (Эти экстенты нужны для обычных данных таблицы, а не для данных индексов, длинных полей или больших объектов, для которых нужны отдельные экстенты.)
- Контейнеры-устройства должны использовать логические тома с ″символьным специальным интерфейсом″, а не физические тома.
- Для табличных пространств DMS можно использовать файлы вместо устройств. Между файлом и устройством нет функциональной разницы, но файл может быть менее эффективным из-за дополнительного расхода времени при работе через файловую систему. Файлы полезны, если:
  - Устройства не поддерживаются напрямую
  - Устройство недоступно
  - Не требуется максимальная производительность
  - Вы не хотите заниматься конфигурированием устройств.
- Если в рабочую нагрузку входят данные больших объектов или LONG VARCHAR, улучшение производительности можно получить при кэшировании файловой системы. Обратите внимание на то, что большие объекты и LONG VARCHAR не буферизуются пулом буферов DB2.
- Некоторые операционные системы допускают использование физических устройств размером более 2 Гбайт. Возможно, следует разделить такое физическое устройство на несколько логических устройств, чтобы ни один контейнер не превышал размера, допустимого операционной системой.

Добавление контейнеров в табличные пространства DMS

Оператор ALTER TABLESPACE позволяет добавить контейнер в существующее табличное пространство, чтобы увеличить его. После этого табличное пространство перебалансируется по всем контейнерам. Во время перебалансировки доступ к табличному пространству не ограничивается. Если вам нужно добавить несколько контейнеров, их следует добавить в этот момент, либо в одном операторе ALTER TABLESPACE, либо в той же транзакции, чтобы менеджеру баз данных не приходилось перебалансировать контейнеры несколько раз.

Проверить, насколько заполнены контейнеры для табличного пространства, можно при помощи команд LIST TABLESPACE CONTAINERS или LIST TABLESPACES. Добавлять новые контейнеры следует до того как существующие контейнеры заполнятся почти до конца или же до конца. Новое место на всех контейнерах будет недоступно, пока не завершится перебалансировка.

Добавление контейнера, размер которого меньше существующих, приводит к неравномерному распределению данных. Из-за этого операции параллельного ввода/вывода, как, например, предварительная выборка данных, могут выполняться менее эффективно, чем для контейнеров одного размера.

Особенности структуры табличных пространств

В этой главе рассматриваются следующие темы:

Особенности ввода/вывода для табличных пространств

Эффективность работы ввода/вывода для табличного пространства определяется типом и структурой этого табличного пространства. Ниже изложены основные понятия, которые необходимо знать до того, как продолжить изучение проектирования структуры табличного пространства и его использования:

Чтение больших блоков

Операция чтения, при которой по одному требованию

получается несколько страниц (обычно - экстент). Чтение сразу нескольких страниц более эффективно, чем чтение отдельных страниц.

Предварительная выборка
Чтение страниц до того, как их потребует запрос. Основная цель - уменьшить время ответа. Этого можно достичь, если предварительная выборка страниц может выполняться асинхронно с выполнением запроса. Наилучшее время ответа достигается, если либо процессор, либо подсистема ввода/вывода работают на максимальной мощности.

Очистка страниц
По мере чтения и изменения страниц они накапливаются в пуле буферов базы данных. При считывании страницы она считывается в страницу пула буферов. Если пул буферов заполнен измененными страницами, до того, как можно будет считать новую страницу, надо записать на диск одну из измененных страниц. Чтобы пул буферов не заполнялся, средства очистки страниц записывают на диск измененные страницы, чтобы гарантировать доступность страниц пула буферов для дальнейших требований чтения.

Во всех случаях, когда это дает преимущество, DB2 выполняет чтение больших блоков. Обычно это происходит при получении данных, которые по своей природе последовательны или частично последовательны. Объем данных, считываемых за одну операцию чтения, зависит от размера экстента – чем больше размер экстента, тем больше страниц можно считать за раз.

Способ хранения экстента на диске влияет на эффективность ввода/вывода. В табличном пространстве DMS, использующем контейнеры-устройства, данные имеют тенденцию к последовательному расположению на диске, и их можно прочесть с минимальным временем доступа и работы диска. Но при использовании файлов файловая система может разбить данные и хранить их в нескольких местах диска. Наиболее часто это происходит при использовании табличных пространств SMS, в которых файлы увеличиваются по одной странице за раз, что увеличивает вероятность фрагментации. Большой файл, выделенный под табличное пространство DMS, обычно хранится на диске непрерывно и последовательно, в особенности, если он был выделен в чистой файловой системе.

Степенью предварительной выборки можно управлять при помощи параметра PREFETCHSIZE оператора CREATE TABLESPACE. (Значение по умолчанию для всех табличных пространств в базе данных задается параметром конфигурации баз данных *dft_prefetch_sz*.) Параметр PREFETCHSIZE задает число страниц, которое DB2 считывает при каждой предварительной выборке. Если задать значение PREFETCHSIZE, кратное значению параметра EXTENTSIZE оператора

CREATE TABLESPACE, можно параллельно считывать несколько экстентов. (Значение по умолчанию для всех табличных пространств в базе данных задается параметром конфигурации баз данных *dft_extent_sz*.) Параметр EXTENTSIZE задает число страниц по 4 Кбайта, которое записывается в контейнер перед тем, как перейти к следующему контейнеру.

Допустим, что у вас было табличное пространство, использующее три устройства. Если задать PREFETCHSIZE, в три раза большее, чем EXTENTSIZE, DB2 может параллельно считать большие блоки со всех устройств, таким образом значительно увеличив пропускную способность ввода/вывода. Здесь предполагается, что все устройства являются отдельными физическими устройствами и что пропускная способность контроллера достаточна для обработки потока от каждого устройства. Учтите, что при работе DB2 значения параметров предварительной выборки могут быть динамически изменены в зависимости от скорости запроса, использования пула буферов и других факторов.

Некоторые файловые системы используют собственный метод предварительной выборки (например, файловая система JFS - Journaled File System - в AIX). В некоторых случаях предварительная выборка файловой системы работает более агрессивно, чем предварительная выборка DB2. Это может вызвать видимость превосходства производительности предварительной выборки для табличных пространств SMS и DMS с контейнерами-файлами над предварительной выборкой для табличных пространств DMS с устройствами. Это заблуждение, так как, скорее всего, это результат еще одного уровня предварительной выборки, происходящей в файловой системе. Табличные пространства DMS должны превосходить по производительности любую такую конфигурацию.

Для эффективной предварительной выборки (и даже чтения) нужно, чтобы в буферном пуле было достаточно чистых страниц. Возьмем, например, требование параллельной предварительной выборки, считывающее из табличного пространства три экстента, и при считывании каждой страницы из пула буферов записывается на диск одна измененная страница. Требование предварительной выборки может замедлиться до точки, когда оно перестает успевать за запросом. Чтобы требование предварительной выборки было удовлетворено, нужно сконфигурировать достаточное число чистильщиков страниц. Нужно определить по меньшей мере по одному чистильщику страниц для каждого реально используемого базой данных диска. Дополнительную информацию по этим темам смотрите в книге *Руководство администратора: Производительность*.

Отображение табличных пространств на пулы буферов

Каждое табличное пространство связано с некоторым пулом буферов. Пул буферов по умолчанию - IBMDEFAULTBP. Если табличное пространство надо связать с другим пулом буферов, этот пул буферов должен существовать (определяется оператором CREATE BUFFERPOOL); тогда связь определяется

при создании табличного пространства (при помощи оператора CREATE TABLESPACE). Связь между табличным пространством и пулом буферов можно изменить при помощи оператора ALTER TABLESPACE.

Наличие нескольких пулов буферов позволяет конфигурировать память, используемую базой данных, для улучшения общей производительности. Для табличных пространств с одной или несколькими большими таблицами, к которым нерегулярно обращаются пользователи, можно ограничить размер пула буферов, так как кэширование страниц данных может быть бесполезным. Табличное пространство для программы сетевых транзакций можно связать с большим пулом буферов, чтобы используемые программой страницы данных дольше хранились в кэше, что ускорит время ответа. При конфигурировании новых пулов буферов следует соблюдать осторожность. Дополнительную информацию на эту тему смотрите в разделе ″Управление пулом буферов базы данных″ книги *Руководство администратора: Производительность*.

Примечание: Если вы определили, что базе данных требуются страницы размером 8 Кбайт, 16 Кбайт или 32 Кбайта, все табличные пространства с такими размерами страницы должны отображаться в пулы буферов с тем же размером страницы.

При запуске базы данных менеджеру баз данных должна быть доступна память, необходимая для всех пулов буферов. Если DB2 не может получить требуемую память, менеджер баз данных будет запущен с пулами буферов по умолчанию (по одному с размерами страницы 4 Кбайта, 8 Кбайт, 16 Кбайт и 32 Кбайта) и выдаст предупреждение.

В среде многораздельных баз данных можно создать пул буферов одного и того же размера для всех разделов базы данных. Можно также создавать пулы буферов разных размеров на разных разделах. Дополнительную информацию об операторе CREATE BUFFERPOOL смотрите в справочнике *SQL Reference*.

Отображение табличных пространств на группы узлов

В среде многораздельных баз данных каждое табличное пространство связано с некоторой группой узлов. Это позволяет применить характеристики табличного пространства к каждому узлу в этой группе узлов. Группа узлов должна существовать (определяется оператором CREATE NODEGROUP); тогда связь между табличным пространством и группой узлов определяется при создании табличного пространства оператором CREATE TABLESPACE.

Связь между табличным пространством и группой узлов нельзя изменить при помощи оператора ALTER TABLESPACE. Можно только изменять задание табличного пространства для отдельных разделов этой группы узлов. В среде одноразделных баз данных все табличные пространства связаны с группой узлов по умолчанию. Группа узлов по умолчанию при определении табличного пространства называется IBMDEFAULTGROUP, если определяется не системное

временное табличное пространство; в противном случае используется IBMTEMPGROUP. Дополнительную информацию об операторе CREATE NODEGROUP смотрите в справочнике *SQL Reference*. Дополнительную информацию о группах узлов и физической структуре базы данных смотрите в разделе “Проектирование групп узлов” на стр. 135.

Отображение таблиц в табличные пространства

При определении, как отображать таблицы в табличные пространства, необходимо учитывать:

- Разделение ваших таблиц.

  Как минимум, следует убедиться, что выбранное табличное пространство находится в группе узлов с нужным разделением.

- Количество данных в таблице.

  Если вы собираетесь хранить в табличном пространстве много маленьких таблиц, подумайте об использовании табличного пространства SMS. Преимущества DMS при вводе/выводе и управлении пространством не настолько важны с маленькими таблицами. Такие преимущества SMS, как выделение места по одной странице за раз и только по мере надобности, более привлекательны для маленьких таблиц. Если одна из таблиц большая или же вам нужен ускоренный доступ к данным в таблицах, рекомендуется использовать табличное пространство DMS с маленьким размером экстента.

  Можно использовать по отдельному табличному пространству для всех очень больших таблиц, а все маленькие таблицы сгруппировать в одном табличном пространстве. Такое разделение позволяет также выбрать подходящий размер экстента в зависимости от использования табличного пространства. (Дополнительную информацию смотрите в разделе “Выбор размера экстента” на стр. 158.)

- Тип данных в таблице.

  Допустим, у вас есть таблицы с редко используемыми хронологическими данными; возможно, конечный пользователь готов мириться с большим временем ответа для запросов к этим данных. В этом случае для хронологических таблиц можно использовать отдельное табличное пространство и назначить этому пространству менее дорогие физические устройства с более медленным временем доступа.

  Другой вариант - выбрать несколько важных таблиц, от которых требуется высокая доступность и короткое время ответа. Эти таблицы можно поместить в табличное пространство, назначенное быстрому физическому устройству, которое сможет поддерживать эти требования для важных данных.

  Используя табличные пространства DMS, можно распределить табличные данные по трем разным табличным пространствам: одно для индексных данных, одно для данных больших объектов и длинных полей и одно для обычных табличных данных. Это позволяет выбрать характеристики табличных пространств и физические устройства, на которых они будут

размещаться, лучше всего подходящие для конкретных данных. Например, можно поместить данные индексов на самых быстрых устройствах из имеющихся и в результате получить существенное улучшение производительности. Если таблица разбита по табличным пространствам DMS и включено восстановление с повтором транзакций, есть возможность создавать резервные копии, а также выполнять восстановление одновременно для всех частей таблицы. Табличные пространства SMS не поддерживают этого типа распределения данных по табличным пространствам.

- Управление.

  Некоторые функции управления можно выполнять на уровне табличного пространства вместо уровня базы данных или таблицы. Например, можно сэкономить время и ресурсы, если выполнять резервное копирование табличного пространства, а не базы данных. Это позволяет часто создавать резервные копии табличных пространств с большим объемом изменений, а резервные копии табличных пространств с небольшим объемом изменений создавать только изредка.

  Можно восстановить базу данных или табличное пространство. Если в табличных пространствах нет лишних таблиц, вы можете восстановить только нужную часть базы данных за меньшее время.

  Хорошим подходом является группировка связанных таблиц в набор табличных пространств. Эти таблицы могут быть связаны реляционными ограничениями или другими определенными ограничениями.

  Если вам часто приходится отбрасывать и переопределять некоторую таблицу, возможно, ее следует определить в отдельном табличном пространстве, так как отбрасывать табличное пространство DMS эффективнее, чем отбрасывать таблицу.

Выбор размера экстента

Размер экстента для табличного пространства отражает число страниц табличных данных, которое записывается в контейнер перед тем, как начать записывать данные в следующий контейнер. При выборе размера экстента следует учитывать:

- Размер и тип таблиц в табличном пространстве.

  В табличных пространствах DMS место для таблицы выделяется по одному экстенту за раз. Когда таблица и экстент заполняются, создается новый экстент.

  Таблица состоит из следующих отдельных табличных объектов:

  – Объект данных. Здесь хранятся обычные данные столбцов.

  – Объект индексов. Здесь хранятся все индексы, определенные для этой таблицы.

  – Объект длинных полей. Здесь, если в таблице есть столбцы LONG, хранятся данные длинных полей.

– Два объекта LOB. Если в таблице есть столбцы больших объектов, они хранятся в этих двух табличных объектах:
  - Один табличный объект для данных больших объектов
  - Второй табличный объект для метаданных, описывающих эти данные.

Все табличные объекты хранятся отдельно, и для каждого объекта по мере надобности выделяются новые экстенты. Для каждого табличного объекта имеется объект метаданных, называющийся *картой экстентов*, который описывает все экстенты в этом табличном пространстве, принадлежащие объекту таблицы. Место для карт экстентов также выделяется по одному экстенту за раз.

Таким образом, изначально для таблицы выделяется два экстента для каждого табличного объекта. Если в табличном пространстве слишком много маленьких таблиц, для хранения относительно небольшого количества данных выделяется относительно большое пространство. В этом случае следует задать маленький размер экстента или использовать табличное пространство SMS, которое выделяет по одной странице за раз.

С другой стороны, если у вас есть очень большая таблица, которая быстро растет, и вы используете табличное пространство DMS с маленьким размером экстента, в связи с частым выделением дополнительных экстентов может образоваться ненужное дополнительное место.

- Тип доступа к таблицам.

  Если при доступе к таблицам выполняется много запросов или транзакций, обрабатывающих большие объемы данных, значительное улучшение производительности достигается предварительной выборкой данных. (Информацию о предварительной выборке данных и о ее связи с размером экстента смотрите в книге *Руководство администратора: Производительность*.)

- Минимально необходимое число экстентов.

  Если в контейнерах не хватает места для пяти экстентов табличного пространства, табличное пространство не создается.

Рекомендации для временных табличных пространств

Рекомендуется определить одно временное табличное пространство SMS с размером страницы, совпадающим с размером страницы большинства обычных табличных пространств. Это должно подходить для типичных состояний среды и рабочей нагрузки. Но, возможно, экспериментирование с различными конфигурациями и нагрузками временных табличных пространств может дать лучшие результаты. Необходимо учитывать следующие особенности:

- В большинстве случаев доступ к временным таблицам выполняется последовательно в виде пакетов. Это значит, что вставляется пакет строк или производится чтение пакета последовательных строк. Таким образом, обычно увеличение размера страницы улучшает производительность, так как для

считывания заданного количества данных нужно меньше логических или физических требований страничного ввода/вывода. Это не обязательно так, если средний размер строки временной таблицы меньше размера страницы, деленного на 255. На любой странице, независимо от размера страницы, может быть до 255 строк. Например, для запроса, требующего временную таблицу со строками по 15 байт, лучше использовать временное табличное пространство с размером страницы 4 Кбайта, поскольку 255 таких строк все помещаются в странице размером 4 Кбайта. Размер страницы 8 Кбайт и более вызовет потерю по меньшей мере 4 Кбайт на каждой странице временной таблицы и не уменьшит достаточного числа требований ввода/вывода.

- Если в базе данных больше пятидесяти процентов обычных табличных пространств используют один и тот же размер страницы, вероятно, стоит задать тот же размер страницы для временных табличных пространств. Это позволяет временному табличному пространству совместно использовать один пул буферов вместе с большинством или всеми обычными табличными пространствами. Это, в свою очередь, упрощает настройку пулов буферов.
- При реорганизации таблицы с использованием временного табличного пространства размеры страниц временного табличного пространства и данной таблицы должны совпадать. Поэтому необходимо, чтобы для всех размеров страниц, используемых существующими таблицами, которые могут быть реорганизованы с использованием временного табличного пространства, были определены временные табличные пространства.

  Другой вариант - реорганизация без временного табличного пространства; таблица реорганизуется "на месте", то есть прямо в табличном пространстве назначения. Конечно, для реорганизации "на месте" необходимо, чтобы в табличном пространстве назначения было достаточно места. Дополнительную информацию о реорганизации таблиц смотрите в книге *Руководство администратора: Производительность*.
- Обычно, если существуют временные табличные пространства с разным размером страницы, оптимизатор выбирает временное табличное пространство с наибольшим пулом буферов. Часто в таких случаях имеет смысл назначить одному из временных табличных пространств большой пул буферов, а остальным назначить меньший пул буферов. Такое назначение пулов буферов обеспечивает эффективное использование основной памяти. Например, если табличное пространство каталога использует страницы по 4 Кбайта, а остальные табличные пространства используют страницы по 8 Кбайт, возможно, что лучшая конфигурация временных табличных пространств - одно временное табличное пространство размером 8 Кбайт с большим пулом буферов и одно табличное пространство размером 4 Кбайта с маленьким пулом буферов.

  Примечание: Табличные пространства каталога могут использовать только размер страницы 4 Кбайта. По существу, менеджер баз данных

всегда требует существования временного табличного пространства 4 Кбайта, чтобы выполнять реорганизацию таблиц каталога.

- Как правило, определение нескольких временных табличных пространств с одним размером страницы не дает никаких преимуществ.
- Для временных табличных пространств почти всегда лучше выбирать SMS, чем DMS, поскольку:
  - В SMS дисковое пространство выбирается по требованию, а в DMS оно должно быть выделено заранее. Предварительное выделение может оказаться сложным, поскольку во временных табличных пространствах хранятся кратковременные данные, для которых в пиковые моменты может потребоваться гораздо больше памяти, чем в среднем. В DMS пиковую необходимую память нужно выделить заранее, а в SMS в периоды средней нагрузки эту память можно использовать для других целей.
  - Менеджер баз данных пытается хранить страницы временных таблиц в памяти, а не записывать их на диск. В результате преимущество DMS в производительности становится менее значительным.
  - Контейнеры SMS могут использовать буферизацию файловой системы; контейнеры DMS этого не могут.

Рекомендации по табличным пространствам каталога

Для каталогов баз данных рекомендуется табличное пространство SMS, поскольку:

- Каталог баз данных состоит из множества таблиц разных размеров. При использовании табличного пространства DMS для каждого табличного объекта выделяется минимум два экстента. В зависимости от выбранного размера экстента можно получить значительный объем выделенного, но неиспользуемого места. При использовании табличного пространства DMS следует выбирать маленький размер экстента (от двух до четырех страниц); в противном случае используйте табличное пространство SMS.
- В таблицах каталога есть столбцы больших объектов. Данные больших объектов не хранятся в пуле буферов вместе с остальными данными, а при каждом обращении считываются с диска. Чтение больших объектов с диска уменьшает производительность. Поскольку обычно у файловой системы есть собственное место для хранения (или кэширования) данных, использование табличного пространства SMS или табличного пространства DMS, построенного на контейнерах-файлах, дает возможность избежать ввода/вывода, если к этому большому объекту уже были обращения.

Из-за этих особенностей для каталогов лучше выбирать табличное пространство SMS.

Еще один фактор, который нужно учесть - потребуется ли увеличивать табличное пространство каталога. Некоторые платформы поддерживают

увеличение места, выделенного для контейнеров SMS, и для увеличения табличного пространства SMS можно воспользоваться восстановлением с перенаправлением, однако добавлять новые контейнеры проще при использовании табличного пространства DMS.

Особенности нагрузки

На выбор типа табличных пространств и размера страницы может повлиять преимущественный тип нагрузки DB2 в вашей среде. Нагрузка оперативной обработки транзакций (OLTP) характеризуется транзакциями, которым нужен нерегулярный доступ к данным и которые обычно возвращают небольшие наборы данных. Если доступ нерегулярный и запрашивается одна или несколько страниц, предварительная выборка невозможна.

В такой ситуации лучше всего производительность у табличных пространств DMS, использующих контейнеры-устройства. Если максимальная производительность не важна, для нагрузок OLTP можно использовать табличные пространства DMS с контейнерами-файлами или табличные пространства SMS. Если ожидается малый объем последовательного ввода/вывода или же его не ожидается вовсе, значения параметров EXTENTSIZE и PREFETCHSIZE оператора CREATE TABLESPACE не влияют на эффективность ввода/вывода.

Нагрузка запросов характеризуется транзакциями, которым нужен последовательный или частично последовательный доступ к данным и которые обычно возвращают большие наборы данных. Табличное пространство DMS, использующее несколько контейнеров-устройств (где каждый контейнер находится на отдельном диске), дает наибольшие возможности эффективной параллельной предварительной выборки. Значение параметра PREFETCHSIZE оператора CREATE TABLESPACE должно быть равно произведению значения параметра EXTENTSIZE и числа контейнеров-устройств. Это позволяет DB2 параллельно выполнять предварительную выборку изо всех контейнеров.

Если файловая система выполняет собственную предварительную выборку, разумная альтернатива для нагрузки запросов - использовать файлы. Файлы могут содержать как контейнеры типа DMS, так и контейнеры типа SMS. Имейте в виду, что, если вы используете SMS, для достижения параллельности ввода/вывода нужно, чтобы контейнеры каталогов отображались на разные физические диски.

Для смешанной нагрузки ваша задача - сделать как можно более эффективными одиночные требования ввода/вывода для нагрузки OLTP и повысить эффективность параллельного ввода/вывода для нагрузки запросов.

При определении размера страницы для рабочего пространства нужно учитывать следующие особенности:

- Для программ OLTP, выполняющих нерегулярные операции чтения и записи строк, обычно предпочтителен меньший размер страницы, чтобы тратить меньше места в пуле буферов на ненужные строки.
- При работе с программами DSS, обращающимися сразу к большому числу последовательных строк, обычно лучше использовать больший размер страницы, так как это уменьшает количество требований ввода/вывода, необходимых для считывания заданного числа строк. Но здесь есть одно исключение. Если ваш размер строки меньше, чем:

  ```
  размер_страницы / 255
  ```

  на всех страницах окажется неиспользуемое место (максимальное число строк на страницу - 255). В этой ситуации, возможно, лучше использовать меньший размер страницы.
- Большие размеры страниц позволяют уменьшить число уровней в индексе.
- Большие страницы поддерживают строки большей длины.
- На страницах по умолчанию размером 4 Кбайта в таблицах может быть максимум 500 столбцов, в то время как большие размеры страниц (8 Кбайт, 16 Кбайт и 32 Кбайта) поддерживают 1012 столбцов.
- Максимальный размер табличного пространства прямо пропорционален размеру страницы в этом табличном пространстве. Эти ограничения описаны в справочнике *SQL Reference*.

Выбор между табличными пространствами SMS и DMS

Выбирая, в табличное пространство какого типа поместить данные, надо принять во внимание следующее:

Достоинства табличного пространства SMS:

- Ресурсы выделяются системой только при необходимости.
- Для создания базы данных требуется меньше подготовительных усилий, так как вам не требуется определять контейнеры заранее.

Достоинства табличного пространства DMS:

- Табличное пространство можно расширять, добавляя контейнеры с помощью оператора ALTER TABLESPACE. Существующие данные автоматически перебалансируются по новому набору контейнеров, что сохраняет максимальную эффективность ввода/вывода.
- Таблицу можно разбить на несколько табличных пространств в зависимости от типа хранимых данных:
  - Данные длинных полей и больших объектов
  - Индексы
  - Обычные данные

Это бывает нужно из соображений производительности или увеличения объема данных, которые можно хранить в таблице. Например, в таблице можно хранить 64 Гбайта обычных данных, 64 Гбайта индексов и 2 Тбайта длинных данных. Если вы используете страницы по 8 Кбайт, табличные данные и данные индексов могут занять до 128 Гбайт. Если вы используете страницы по 16 Кбайт, они могут достигать 256 Гбайт. Если вы используете страницы по 32 Кбайта, табличные данные и данные индексов могут достигать 512 Гбайт.

- Если операционная система позволяет, можно управлять размещением данных на диске.
- Если все табличные данные находятся в одном табличном пространстве, его можно отбросить и переопределить с меньшими затратами, чем потребовалось бы для отбрасывания и переопределения таблицы.
- У хорошо настроенного набора табличных пространств DMS производительность в целом выше, чем у табличных пространств SMS.

Примечание: В Solaris и PTX (IBM NUMA-Q) для нагрузок, где важна производительность, настоятельно рекомендуются табличные пространства DMS с непосредственными устройствами.

Небольшие персональные базы данных обычно проще вести, используя табличные пространства SMS. С другой стороны, в больших, растущих базах данных табличные пространства SMS скорее всего будут использованы лишь как временные, а для всех таблиц будут выделены табличные пространства DMS, включающие по несколько контейнеров. Кроме того, вероятно, вы захотите хранить данные длинных полей и индексы в отдельных табличных пространствах.

Если вы выбрали табличные пространства DMS с контейнерами-устройствами, вам стоит настроить вашу среду и управлять ей. Дополнительную информацию смотрите в разделе ″Особенности производительности для устройств DMS″ книги *Руководство администратора: Производительность*.

Повышение производительности при размещении данных на устройствах RAID

В этом разделе описывается, как повысить производительность, если данные находятся на устройствах RAID (Redundant Array of Independent Disks - дублированный массив независимых дисков). Обычно для каждого табличного пространства, использующего устройство RAID, нужно:

- Определить для табличного пространства один контейнер (устройство RAID).
- Сделать EXTENTSIZE табличного пространства равным или кратным размеру полосы RAID.
- Убедитесь, что значение PREFETCHSIZE табличного пространства:
  - равно или кратно произведению размера полосы RAID и числа параллельных устройств RAID;

– кратно EXTENTSIZE.

- Чтобы разрешить для табличного пространства параллельный ввод/вывод, используйте переменную реестра DB2_PARALLEL_IO (описана ниже).
- Чтобы в табличном пространстве выравнивались границы экстентов, используйте переменную реестра DB2_STRIPED_CONTAINERS (описана ниже).

DB2_PARALLEL_IO

При чтении данных из контейнеров табличного пространства или записи в них, если в базе данных несколько контейнеров, DB2 может использовать параллельный ввод/вывод. Однако есть ситуации, когда имеет смысл включать параллельный ввод/вывод для табличных пространств с одним контейнером. Например, если контейнер создан на одном устройстве RAID, состоящем из нескольких физических дисков, имеет смысл выполнять параллельные требования чтения и записи.

Чтобы принудительно разрешить параллельный ввод/вывод для табличного пространства с одним контейнером, используйте переменную реестра DB2_PARALLEL_IO. Для этой переменной можно задать значение "*" (звездочка), что означает все табличные пространства, или список ID табличных пространств через запятую. Например:

```
db2set DB2_PARALLEL_IO=*        {вкл. паралл. ввод/вывод для всех табл.
                                        простр.}
db2set DB2_PARALLEL_IO=1,2,4,8  {вкл. паралл. ввод/вывод для табл.
                                        простр. 1, 2, 4 и 8}
```

Чтобы изменения вступили в силу, после задания переменной реестра нужно остановить DB2 (**db2stop**) и запустить снова (**db2start**).

DB2_STRIPED_CONTAINERS

В настоящее время при создании контейнера табличного пространства DMS (устройства или файла) в начало контейнера записывается одностраничная метка. Остальные страницы доступны DB2 для хранения данных и сгруппированы в блоки размером по одному экстенту.

При использовании в качестве контейнеров табличных пространств устройств RAID рекомендуется создавать табличное пространство с размером экстентов, равным или кратным размеру полосы RAID. Однако из-за одностраничной метки контейнера экстенты не будут совпадать с полосами RAID, и при требовании ввода/вывода может понадобиться доступ к большему числу физических дисков, чем было бы оптимально.

Контейнеры табличных пространств DMS теперь можно создавать так, чтобы метка хранилась в отдельном (полном) экстенте. Это позволяет избежать

описанной проблемы, но требует выделения лишнего экстента внутри контейнера. Чтобы создавать такие контейнеры, нужно задать для переменной реестра DB2 DB2_STRIPED_CONTAINERS значение ″ON″, а затем остановить и перезапустить экземпляр:

```
db2set DB2_STRIPED_CONTAINERS=ON
db2stop
db2start
```

При создании любого контейнера DMS (при помощи операторов CREATE TABLESPACE или ALTER TABLESPACE) метки будут занимать целый экстент. Уже существующие контейнеры не будут изменены.

Чтобы прекратить создавать контейнеры с этим атрибутом, сбросьте значение переменной, а затем остановите и перезапустите экземпляр:

```
db2set DB2_STRIPED_CONTAINERS=
db2stop
db2start
```

Центр управления и команда LIST TABLESPACE CONTAINERS не показывают, был ли контейнер создан как многополосный. Они показывают только метку ″файл″ или ″устройство″, в зависимости от того, как был создан контейнер. Чтобы проверить, был ли контейнер создан как многополосный, можно воспользоваться опцией /DTSF команды DB2DART, чтобы вывести данные о табличных пространствах и контейнерах, а затем посмотреть поле ″тип″ интересующего вас контейнера. При помощи API опроса контейнера (**sqlbftcq** и **sqlbtcq**) можно создать простую программу, выводящую тип контейнера.

Особенности проектирования объединенных баз данных

При проектировании базы данных объединения следует учитывать следующее:

- Требования к пространству
- Задание приоритетов сети.

Обычно данные, доступные из базы данных объединения, не хранятся в этой базе данных. В системном каталоге хранятся ссылки на таблицы и производные таблицы источника данных, а фактические данные находятся на источнике данных. По существу, для базы данных объединения может потребоваться меньше места, чем для обычной. Это общее правило может не действовать, если ваши запросы в связи с различиями системы сортировки или с отсутствием функции на источнике данных приходится выполнять локально. В этом случае таблицы материализуются DB2 для обработки.

Поскольку большинство данных системы объединения обычно находится на одном или нескольких источниках данных, находящихся в сети, возможно,

следует изменить ресурсы, назначенные DB2 и системе в сети. Если на системе DB2 выделить сети больше ресурсов, чем собственно менеджеру баз данных, можно повысить производительность.

Глава 9. Проектирование распределенных баз данных

В DB2 транзакции обычно называют *единицами работы*. Единица работы - это восстановимая последовательность операций процесса прикладной программы. Единица работы используется менеджером баз данных для обеспечения согласованного состояния базы данных. Все операции чтения и записи для базы данных выполняются внутри единицы работы.

Например, банковская транзакция может включать в себя перевод денег с накопительного счета на расходный счет. После того как прикладная программа вычтет величину перевода с накопительного счета, эти два счета будут в несогласованном состоянии до тех пор, пока эта же величина не будет добавлена к расходному счету. После выполнения *обоих* шагов достигается точка согласованности. Изменения могут быть приняты и стать доступными для других прикладных программ.

Единица работы начинается при передаче базе данных первого оператора SQL. Прикладная программа должна завершить эту единицу работы с помощью оператора COMMIT или ROLLBACK. Оператор COMMIT делает постоянными все изменения, сделанные в единице работы. Оператор ROLLBACK удаляет эти изменения из базы данных. Если прикладная программа завершается нормально, но не выполняет явно ни один из этих операторов, для единицы работы автоматически выполняется принятие. Если прикладная программа завершается аварийно в середине единицы работы, для этой единицы работы выполняется откат. Если операция COMMIT или ROLLBACK запущена, остановить ее нельзя. Для некоторых многопоточных прикладных программ и некоторых операционных систем (например, Windows), если прикладная программа завершается нормально, но не выполняет явно ни один из этих операторов, для единицы работы автоматически выполняется откат. Рекомендуется, чтобы прикладные программы всегда явно выполняли принятие или откат в конце единицы работы. Если часть единицы работы не выполнена успешно, для изменений выполняется откат и использовавшиеся таблицы остаются в том состоянии, в котором они были перед началом транзакции. Это гарантирует, что требования не будут потеряны или выполнены дважды.

Дополнительная информация содержится в следующих разделах:

Информацию о создании прикладных программ для работы с распределенными базами данных смотрите в руководствах *Application Development Guide* и *CLI Guide and Reference*.

Использование в транзакции одной базы данных

Самая простая форма транзакции выполняет чтение или запись только для одной базы данных в одной единице работы. Такой тип доступа к базе данных называется *удаленной единицей работы*.

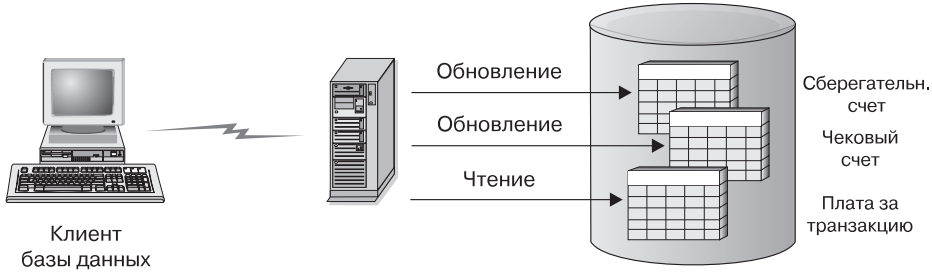


Рисунок 38. Использование в транзакции одной базы данных

На рис. 38 показан клиент базы данных, на котором выполняется прикладная программа денежных переводов; она обращается к базе данных, содержащей таблицы накопительных и расходных счетов, а также план оплаты банковских услуг. Эта программа должна:

- Получить значение переводимой суммы от пользовательского интерфейса
- Вычесть эту величину с накопительного счета и определить новый остаток
- Прочитать план оплаты, чтобы определить стоимость перевода денег для накопительного счета с данным остатком
- Вычесть стоимость перевода с накопительного счета
- Добавить переводимую сумму к расчетному счету
- Выполнить принятие транзакции (единицы работы).

Для конфигурирования такой программы надо:

1. Создать таблицы для накопительного счета, расчетного счета и плана оплаты в одной базе данных (смотрите раздел "Реализация проекта" в руководстве *Руководство администратора: Реализация*)
2. Для физически удаленной базы данных: задать для сервера баз данных использование подходящего протокола связи, как описано в книге *Дополнение по установке и настройке*
3. Для физически удаленной базы данных - внести в каталог узел и базу данных, чтобы определить эту базу для сервера, как описано в книгах *Quick Beginnings*.

4. Прекомпилировать прикладную программу, задав соединение типа 1; для этого надо задать CONNECT 1 (значение по умолчанию) в команде PRECOMPILE PROGRAM, как описано в книге *Application Development Guide*.

Использование в транзакции нескольких баз данных

При использовании в одной транзакции нескольких баз данных требования к настройке и управлению средой будут различными в зависимости от числа изменяемых в транзакции баз данных.

Изменение одной базы данных

Когда данные распределены по нескольким базам данных, может понадобиться читать данные из одной или нескольких баз данных и изменять данные в одной базе данных. Этот тип доступа, называемый *многоузловым изменением* или *двухфазным принятием*, может выполняться в одной единице работы (транзакции). Другой пример многоузлового изменения смотрите в разделе “Изменение нескольких баз данных” на стр. 172.

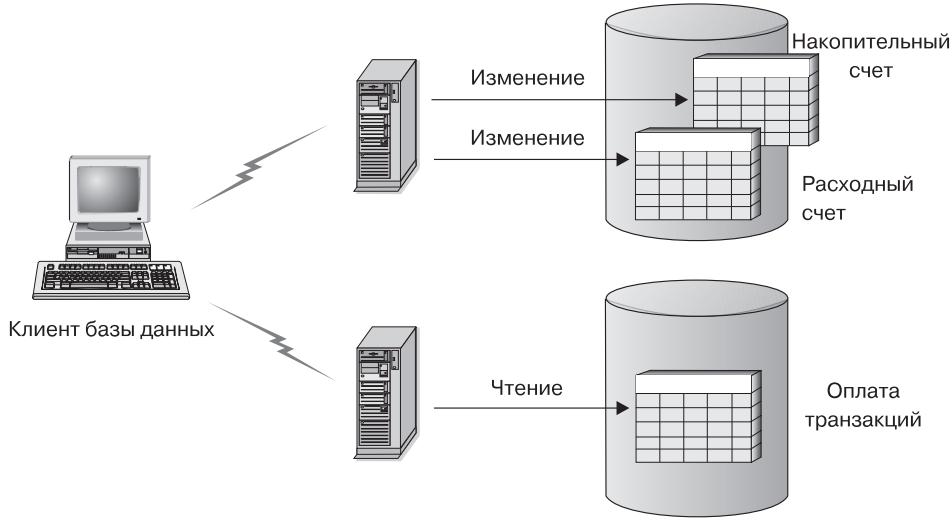


Рисунок 39. Использование в транзакции нескольких баз данных

На рис. 39 показан клиент базы данных, на котором выполняется прикладная программа денежных переводов; она обращается к двум серверам баз данных: на одном хранятся накопительные и расходные счета, а на другом - план оплаты банковских услуг. Этот пример подобен примеру на рис. 38 на стр. 170, но баз данных теперь несколько и таблицы размещены по-другому.

Чтобы сконфигурировать прикладную программу денежных переводов для этой среды, нужно:

1. Создать необходимые таблицы в соответствующих базах данных (смотрите раздел "Реализация вашего проекта" руководства *Руководство администратора: Реализация*)
2. Для физически удаленных баз данных: задать для серверов баз данных использование подходящих протоколов связи, как описано в книге *Дополнение по установке и настройке*
3. Для физически удаленных баз данных: внести в каталог узлы и базы данных, чтобы определить эти базы для серверов баз данных, как описано в книгах *Quick Beginnings*
4. Прекомпилировать прикладную программу, задав соединение типа 2 (для этого надо задать CONNECT 2 в команде PRECOMPILE PROGRAM) и однофазное принятие (для этого надо задать SYNCPOINT ONEPHASE в команде PRECOMPILE PROGRAM), как описано в книге *Application Development Guide*.

Если база данных находится на сервере баз данных хоста или AS/400, для связи с этим сервером требуется DB2 Connect. Информацию об установке DB2 Connect смотрите в одной из книг *Quick Beginnings* по DB2 Connect. Информацию об использовании DB2 Connect смотрите в книге *DB2 Connect. Руководство пользователя*.

Изменение нескольких баз данных

Когда данные распределены по нескольким базам данных, может понадобиться читать и изменять данные в нескольких базах данных в одной транзакции. Такой тип доступа к базе данных называется *многоузловым изменением*.

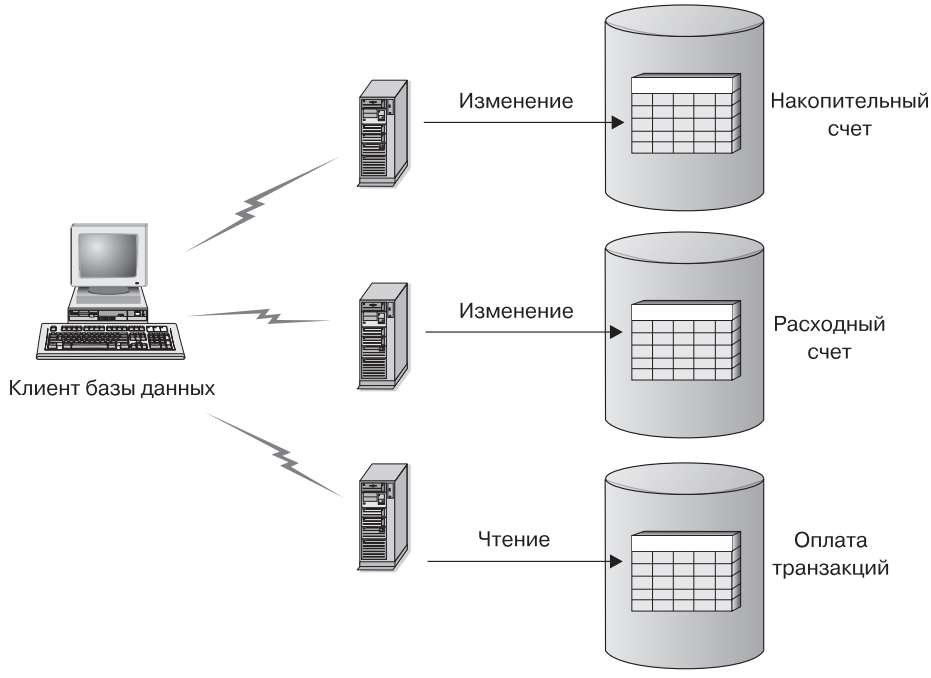


Рисунок 40. Изменение нескольких баз данных в одной транзакции

На рис. 40 показан клиент базы данных, на котором выполняется прикладная программа денежных переводов; она обращается к трем серверам баз данных: на одном хранятся накопительные счета, на другом - расходные счета, а на третьем - план оплаты банковских услуг.

Чтобы сконфигурировать прикладную программу денежных переводов для этой среды, нужно:

1. Создать необходимые таблицы в соответствующих базах данных (смотрите раздел ″Реализация вашего проекта″ руководства *Руководство администратора: Реализация*)
2. Для физически удаленных баз данных: задать для серверов баз данных использование подходящих протоколов связи, как описано в книге *Дополнение по установке и настройке*
3. Для физически удаленных баз данных: внести в каталог узлы и базы данных, чтобы определить эти базы для серверов баз данных, как описано в книгах *Quick Beginnings*
4. Прекомпилировать прикладную программу, задав соединение типа 2 (для этого надо задать CONNECT 2 в команде PRECOMPILE PROGRAM) и двухфазное принятие (для этого надо задать SYNCPOINT TWOPHASE в команде PRECOMPILE PROGRAM), как описано в книге *Application Development Guide*.

5. Сконфигурировать менеджер транзакций DB2, как описано в разделе “Использование менеджера транзакций DB2”.

Использование менеджера транзакций DB2

Менеджер баз данных поддерживает функции менеджера транзакций, которые могут использоваться для координации изменений нескольких баз данных в одной единице работы. Клиент базы данных автоматически координирует единицу работы и использует *базу данных менеджера транзакций* для регистрации каждой транзакции и отслеживания состояния ее выполнения.

Если используется менеджер транзакций, совместимый с XA (например, IBM TXSeries, BEA Tuxedo или Microsoft Transaction Server), инструкции по его интеграции в систему смотрите в разделе “Глава 10. Планирование использования менеджеров транзакций” на стр. 187.

Если для координации транзакций используется DB2 UDB для систем на основе UNIX, операционных систем Windows или OS/2, должны быть удовлетворены некоторые требования к конфигурации. Если для связи используется только протокол TCP/IP и в транзакциях участвуют только серверы баз данных DB2 UDB OS/390 и DB2 for OS/390, конфигурирование будет несложным.

DB2 UDB for OS/390 и DB2 for OS/390 при использовании связи TCP/IP: Если в вашей среде выполняются следующие условия, конфигурирование многоузлового изменения будет несложным.

- Для связи со всеми удаленными серверами баз данных (включая DB2 UDB for OS/390) используется только TCP/IP.
- В транзакции используются только серверы баз данных DB2 UDB для систем на основе UNIX, операционных систем Windows, OS/2 или OS/390.
- Не сконфигурирован менеджер точек синхронизации (SPM) DB2 Connect.

  Менеджер точек синхронизации DB2 Connect конфигурируется автоматически во время создания экземпляра DB2; он требуется, если:

  – Для операций многоузлового изменения используется связь SNA с серверами баз данных хоста или AS/400.

  – Двухфазное принятие координируется менеджером транзакций, совместимым с XA (например, IBM TXSeries CICS).

    Это верно как для связи SNA, так и для связи TCP/IP с серверами баз данных хоста или AS/400. Подробную информацию смотрите в разделе “Глава 10. Планирование использования менеджеров транзакций” на стр. 187. Если вашей среде не требуется менеджер точек синхронизации DB2 Connect, его можно отключить, выполнив команду `db2 update dbm cfg using spm_name NULL` на сервере DB2 Connect. Затем остановите и перезапустите DB2.

Базу данных, которая будет использоваться в качестве базы данных менеджера транзакций, задает на клиенте базы данных параметр конфигурации *tm_database*. Дополнительную информацию об этом параметре конфигурации смотрите в разделе ″Конфигурирование DB2″ руководства *Руководство администратора: Производительность*. При задании этого параметра конфигурации учтите следующие факторы:

- Менеджер транзакций может быть:
  - Базой данных DB2 UDB для систем на основе UNIX, операционных систем Windows или OS/2
  - Базой данных DB2 for OS/390 Версии 5 или более новой.

    В качестве базы данных менеджера транзакций рекомендуется использовать именно этот сервер баз данных. Системы OS/390 в целом более надежны, чем серверы рабочих станций, и на них меньше вероятность случайных отключений питания, перезагрузок и так далее. Поэтому будут более надежными и журналы восстановления, используемые при ресинхронизации событий.
- Если для параметра конфигурации *tm_database* задано значение `1ST_CONN`, в качестве базы данных менеджера транзакций будет использоваться первая база данных, с которой соединится прикладная программа.

  При использовании значения `1ST_CONN` необходима осторожность. Используйте его, только если легко обеспечить правильное внесение в каталог всех используемых баз данных; то есть если:
  - Клиент базы данных, запускающий транзакцию, принадлежит тому же экземпляру, что и участвующие в транзакции базы данных, включая базу данных менеджера транзакций.
  - Для каталогизации и управления доступом к базам данных используются службы каталогов DCE.

  Учтите, что, если прикладная программа попытается отсоединиться от базы данных, используемой в качестве базы данных менеджера транзакций, вы получите предупреждение, а соединение будет удерживаться до принятия единицы работы.

Другие среды: Если в вашей среде:

- Для связи с удаленными серверами баз данных используется не только TCP/IP (например, NETBIOS)
- Выполняются обращения к DB2 for MVS Версии 3 или Версии 4, DB2 for AS/400 или DB2 for VM&VSE
- Выполняются обращения к DB2 for OS/390 с использованием SNA
- Для доступа к серверам баз данных хоста или AS/400 используется менеджер точек синхронизации DB2 Connect

шаги конфигурирования многоузлового изменения будут сложнее.

База данных, которая будет использоваться в качестве базы данных менеджера транзакций, задается на клиенте базы данных параметром конфигурации *tm_database*. Дополнительную информацию об этом параметре конфигурации смотрите в разделе "Конфигурирование DB2" руководства *Руководство администратора: Производительность*. При задании этого параметра конфигурации учтите следующие факторы:

- Базой данных менеджера транзакций может быть база данных DB2 UDB для систем на основе UNIX, операционных систем Windows или OS/2.
- Если для параметра конфигурации *tm_database* задано значение `1ST_CONN`, в качестве базы данных менеджера транзакций будет использоваться первая база данных, с которой соединится прикладная программа.

  При использовании значения `1ST_CONN` необходима осторожность. Используйте его, только если легко обеспечить правильное внесение в каталог всех используемых баз данных; то есть если:

  – Клиент базы данных, запускающий транзакцию, принадлежит тому же экземпляру, что и участвующие в транзакции базы данных, включая базу данных менеджера транзакций.
  – Для каталогизации и управления доступом к базам данных используются службы каталогов DCE.

  Учтите, что, если прикладная программа попытается отсоединиться от базы данных, используемой в качестве базы данных менеджера транзакций, вы получите предупреждение, а соединение будет удерживаться до принятия единицы работы.

Другие особенности конфигурирования

При конфигурировании среды надо рассмотреть следующие параметры конфигурации. Дополнительную информацию о задании этих параметров смотрите в руководстве *DB2 Connect. Руководство пользователя*.

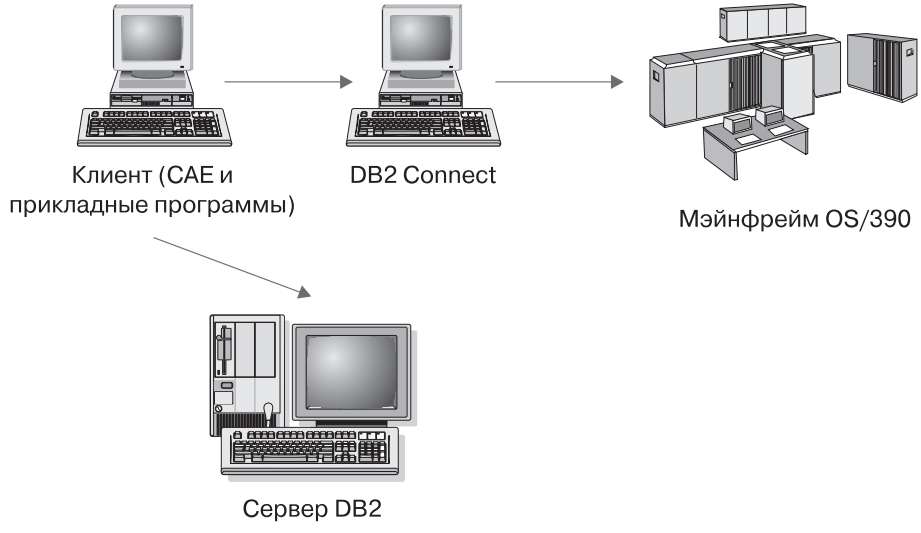


Рисунок 41. Особенности конфигурирования

Параметры конфигурации менеджера баз данных

- *tm_database*

  Этот параметр задает имя базы данных Transaction Manager (TM) для каждого экземпляра DB2.

- *spm_name*

  Этот параметр задает имя экземпляра менеджера точек синхронизации DB2 Connect для менеджера баз данных. Для успешной ресинхронизации это имя должно быть уникальным в сети.

- *resync_interval*

  Этот параметр задает интервал (в секундах), после которого менеджер транзакций DB2, менеджер баз данных сервера DB2 и менеджер точек синхронизации DB2 Connect или менеджер точек синхронизации DB2 UDB должны попытаться восстановить все незавершенные неоднозначные транзакции.

- *spm_log_file_sz*

  Этот параметр задает размер (в 4-Кбайтных страницах) файла журнала SPM.

- *spm_max_resync*

  Этот параметр задает число агентов, которые могут одновременно выполнять операции ресинхронизации.

- *spm_log_path*

  Этот параметр задает путь журналов для файлов журналов SPM.

Параметры конфигурации базы данных

- *maxappls*

  Этот параметр задает максимальное разрешенное число активных прикладных программ. Его значение должно быть больше или равно числу соединенных прикладных программ, плюс число таких прикладных программ, для которых может одновременно выполняться двухфазное принятие или откат, плюс ожидаемое число возможных неоднозначных транзакций в любой момент времени. Дополнительную информацию о неоднозначных транзакциях смотрите в разделе “Восстановление после ошибок двухфазного принятия” на стр. 183.

- *autorestart*

  Этот параметр конфигурации определяет, будет ли при необходимости автоматически выполняться процедура перезапуска базы данных (RESTART DATABASE). Значение по умолчанию - YES (перезапуск разрешен).

  Чтобы можно было использовать базу данных, содержащую неоднозначные транзакции, для нее должна быть выполнена операция перезапуска. Если параметр *autorestart* имеет значение NO, после прекращения последнего соединения с базой данных следующее соединение будет неудачным и потребуется явно запустить команду RESTART DATABASE. Это условие сохраняется, пока неоднозначные транзакции не будут удалены операцией ресинхронизации менеджера транзакций или запущенной администратором эвристической операцией. Если при запуске команды RESTART DATABASE в базе данных есть неоднозначные транзакции, выдается сообщение. Администратор может затем использовать команду LIST INDOUBT TRANSACTIONS и другие команды процессора командной строки, чтобы получить информацию об этих неоднозначных транзакциях.

Дополнительную информацию об этих параметрах конфигурации смотрите в руководстве *Руководство администратора: Производительность*.

Прикладные программы хоста или AS/400, обращающиеся к серверу DB2 Universal Database в локальной сети при многоузловом изменении

DB2 Universal Database не поддерживает многоузловое изменение с клиентов баз данных хоста или AS/400 с использованием связи TCP/IP. В этой ситуации поддерживается только связь SNA (Systems Network Architecture). Для многоузлового изменения требуется менеджер точек синхронизации DB2. DB2 Connect при этом не используется.

Сервер баз данных, к которому обращаются с клиента баз данных хоста или AS/400, не обязан быть локальным для рабочей станции, на которой установлен менеджер точек синхронизации DB2. Клиент баз данных хоста или AS/400 может соединяться с сервером DB2 UDB, используя рабочую станцию менеджера точек синхронизации DB2 в качестве временного шлюза. Это позволяет установить рабочую станцию менеджера точек синхронизации DB2 в защищенной среде, в то время как сами серверы DB2 UDB будут удаленными.

Это также позволяет использовать базу данных DB2 Common Server Версии 2 в многоузловом изменении с клиентов баз данных хоста или AS/400.

Шаги конфигурирования:

- На рабочей станции, к которой будут напрямую обращаться прикладные программы хоста или AS/400:
  1. Для поддержки многоузлового изменения с клиентов баз данных хоста или AS/400 установите DB2 Universal Database Enterprise Edition или Enterprise - Extended Edition.
  2. Создайте экземпляр базы данных на той же системе. Можно, например, использовать экземпляр по умолчанию, DB2, или создать новый экземпляр с помощью команды:

     ```
     db2icrt мой_экземпляр
     ```
  3. Задайте требуемую лицензионную информацию.
  4. Убедитесь, что в значении реестра DB2COMM указано APPC.
  5. Правильно сконфигурируйте связь SNA. Если используются поддерживаемые IBM продукты SNA, необходимые для менеджера точек синхронизации DB2 профили SNA создаются автоматически на основе значения параметра конфигурации менеджера баз данных *spm_name*. Для других поддерживаемых стеков SNA требуется ручная настройка. Подробную информацию смотрите в руководстве *Дополнение по установке и настройке*.
  6. Определите значение, которое нужно задать для параметра конфигурации менеджера баз данных *spm_name*. При создании экземпляра DB2 для этого параметра задается значение, полученное на основе имени хоста TCP/IP для этого компьютера. Если это значение подходит и уникально в вашей среде, не меняйте его.
  7. При необходимости измените значение параметра *spm_name* на сервере DB2 Universal Database, используя команду UPDATE DATABASE MANAGER CONFIGURATION.
  8. Сконфигурируйте связь, чтобы эта рабочая станция DB2 могла в случае необходимости соединяться с удаленными серверами DB2 UDB.
  9. Сконфигурируйте связь, чтобы удаленные серверы DB2 UDB могли соединяться с этим сервером DB2.
  10. Чтобы в первый раз запустить SPM, остановите и снова запустите менеджер баз данных на сервере DB2 Universal Database.

      Необходимо, чтобы с этой рабочей станции DB2 UDB можно было соединяться с удаленными серверами DB2 UDB.
- На каждом удаленном сервере DB2 UDB, к которому будет обращаться клиент базы данных хоста или AS/400:

1. Сконфигурируйте связь, чтобы рабочая станция удаленного сервера DB2 UDB с менеджером точек синхронизации DB2 могла соединяться с этим сервером DB2 UDB.
2. Остановите и снова запустите менеджер баз данных.

Процесс двухфазного принятия

На рис. 42 на стр. 181 показаны шаги многоузлового изменения. Понимание того, как выполняется управление транзакциями, поможет вам исправлять ошибки, возникшие в процессе двухфазного принятия.

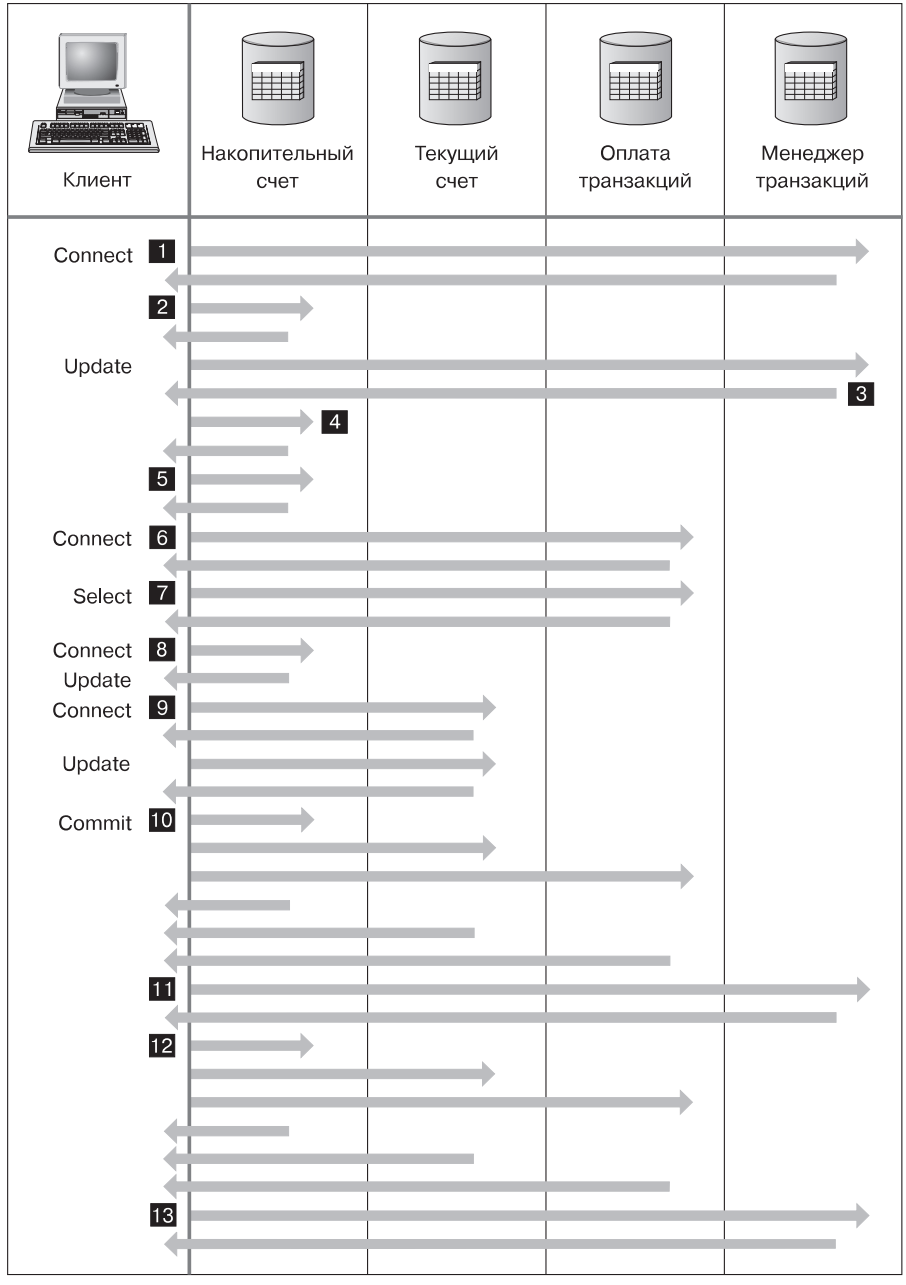


Рисунок 42. Изменение нескольких баз данных

0 Прикладная программа подготавливается для двухфазного принятия. Для этого могут использоваться опции прекомпиляции (подробную информацию смотрите в руководстве *Application Development Guide*).

Для этого также может использоваться конфигурация CLI (интерфейс уровня вызовов) DB2 (подробную информацию смотрите в руководстве *CLI Guide and Reference*).

1 Когда клиенту базы данных нужно соединиться с базой данных SAVINGS_DB, он сначала соединяется с базой данных менеджера транзакций. База данных менеджера транзакций возвращает подтверждение клиенту баз данных. Если параметр конфигурации менеджера баз данных *tm_database* имеет значение `1ST_CONN`, на время работы этого экземпляра прикладной программы в качестве SAVINGS_DB используется база данных менеджера транзакций.

2 Установлено соединение с базой данных SAVINGS_DB и получено подтверждение.

3 Клиент базы данных начал изменять таблицу SAVINGS_ACCOUNT. Это начинает единицу работы. База данных менеджера транзакций отвечает клиенту базы данных, сообщая ID транзакции для этой единицы работы. Обратите внимание на то, что регистрация единицы работы происходит при выполнении первого оператора SQL в рабочей единице, а не при установлении соединения.

4 Получив ID транзакции, клиент базы данных регистрирует эту единицу работы в базе данных, содержащей таблицу SAVINGS_ACCOUNT. Клиенту посылается ответ, сообщающий об успешной регистрации единицы работы.

5 Выдается оператор SQL для базы данных SAVINGS_DB; он обрабатывается обычным образом. При работе с встроенными в программу операторами SQL ответ для каждого оператора возвращается в SQLCA. (Описание SQLCA смотрите в руководстве *Application Development Guide* и справочнике *SQL Reference*.)

6 ID транзакции регистрируется в базе данных FEE_DB, содержащей таблицу TRANSACTION_FEE, при первом обращении к этой базе данных в данной единице работы.

7 Все операторы SQL для базы данных FEE_DB обрабатываются обычным образом.

8 При необходимости при установлении соединения для базы данных SAVINGS_DB могут выполняться дополнительные операторы SQL. Поскольку единица работы уже зарегистрирована в базе данных SAVINGS_DB **4** , клиенту базы данных не нужно опять выполнять шаг регистрации.

9 Для соединения с базой данных CHECKING_DB и ее использования применяются правила, описанные в **6** и **7** .

10 Когда клиент базы данных запрашивает принятие единицы работы, всем базам данных, принимающим участие в этой транзакции,

посылается сообщение *prepare*. Каждая база данных помещает запись ″PREPARED″ в свой файл журнала и отвечает клиенту базы данных.

11 Получив положительные ответы от всех баз данных, клиент базы данных посылает сообщение базе данных менеджера транзакций, информируя ее, что единица работы теперь готова к выполнению принятия (PREPARED). База данных менеджера транзакций помещает запись ″PREPARED″ в свой файл журнала и отвечает клиенту, сообщая ему, что может быть начата вторая фаза процесса принятия.

12 Во время второй фазы процесса принятия клиент базы данных посылает сообщение всем базам данных, принимающим участие в этой транзакции, указывая им, что нужно выполнить принятие. Каждая база данных помещает запись ″COMMITTED″ в свой файл журнала и освобождает все блокировки, удерживаемые для этой единицы работы. Завершив принятие изменений, база данных посылает ответ клиенту.

13 Получив положительные ответы от всех баз данных, принимающих участие в этой транзакции, клиент базы данных посылает сообщение базе данных менеджера транзакций, информируя ее, что единица работы завершена. Затем база данных менеджера транзакций помещает запись ″COMMITTED″ в свой файл журнала, указывающую, что единица работы завершена, и отвечает клиенту, сообщая ему, что операция завершена.

Восстановление после ошибок двухфазного принятия

Восстановление после ошибок - это обычная задача, связанная с созданием прикладных программ, управлением системой, управлением базой данных и управлением системой. При распределении баз данных по нескольким удаленным серверам возрастает вероятность ошибок, вызванных ошибками сети или ошибками связи. Для обеспечения целостности данных менеджер баз данных использует двухфазный процесс принятия, описанный в разделе “Процесс двухфазного принятия” на стр. 180 (точки **10** , **11** и **12**). Далее объясняется, как менеджер баз данных обрабатывает ошибки, возникшие в процессе двухфазного принятия:

- **Ошибка на первой фазе**

  Если база данных сообщает, что она не смогла подготовиться к принятию единицы работы, на второй фазе принятия клиент базы данных выполняет откат этой единицы работы. В этом случае базе данных менеджера транзакций *не* посылается сообщение о подготовке.

  На второй фазе клиент посылает сообщение об откате всем принимающим участие в транзакции базам данных, которые успешно подготовили принятие на первой фазе. Затем каждая база данных помещает запись ″ABORT″ в свой файл журнала и освобождает все блокировки, удерживаемые для этой единицы работы.

- **Ошибка на второй фазе**

  Обработка ошибки на этом этапе зависит от операции, выполняемой для транзакции на второй фазе (принятие или откат). Если на первой фазе возникла ошибка, на второй фазе для транзакции будет выполняться откат.

  Если одной из принимающих участие в транзакции баз данных не удалось выполнить принятие единицы работы (возможно, из-за ошибки связи), база данных менеджера транзакций попытается еще раз выполнить принятие в базе данных, в которой возникла ошибка. Однако прикладной программе в SQLCA будет передана информация о том, что принятие было успешным. DB2 обеспечит принятие непринятых транзакций на сервере баз данных. Для задания срока ожидания между попытками базы данных менеджера транзакций выполнить принятие единицы работы используется параметр конфигурации менеджера баз данных *resync_interval* (смотрите раздел ″Конфигурирование DB2″ руководства *Руководство администратора: Производительность*). Сервер баз данных удерживает все блокировки, пока единица работы не будет принята.

  Если возникла ошибка базы данных менеджера транзакций, она ресинхронизирует единицу работы при своем перезапуске. Процесс ресинхронизации попытается завершить все *неоднозначные транзакции*, то есть транзакции, для которых была завершена первая, но не завершена вторая фаза процесса принятия. Для выполнения ресинхронизации менеджер баз данных, связанный с базой данных менеджера транзакций, выполняет следующие операции:

  1. Соединяется с базами данных, которые были подготовлены к принятию (″PREPARED″) на первой фазе процесса принятия.
  2. Пытается выполнить принятие неоднозначных транзакций в этих базах данных. (Если неоднозначные транзакции не удалось найти, менеджер баз данных предполагает, что база данных успешно выполнила принятие транзакций на второй фазе процесса принятия.)
  3. Выполняет принятие неоднозначных транзакций в базе данных менеджера транзакций после того, как все неоднозначные транзакции были приняты в базах данных, принимающих участие в транзакции.

  Если в одной из принимающих участие в транзакции баз данных возникла ошибка и эта база данных была перезапущена, менеджер баз данных для этой базы данных запросит у базы данных менеджера транзакций информацию о состоянии этой транзакции, чтобы узнать, нужно ли выполнить откат этой транзакции. Если транзакция не найдена в журнале, менеджер баз данных предполагает, что для транзакции был выполнен откат, и выполняет откат этой неоднозначной транзакции в этой базе данных. В противном случае база данных ожидает запроса принятия от базы данных менеджера транзакций.

Если транзакция координируется монитором обработки транзакций (менеджером транзакций, совместимым с XA), база данных будет всегда зависеть от монитора обработки транзакций, инициализирующего ресинхронизацию.

Если по каким-то причинам вы не можете ждать, пока менеджер транзакций автоматически завершит неоднозначные транзакции, это можно сделать вручную. Этот процесс восстановления вручную иногда называют ″принятием эвристического решения″. Дополнительную информацию о ручном восстановлении неоднозначных транзакций смотрите в разделе “Принятие эвристических решений” на стр. 199.

Ресинхронизация неоднозначных транзакций, если AUTORESTART=OFF

Информацию об особенностях конфигурирования в среде DB2 Universal Database с двухфазным принятием смотрите в разделе “Другие особенности конфигурирования” на стр. 176.

В частности, если параметр конфигурации базы данных *autorestart* имеет значение OFF и в базах данных менеджера транзакций или менеджера восстановления есть неоднозначные транзакции, для запуска процесса ресинхронизации требуется команда RESTART DATABASE. Для ввода команды RESTART DATABASE из процессора командной строки используйте разные сеансы. Если из того же сеанса перезапустить другую базу данных, соединение, установленное предыдущей командой, будет прекращено и его придется перезапустить еще раз. Когда команда LIST INDOUBT TRANSACTIONS более не сообщает о неоднозначных транзакциях, завершите соединение с помощью команды TERMINATE.

Глава 10. Планирование использования менеджеров транзакций

Если кроме баз данных DB2 у вас есть и другие ресурсы, которые вы хотите использовать в двухфазном принятии транзакций, можно использовать для баз данных XA-совместимый менеджер транзакций. Если в транзакциях принимают участие только базы данных DB2, надо использовать менеджер транзакций DB2, описанный в разделе “Изменение нескольких баз данных” на стр. 172.

Следующие темы помогут вам освоить использование менеджера баз данных с такими XA-совместимыми менеджерами транзакций, как IBM TXSeries CICS, IBM TXSeries Encina, BEA Tuxedo и Microsoft Transaction Server:

Если вы используете или собираетесь использовать XA-совместимый менеджер транзакций, дополнительную информацию вы сможете найти на Web-сайте технической поддержки IBM:

```
http://www.ibm.com/software/data/db2/library/
```

На этой странице выберите ″DB2 Universal Database″, а затем выполните поиск по ключевому слову ″XA″, чтобы найти самую свежую информацию о XA-совместимых менеджерах транзакций.

Модель распределенной обработки транзакций X/Open

Модель распределенной обработки транзакций X/Open (Distributed Transaction Processing - DTP) включает в себя три взаимосвязанных компонента:

- “Прикладная программа”
- “Менеджер транзакций” на стр. 190
- “Менеджеры ресурсов” на стр. 191.

Эта модель и отношения между ее компонентами показаны на рис. 43.

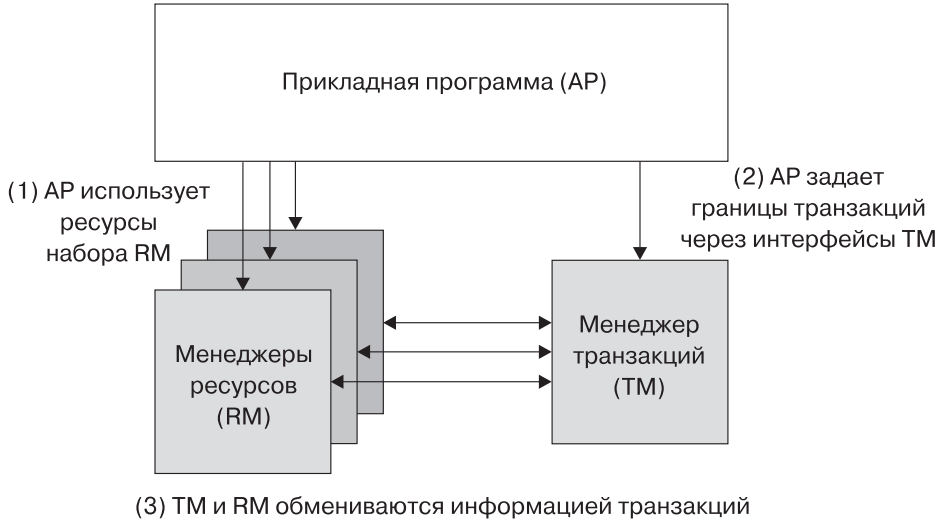


Рисунок 43. Модель распределенной обработки транзакций X/Open (DTP)

Прикладная программа

Прикладная программа определяет границы транзакции и входящие в состав транзакции действия, требуемые для выполнения этой прикладной программы.

Например, прикладная программа CICS* может обращаться к менеджерам ресурсов, таким, как база данных и очередь временных данных CICS, и обрабатывать данные при помощи своей внутренней логики. Каждое требование доступа передается соответствующим менеджерам ресурсов при помощи вызовов функций, специфичных для конкретных менеджеров. В DB2 это могут быть вызовы функций, генерируемые для каждого оператора SQL прекомпилятором DB2, или вызовы баз данных, кодируемые непосредственно программистом с использованием API.

Менеджер транзакций (TM) обычно включает в себя монитор транзакций (TP) для запуска пользовательских программ. Монитор транзакций предоставляет

API, позволяющие программе начать и закончить транзакцию, а также включить программу в расписание и сбалансировать нагрузку для большого числа пользователей, которые запускают программу. Реально прикладная программа в среде распределенной обработки транзакций - это сочетание пользовательской программы и монитора транзакций.

Для упрощения работы среды оперативной обработки транзакций (OLTP) монитор транзакций размещает несколько процессов сервера при запуске, а затем планирует и использует их многократно во многих пользовательских транзакциях. Это экономит системные ресурсы, позволяя меньшему числу процессов серверов и соответствующих им процессов менеджера ресурсов поддерживать большее число одновременно работающих пользователей. Кроме того, многократное использование процессов позволяет избежать расходов, связанных с запуском процесса в менеджере транзакций и в менеджерах ресурсов для каждой пользовательской транзакции или программы. (Программа вызывает одну или несколько транзакций.) Это означает также, что для менеджера транзакций и менеджеров ресурсов процессы серверов являются ″пользовательскими процессами″. Это имеет значение для управления защитой и для написания программ. Подробности смотрите в разделе “Особенности защиты” на стр. 202.

Для монитора транзакций возможны следующие типы транзакций:

- Транзакции без XA

  Эти транзакции используют менеджеры ресурсов, которые не определены для менеджера транзакций и поэтому не координируются протоколом двухфазного принятия TM. Это необходимо, если программа должна обратиться к менеджеру ресурсов, который не поддерживает интерфейс XA. Монитор транзакций в этом случае просто обеспечивает расписание программ и балансировку нагрузки. Поскольку менеджер транзакций не ″открывает″ явно менеджеры ресурсов для обработки XA, менеджер ресурсов рассматривает эту программу, как и любую другую программу, выполняемую не в среде DTP.

- Глобальные транзакции

  Эти транзакции используют менеджеры ресурсов, которые определены для менеджера транзакций и координируются протоколом двухфазного принятия TM. Глобальная транзакция - это единица работы, которая может вовлекать один или несколько менеджеров ресурсов. Часть единицы работы между менеджером транзакций и менеджером ресурсов, поддерживающим глобальную транзакцию, называется *ветвью транзакции*. У глобальной транзакции может быть несколько ветвей транзакции, если один или несколько процессов программ, координируемых TM, обращаются к нескольким менеджерам ресурсов.

  В свободно связанных глобальных транзакциях каждый из процессов прикладной программы обращается к менеджерам ресурсов, как если бы он был в отдельной глобальной транзакции, хотя эти программы и

координируются менеджером транзакций. В менеджере ресурсов у каждого процесса прикладной программы будет своя собственная ветвь транзакции. Когда прикладная программа, менеджер транзакций или менеджер ресурсов требуют принятия или отката, ветви транзакции полностью завершаются. Ответственность за тупиковые ситуации между ветвями транзакции несет программа. (Заметим, что координация транзакций, выполняемая менеджером транзакций DB2 для программ, подготовленных с опцией SYNCPOINT(TWOPHASE), примерно эквивалентна таким свободно связанным глобальным транзакциям. Смотрите раздел “Изменение нескольких баз данных” на стр. 172.)

В тесно связанной глобальной транзакции несколько процессов прикладной программы запускаются по очереди под одной ветвью транзакции в менеджере ресурсов. Для менеджера ресурсов эти процессы прикладной программы являются одним объектом. Ответственность за тупиковые ситуации в этой ветви транзакции несет менеджер ресурсов.

Менеджер транзакций

Менеджер транзакций приписывает транзакциям идентификаторы, отслеживает ход транзакций и несет ответственность за их успешное или неудачное завершение. Идентификаторы ветвей транзакций (XID) назначаются менеджером транзакций для идентификации как глобальных транзакций, так и их отдельных ветвей в менеджере ресурсов. Они используются как метки соответствия между журналом менеджера транзакций и журналом менеджера ресурсов. XID используются в двухфазном принятии или откате для выполнения операции *ресинхронизации* при запуске системы, или чтобы в случае необходимости администратор мог выполнить *эвристическую* операцию (*ручное вмешательство*).

После запуска монитора транзакций он просит менеджер транзакций открыть все менеджеры ресурсов, определенные набором серверов прикладных программ. Менеджер транзакций передает менеджерам ресурсов вызовы **xa_open**, чтобы инициировать их для обработки DTP. В ходе процедуры запуска менеджер транзакций выполняет ресинхронизацию, чтобы восстановить *неоднозначные транзакции*. Неоднозначная транзакция - это глобальная транзакция, которая осталась в неопределенном состоянии. Такое случается, когда менеджер транзакций (или как минимум один менеджер ресурсов) стал недоступным после успешного завершения первой фазы (фазы подготовки) протокола двухфазного принятия. Менеджер ресурсов не будет знать, выполнять ему принятие или откат своей ветви транзакции, пока менеджер транзакций не согласует свой журнал с журналами менеджеров ресурсов, когда они снова станут доступными. Для выполнения ресинхронизации менеджер транзакций один или несколько раз передает вызов **xa_recover** каждому менеджеру ресурсов, чтобы идентифицировать все неоднозначные транзакции. Менеджер транзакций сравнивает ответы с соответствующей информацией в своем журнале и определяет, какой вызов надо послать менеджерам ресурсов: **xa_commit** или **xa_rollback**. Если какой-то менеджер ресурсов для своей ветви

неоднозначной транзакции уже выполнил принятие или откат в результате эвристической операции своего администратора, менеджер транзакций для ресинхронизации посылает этому менеджеру ресурсов вызов **xa_forget**.

Когда программа пользователя требует принятия или отката, она должна использовать API, предоставляемые монитором транзакций или менеджером транзакций, чтобы менеджер транзакций мог координировать принятия и откаты по всем вовлеченным менеджерам ресурсов. Например, если программа CICS выдает для принятия транзакции требование CICS SYNCPOINT, менеджер транзакций CICS XA (реализованный в сервере Encina Server) в свою очередь выдаст вызовы XA **xa_end**, **xa_prepare**, **xa_commit** или **xa_rollback** в качестве требования менеджеру ресурсов выполнить принятие или откат данной транзакции. Если в транзакции участвует только один менеджер ресурсов или менеджер ресурсов отвечает, что в его ветви транзакции выполнялось только чтение, менеджер транзакций вместо двухфазного принятия может использовать однофазное.

Менеджеры ресурсов

Менеджер ресурсов обеспечивает доступ к совместно используемым ресурсам, например, к базам данных.

DB2, как менеджер ресурсов базы данных, может участвовать в *глобальной транзакции*, которая координируется XA-совместимым менеджером транзакций. Как того требует интерфейс XA, менеджер баз данных обеспечивает внешнюю переменную *db2xa_switch* языка C типа xa_switch_t для возврата менеджеру транзакций структуры переключателя XA. Эта структура данных содержит адреса различных подпрограмм XA, вызываемых менеджером транзакций, и операционные характеристики менеджера ресурсов. Дополнительную информацию о функциях XA, поддерживаемых менеджером баз данных, смотрите в разделе “Поддерживаемая функция XA” на стр. 204.

Есть два метода регистрации участия менеджера ресурсов в каждой глобальной транзакции: *статическая регистрация* и *динамическая регистрация*.

- При статической регистрации требуется, чтобы менеджер транзакций выдавал (для каждой транзакции) серии вызовов **xa_start**, **xa_end** и **xa_prepare** всем менеджерам ресурсов, определенным для программы сервера, независимо от того, используется ли данный менеджер ресурсов транзакцией. Если не все менеджеры ресурсов участвуют в каждой транзакции, это неэффективно, причем, чем больше определено менеджеров транзакций, тем сильнее эта неэффективность.
- Динамическая регистрация (используемая DB2) гибче и эффективней. Менеджер ресурсов регистрируется с менеджером транзакций, используя вызов **ax_reg**, только когда он получает требование на свой ресурс. Заметим, что в этом методе производительность не ухудшается, даже если определен только один менеджер ресурсов или если каждый менеджер ресурсов

используется всеми транзакциями, так как в менеджере транзакций вызовы **ax_reg** и **xa_start** обрабатываются одинаково.

Интерфейс XA обеспечивает двухстороннюю связь между менеджером транзакций и менеджером ресурсов. Это интерфейс системного уровня между двумя программными компонентами DTP, а не обычный программный интерфейс, код которого разрабатывают прикладные программисты. Однако программисты должны хорошо знать программные ограничения, которые налагаются компонентами программного обеспечения DTP. Информацию о вопросах программирования интерфейса X/Open XA смотрите в книге *Application Development Guide*.

Хотя интерфейс XA един, у каждого XA-совместимого менеджера транзакций могут быть свои способы интеграции менеджеров ресурсов. Информацию об интеграции вашего продукта DB2 в качестве менеджера ресурсов с конкретным менеджером транзакций смотрите в документации по соответствующему менеджеру транзакций. Информация об интеграции наиболее распространенных мониторов транзакций приводится в разделе “Конфигурирование менеджеров транзакций XA для использования DB2 UDB” на стр. 207.

Задание базы данных в качестве менеджера ресурсов

Каждая база данных определяется для менеджера транзакций как отдельный менеджер ресурсов; база данных должна идентифицироваться строкой xa_open. Описание формата строки DB2 xa_open смотрите в разделе “Использование строк xa_open и xa_close”.

Использование строк xa_open и xa_close

У строки менеджера баз данных xa_open есть два принятых формата. Один формат появился в DB2 Версии 7. Второй формат используется прежними версиями DB2 и оставлен в новой версии для совместимости. Новые реализации баз данных должны использовать новый формат, а более старые следует перенастроить на новый формат по возможности. Будущие версии DB2, возможно, не будут поддерживать старый формат строки xa_open. Информацию об исходном формате строки xa_open смотрите в разделе “Формат строки xa_open для прежних версий DB2” на стр. 198.

При настройке базы данных в качестве менеджера ресурсов строка xa_close необязательна. Если ее задать, менеджер баз данных будет игнорировать ее.

Новый формат строки xa_open для DB2 Версии 7

В DB2 Версии 7 используется следующий новый формат строки xa_open 7:

```
id1_параметра = <значение параметра>,id2_параметра = <значение параметра>, ...
```

Порядок этих параметров несуществен. Допустимые значения для *id_параметра* описаны в следующей таблице.

Таблица 22. Допустимые значения для id_параметра

Имя параметра	Значение	Обязат-ть	Регистрозавис.	Значение по умолчанию
DB	Алиас базы данных	Да	Нет	Нет
Алиас базы данных используется программой для обращения к базе данных.				
UID	User ID	Нет	Да	Нет
ID пользователя, имеющего полномочия соединяться с этой базой данных. Обязательный, если задается пароль.				
PWD	Пароль	Нет	Да	Нет
Пароль, соответствующий данному ID пользователя. Обязателен, если задается ID пользователя.				
TPM	Имя монитора транзакций	Нет	Нет	Нет
Имя используемого монитора транзакций. Поддерживаемые значения смотрите в разделе “Значения параметров TPM и TP_MON_NAME” на стр. 194. Этот параметр можно задать, чтобы несколько мониторов транзакций могли использовать один экземпляр DB2. Задаваемое значение заменяет значение, заданное параметром конфигурации менеджера баз данных *tp_mon_name*.				
AXLIB	Библиотека, содержащая функции монитора транзакций **ax_reg** и **ax_unreg**.	Нет	Да	Нет
Это значение DB2 использует для получения адресов требуемых функций **ax_reg** и **ax_unreg**. Его можно использовать для переопределения значений, основанных на значении параметра TPM, его могут использовать также мониторы транзакций, которых нет в списке TPM.				
CHAIN_END	Флаг цепочки xa_end. Допустимые значения - T, F или нет значения.	Нет	Нет	F
Цепочка XA_END - это оптимизация, которую DB2 может использовать для сокращения сетевых потоков. Если среда монитора транзакций гарантирует, что **xa_prepare** будет вызван в том же потоке или процессе немедленно после вызова **xa_end**, и если CHAIN_END включен, флаг xa_end будет связан с вызовом **xa_prepare**, исключая тем самым одну операцию передачи по сети. Значение T указывает, что CHAIN_END включен; значение F указывает, что CHAIN_END выключен; если значение не задано, предполагается, что CHAIN_END включен. Этот параметр может быть использован для переопределения значения, полученного из заданного значения TPM.				

Таблица 22. Допустимые значения для id_параметра (продолжение)

Имя параметра	Значение	Обязат-ть	Регистрозавис.	Значение по умолчанию
SUSPEND_CURSOR	Задает, должны ли храниться указатели, если поток транзакции управления приостанавливается. Допустимые значения - T, F или нет значения.	Нет	Нет	F
Мониторы транзакций, которые приостановили выполнение ветви транзакции, могут повторно использовать приостановленный поток или процесс для других транзакций. В таких ситуациях указатели надо закрыть, чтобы их не унаследовала новая транзакция. Когда приостановленная транзакция возобновляется, программа должна получить эти указатели снова. Если SUSPEND_CURSOR включен, никакие открытые указатели не закрываются, но при этом поток или процесс не может повторно использоваться для других транзакций. Для приостановленной транзакции разрешается только возобновление. Значение T указывает, что SUSPEND_CURSOR включен; значение F указывает, что SUSPEND_CURSOR выключен; если значение не задано, предполагается, что SUSPEND_CURSOR включен. Этот параметр может быть использован для переопределения значения, полученного из заданного значения TPM.				
HOLD_CURSOR	Задает, удерживаются ли указатели после принятия транзакции. Допустимые значения - T, F или нет значения.	Нет	Нет	F
Мониторы транзакций обычно многократно используют потоки или процессы для нескольких программ. Чтобы гарантировать, что вновь загружаемая программа не унаследует указатели, открытые предыдущей программой, указатели закрываются после принятия. Если HOLD_CURSOR включен, указатели удерживаются после принятия транзакций. Значение T указывает, что HOLD_CURSOR включен; значение F указывает, что HOLD_CURSOR выключен; если значение не задано, предполагается, что HOLD_CURSOR включен. Этот параметр может быть использован для переопределения значения, полученного из заданного значения TPM.				

Значения параметров TPM и TP_MON_NAME

Параметр TPM строки xa_open и параметр конфигурации *tp_mon_name* менеджера баз данных указывают для DB2, какой монитор транзакций используется. Параметр *tp_mon_name* применяется для всего экземпляра DB2. Параметр TPM применяется только для заданного менеджера ресурсов XA. Значение TPM переопределяет значение параметра *tp_mon_name*. Для

параметров TPM и *tp_mon_name* допустимы следующие значения:

Таблица 23. Допустимые значения для параметров TPM и tp_mon_name

Значение TPM	**Монитор транзакций**	**Внутренние параметры**
CICS	IBM TxSeries CICS	AXLIB=libEncServer (для Windows) =/usr/lpp/encina/lib/libEncServer (для систем на основе UNIX) HOLD_CURSOR=T CHAIN_END=T SUSPEND_CURSOR=F
ENCINA	IBM TxSeries Encina Monitor	AXLIB=libEncServer (для Windows) =/usr/lpp/encina/lib/libEncServer (для систем на основе UNIX) HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
MQ	IBM MQSeries	AXLIB=mqmax (для Windows) =/usr/mqm/lib/libmqmax.a (для AIX) =/opt/mqm/lib/libmqmax.a (для Solaris) HOLD_CURSOR=F CHAIN_END=F SUSPEND_CURSOR=F
CB	IBM Component Broker	AXLIB=somtrx1i (для Windows) =libsomtrx1 (для систем на основе UNIX) HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
SF	IBM San Francisco	AXLIB=ibmsfDB2 HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
TUXEDO	BEA Tuxedo	AXLIB=libtux HOLD_CURSOR=F CHAIN_END=F SUSPEND_CURSOR=F
MTS	Microsoft Transaction Server	Конфигурировать DB2 для MTS необязательно. MTS определяется автоматически драйвером DB2 ODBC.

Таблица 23. Допустимые значения для параметров TPM и tp_mon_name (продолжение)

Значение TPM	Монитор транзакций	Внутренние параметры	
JTA	Java Transaction API		Конфигурировать DB2 для серверов Enterprise Java Server (EJS), например для сервера IBM WebSphere, необязательно. Драйвер DB2 JDBC определяет эту среду автоматически.

Примеры

1. Вы используете IBM TxSeries CICS в Windows NT. В документации по TxSeries говорится, что для параметра *tp_mon_name* надо задать значение `libEncServer:C`. Это приемлемый формат, однако в DB2 UDB или DB2 Connect Версии 7 есть выбор:
   - Задать для *tp_mon_name* значение `CICS` (это рекомендуется для данного сценария):

     ```
     db2 update dbm cfg using tp_mon_name CICS
     ```

     Для каждой базы данных, определенной для CICS, в строке инициализации Region-> Resources-> Product-> XAD-> Resource manager (Регион-> Ресурсы-> Продукт-> XAD-> Менеджер ресурсов) задайте:

     ```
     db=алиас_базы_данных,uid=id_пользователя,pwd=пароль
     ```

   - Для каждой базы данных, определенной для CICS, в строке инициализации Region-> Resources-> Product-> XAD-> Resource manager (Регион-> Ресурсы-> Продукт-> XAD-> Менеджер ресурсов) задайте:

     ```
     db=алиас_базы_данных,uid=id_пользователя,pwd=пароль,tpm=cics
     ```

2. Вы используете IBM MQSeries CICS в Windows NT. В документации по MQSeries говорится, что для параметра *tp_mon_name* надо задать значение `mqmax`. Это приемлемый формат, однако в DB2 UDB или DB2 Connect Версии 7 есть выбор:
   - Задать для *tp_mon_name* значение `MQ` (это рекомендуется для данного сценария):

     ```
     db2 update dbm cfg using tp_mon_name MQ
     ```

     Для каждой базы данных, определенной для CICS, в строке инициализации Region-> Resources-> Product-> XAD-> Resource manager (Регион-> Ресурсы-> Продукт-> XAD-> Менеджер ресурсов) задайте:

```
uid=id_пользователя,db=алиас_базы_данных,pwd=пароль
```

- Для каждой базы данных, определенной для CICS, в строке инициализации Region-> Resources-> Product-> XAD-> Resource manager (Регион-> Ресурсы-> Продукт-> XAD-> Менеджер ресурсов) задайте:

```
uid=id_пользователя,db=алиас_базы_данных,pwd=пароль,tpm=mq
```

3. Вы используете в Windows NT и IBM TxSeries CICS, и IBM MQSeries. Используется единственный экземпляр DB2. В этом сценарии следует создать следующую конфигурацию:
   a. Для каждой базы данных, определенной для CICS, в строке инициализации Region-> Resources-> Product-> XAD-> Resource manager (Регион-> Ресурсы-> Продукт-> XAD-> Менеджер ресурсов) задайте:

```
pwd=пароль,uid=id_пользователя,tpm=cics,db=алиас_базы_данных
```

   b. Для каждой базы данных, определенной в качестве ресурса в свойствах менеджера очередей, задайте XaOpenString как:

```
db=алиас_базы_данных,uid=id_пользователя,pwd=пароль,tpm=mq
```

4. Вы разрабатываете свой собственный XA-совместимый менеджер транзакций (XA TM) в Windows NT и хотите указать DB2, что требуемые функции **ax_reg** и **ax_unreg** находятся в библиотеке ″myaxlib″. Библиотека ″maxlib″ находится в каталоге, заданном в операторе PATH. Есть выбор:
   - Задать *tp_mon_name* для библиотеки myaxlib:

```
db2 update dbm cfg using tp_mon_name myaxlib
```

     и для каждой базы данных, определенной для XA TM, задать строку xa_open:

```
db=алиас_базы_данных,uid=id_пользователя,pwd=пароль
```

   - Для каждой базы данных, определенной для XA TM, задать строку xa_open:

```
db=алиас_базы_данных,uid=id_пользователя,pwd=пароль,axlib=myaxlib
```

5. Вы разрабатываете свой собственный XA-совместимый менеджер транзакций (XA TM) в Windows NT и хотите указать DB2, что в библиотеке ″myaxlib″ есть требуемые функции **ax_reg** и **ax_unreg**. Библиотека ″maxlib″ находится в каталоге, заданном в операторе PATH. Кроме того, вы хотите задать цепочку XA END. Есть выбор:
   - Для каждой базы данных, определенной для XA TM, задать строку xa_open:

```
db=алиас_базы_данных,uid=id_пользователя,
pwd=пароль,axlib=myaxlib,chain_end=T
```

   - Для каждой базы данных, определенной для XA TM, задать строку xa_open:

```
db=алиас_базы_данных,uid=id_пользователя,
pwd=пароль,axlib=myaxlib,chain_end
```

Формат строки xa_open для прежних версий DB2

Ниже описывается формат строки xa_open, который использовался в прежних версиях DB2. Этот формат по-прежнему поддерживается из соображений совместимости. По возможности программы следует перенастроить на новый формат (смотрите раздел “Новый формат строки xa_open для DB2 Версии 7” на стр. 192).

Каждая база данных определяется для менеджера транзакций как отдельный менеджер ресурсов и должна идентифицироваться строкой xa_open:

```
"алиас_базы_данных<,id_пользователя,пароль>"
```

Алиас_базы_данных нужен для задания алиаса базы данных. Алиас совпадает с именем базы данных, если после создания базы данных не внести в каталог алиас явно. Имя пользователя и пароль необязательны и, в зависимости от метода аутентификации, используются для задания информации аутентификации.

При задании базы данных в качестве менеджера ресурсов строка xa_close необязательна. Если ее задать, менеджер баз данных будет игнорировать ее.

Изменения на серверах баз данных хоста или AS/400

Возможность изменения для серверов баз данных хоста или AS/400 зависит от архитектуры менеджера транзакций XA. Для поддержки последовательностей принятий от различных процессов должен быть включен концентратор DB2 Connect. Чтобы включить концентратор DB2 Connect EE, задайте для параметра конфигурации менеджера баз данных *max_logicagents* значение больше значения параметра *maxagents*. Имейте в виду, что для поддержки последовательностей принятий XA от различных процессов концентратору DB2 Connect EE требуется клиент DB2 Версии 7.1. Информацию об операторах SQL, допустимых в этой среде, смотрите в книге *Application Development Guide*. Информацию о концентраторе смотрите в книге *DB2 Connect. Руководство пользователя*.

Если вы будете вносить изменения на серверах баз данных хоста или AS/400, понадобится DB2 Connect со сконфигурированным менеджером точек синхронизации (SPM) DB2. Указания смотрите в одной из книг *Quick Beginnings*.

Особенности соединения с базой данных

В этом разделе обсуждаются следующие темы:

- “Оператор RELEASE”
- “Доступ к транзакциям в многораздельных базах данных” на стр. 199

Оператор RELEASE

Если для освобождения соединения с базой данных используется оператор RELEASE, для восстановления соединения следует использовать оператор CONNECT, а не SET CONNECTION.

Доступ к транзакциям в многораздельных базах данных

В среде многораздельных баз данных пользовательские данные могут быть распределены по разделам базы данных. Программа, обращаясь к базе данных, соединяется с одним из ее разделов (узел координатора) и посылает ему требования. Разные программы могут соединяться с разными разделами базы данных, а одна и та же программа может выбирать в разных соединениях разные разделы базы данных.

Для транзакций в среде многораздельных баз данных весь доступ должен осуществляться через *один* раздел базы данных. Это значит, что с момента запуска транзакции и до момента ее принятия (включительно) должен использоваться один и тот же раздел.

Любая транзакция в многораздельной базе данных должна быть принята до того, как произойдет разъединение.

Принятие эвристических решений

XA-совместимый менеджер транзакций (монитор обработки транзакций) использует процесс двухфазного принятия, сходный с процессом, описанным в разделе “Процесс двухфазного принятия” на стр. 180, который использует менеджер транзакций DB2. Принципиальное отличие между этими двумя средами заключается в том, что вместо менеджера транзакций DB2 и базы данных менеджера транзакций функцию регистрации и управления транзакциями обеспечивает монитор транзакций.

При использовании XA-совместимого менеджера транзакций могут случаться ошибки, сходные с ошибками, которые обсуждались для менеджера транзакций DB2 (смотрите раздел “Восстановление после ошибок двухфазного принятия” на стр. 183). Так же, как и менеджер транзакций DB2, XA-совместимый менеджер транзакций будет пытаться ресинхронизировать неоднозначные транзакции.

Если по каким-то причинам вы не можете ждать, пока менеджер транзакций автоматически разрешит неоднозначные транзакции, это можно сделать вручную. Этот процесс восстановления иногда называют ″принятием эвристического решения″.

Команда LIST INDOUBT TRANSACTIONS (с использованием опции WITH PROMPTING) или соответствующий ей набор API позволяет выполнить запрос, принятие или откат неоднозначных транзакций. Кроме того, она позволяет ″забыть″ транзакции, для которых было выполнено эвристическое принятие или откат, удалив записи журнала и освободив место. Для получения информации о неоднозначных транзакциях из DB2 UDB в системах на основе UNIX, в системах Windows или OS/2 соединитесь с базой данных и введите команду LIST INDOUBT TRANSACTIONS WITH PROMPTING или вызовите эквивалентный API. Информацию об этой команде и соответствующих ей API управления смотрите в книгах *Command Reference* или *Administrative API Reference*.

Информацию о неоднозначности транзакций, относящуюся к серверам баз данных хоста или AS/400 можно получить двумя способами:

- Информацию о неоднозначности можно получить непосредственно с сервера хоста или AS/400.

  Чтобы получить информацию о неоднозначности непосредственно от DB2 for OS/390, введите команду DISPLAY THREAD TYPE(INDOUBT). Для принятия эвристического решения используйте команду RECOVER. Чтобы получить информацию о неоднозначности непосредственно от DB2 for OS/400, введите команду **wrkcmtdfn**.
- Информацию о неоднозначности можно получить от сервера DB2 Connect, использовавшегося для доступа к серверу баз данных хоста или AS/400.

  Чтобы получить информацию о неоднозначности от сервера DB2 Connect, сначала соединитесь с менеджером точек синхронизации DB2, соединившись с экземпляром DB2, который задается значением параметра конфигурации менеджера баз данных *spm_name*. Затем введите команду LIST DRDA INDOUBT TRANSACTIONS WITH PROMPTING для вывода неоднозначных транзакций и принятия эвристических решений.

Пользуйтесь этими командами (или соответствующими API) с *чрезвычайной осторожностью* и только в крайнем случае. Лучшая стратегия - это подождать, пока ресинхронизацию проведет менеджер транзакций. Если выполнить вручную принятие или откат транзакции в одной из участвующих баз данных, могут возникнуть ошибки, связанные с нарушением целостности данных. Исправление таких ошибок требует от вас понимания логики программы, чтобы идентифицировать данные, которые были изменены или для которых был выполнен откат, а затем выполнить восстановление базы данных до указанного момента времени или вручную отменить или применить изменения повторно.

Если вы не можете ждать, пока менеджер транзакций инициирует процесс ресинхронизации, а нужно освободить ресурсы, связанные неоднозначной транзакцией, необходимы эвристические операции. Такая ситуация может случиться, если менеджер транзакций не в состоянии выполнить ресинхронизацию в течение длительного времени, и неоднозначная транзакция связывает ресурсы, которые срочно требуются. Неоднозначной транзакцией оказываются связаны те ресурсы, которые были назначены ей до того, как стал недоступен менеджер транзакций или менеджер ресурсов. Для менеджера баз данных к таким ресурсам относятся блокировки в таблицах и индексах, пространство журнала и память, задействованная транзакцией. Кроме того, каждая транзакция сокращает (на единицу) максимальное число параллельных транзакций, которые может обработать база данных.

Хотя универсального рецепта выполнения эвристических операций не существует, приведем несколько общих указаний:

1. Соединитесь с базой данных, для которой требуется выполнение всех транзакций.

2. Для вывода неоднозначных транзакций воспользуйтесь командой LIST INDOUBT TRANSACTIONS. *xid* - это ID глобальной транзакции; он идентичен *xid*, использовавшемуся менеджером транзакций и остальными менеджерами ресурсов, участвующими в транзакции.
3. Для каждой неоднозначной транзакции используйте известную информацию о программе и операционной среде, чтобы определить другие участвующие менеджеры ресурсов.
4. Определите доступность менеджера ресурсов:
   - Если менеджер ресурсов доступен, а неоднозначная транзакция была вызвана менеджером ресурсов, недоступным во второй фазе двухфазного принятия либо в предыдущем процессе ресинхронизации, следует проверить журнал менеджера ресурсов, чтобы определить, какое действие предпринималось другими менеджерами ресурсов. Затем следует выполнить то же действие на этой базе данных, то есть, воспользовавшись командой LIST INDOUBT TRANSACTIONS, выполнить либо эвристическое принятие транзакции, либо ее эвристический откат.
   - Если менеджер транзакций *недоступен*, необходимо будет посмотреть состояние транзакции в других участвующих менеджерах ресурсов, чтобы определить, какое действие вы должны выполнить:
     – Если по крайней мере один из остальных менеджеров ресурсов выполнил принятие данной транзакции, в остальных менеджерах ресурсов следует выполнить эвристическое принятие транзакции.
     – Если по крайней мере один из остальных менеджеров ресурсов выполнил откат данной транзакции, следует выполнить эвристический откат транзакции.
     – Если транзакция находится в ″подготовленном″ (неоднозначном) состоянии во всех участвующих менеджерах ресурсов, следует выполнить эвристический откат транзакции.
     – Если один или несколько из других менеджеров ресурсов недоступны, следует выполнить эвристический откат транзакции.

Не выполняйте эвристическую функцию forget (″забыть″), если только эвристическое принятие или откат транзакции не приведет к переполнению журнала, которое указывается в выводе команды INDOUBT TRANSACTIONS. Эвристическая функция forget освобождает пространство журнала, занятое неоднозначной транзакцией. Это значит, что, если менеджер транзакций в конечном счете выполнит операцию ресинхронизации для данной неоднозначной транзакции, потенциально он может принять неправильное решение для другого менеджера ресурсов, так как в этом менеджере ресурсов будет отсутствовать запись журнала для данной транзакции. Обычно ″пропущенная″ запись журнала подразумевает выполнение менеджером ресурсов отката транзакции.

Особенности защиты

Монитор транзакций заранее размещает набор процессов сервера и запускает транзакции от различных пользователей с ID процессов сервера. Для базы данных каждый процесс сервера выглядит как большая программа, состоящая из множества единиц работы, которые все запускаются под одним ID, относящимся к данному процессу сервера.

Например, в среде AIX с использованием CICS, где запускается регион TXSeries CICS, этот ID соответствует имени пользователя AIX, под которым он определен. Все процессы CICS Application Server также выполняются с этим ″главным″ ID TXSeries CICS, который обычно задается как ″cics″. Пользователи CICS могут вызывать транзакции CICS под своими ID регистрации DCE, а находясь в CICS, могут также изменять свои ID с использованием транзакции регистрации CESN. В любом случае ID конечного пользователя недоступен для менеджера ресурсов. Таким образом, CICS Application Process может выполнять транзакции с несколькими именами пользователей, но для менеджера ресурсов они будут выглядеть как единая программа с несколькими единицами работы, но одним ID ″cics″. Вы можете, если хотите, задать ID пользователя и пароль в строке xa_open; тогда для соединения с базой данных вместо ID ″cics″ будет использоваться этот ID пользователя.

Это не повлияет существенно на статические операторы SQL, так как для доступа к базе данных используются не привилегии конечных пользователей, а привилегии связывающего. Однако это означает, что привилегия пакетов базы данных EXECUTE должна быть предоставлена ID сервера, а не ID конечного пользователя.

Для динамических операторов, аутентификация которых производится во время выполнения, привилегии доступа к объектам базы данных должны быть предоставлены ID сервера, а не реальным пользователям этих объектов. Вместо того, чтобы доверить управление доступом к базе данных заданным пользователям, вы должны положиться на систему монитора транзакций для определения, какие пользователи смогут запускать конкретные программы. ID сервера нужно предоставить все привилегии, которые требуются пользователям SQL.

Чтобы определить, у кого есть доступ к таблице или производной таблице базы данных, можно выполнить следующие действия:

1. Из производной таблицы каталога SYSCAT.PACKAGEDEP получите список всех пакетов, зависимых от данной таблицы или производной таблицы.
2. Определите имена программ сервера (например, программ CICS), соответствующие этим пакетам по соглашению об именах, использующемуся в вашей системе.

3. Определите программы клиентов (например, ID транзакций CICS), которые могут вызывать эти программы, а затем с помощью журнала монитора транзакций (например, журнала CICS) определите, кто и когда запускал эти транзакции или программы.

Особенности конфигурирования

При настройке среды монитора транзакций следует рассмотреть следующие параметры конфигурации:

- *tp_mon_name*

  Этот параметр конфигурации идентифицирует имя используемого продукта монитора транзакций (например, ″CICS″ или ″ENCINA″).

- *tpname*

  Этот параметр конфигурации менеджера баз данных определяет имя удаленной программы транзакций, которую должен использовать клиент базы данных при выдаче требования на выделение серверу баз данных с использованием протокола связи APPC. Значение этого параметра устанавливается в файле конфигурации на сервере, и должно совпадать с именем процессора транзакций, сконфигурированным в программе транзакций SNA. Дополнительную информацию смотрите в руководствах *Quick Beginnings*.

- *tm_database*

  Так как DB2 *не* координирует транзакции в среде XA, этот параметр конфигурации менеджера баз данных для XA-координируемых транзакций не используется.

- *maxappls*

  Этот параметр конфигурации базы данных задает максимальное число допустимых активных программ. Значение этого параметра должно быть равно или больше суммы числа соединяющихся программ плюс число части из этих программ, которые смогут работать параллельно в процессе двухфазного принятия или отката. Затем эту сумму следует увеличить на ожидаемое число неоднозначных транзакций, которые могут существовать в любое время. Дополнительную информацию о неоднозначных транзакциях смотрите в разделе “Восстановление после ошибок двухфазного принятия” на стр. 183.

  Для среды монитора TP (например, TXSeries CICS) значение параметра *maxappls*, возможно, придется увеличить. Это должно помочь запустить все процессы монитора транзакций.

- *autorestart*

  Этот параметр конфигурации определяет, будет ли при необходимости автоматически выполняться процедура перезапуска базы данных (RESTART DATABASE). Значение по умолчанию - `YES` (перезапуск разрешен).

  Чтобы можно было использовать базу данных, содержащую неоднозначные транзакции, для нее должна быть выполнена операция перезапуска. Если

параметр *autorestart* имеет значение NO, после прекращения последнего соединения с базой данных следующее соединение будет неудачным и потребуется явно запустить команду RESTART DATABASE. Это состояние будет существовать, пока неоднозначные транзакции не будут удалены либо операцией ресинхронизации менеджера транзакций, либо с помощью эвристической операции, инициированной администратором. Если при запуске команды RESTART DATABASE в базе данных есть неоднозначные транзакции, выдается сообщение. Администратор может затем использовать команду LIST INDOUBT TRANSACTIONS и другие команды процессора командной строки, чтобы получить информацию об этих неоднозначных транзакциях.

Поддерживаемая функция XA

DB2 Universal Database поддерживает спецификацию XA91, определенную в документе *X/Open CAE Specification Distributed Transaction Processing: The XA Specification*, со следующими исключениями:

- Асинхронные службы

  Спецификация XA позволяет интерфейсу использовать асинхронные службы, поэтому результат требования может быть проверен позднее. Менеджер баз данных требует, чтобы требования вызывались в синхронном режиме.

- Статическая регистрация

  Интерфейс XA предусматривает два способа регистрации менеджера ресурсов: статическую регистрацию и динамическую регистрацию. DB2 Universal Database поддерживает только динамическую регистрацию, которая более совершенна и эффективна. Дополнительную информацию об этих двух методах смотрите в разделе “Менеджеры ресурсов” на стр. 191.

- Передача транзакций

  DB2 Universal Database не поддерживает передачу транзакций от одного управляющего потока другому.

Дополнительную информацию об использовании строк xa_open и xa_close смотрите в разделе “Использование строк xa_open и xa_close” на стр. 192.

Использование и положение переключателя XA

Как того требует интерфейс XA, менеджер баз данных обеспечивает внешнюю переменную *db2xa_switch* языка C типа xa_switch_t для возврата менеджеру транзакций структуры переключателя XA. Кроме адресов различных функций XA, возвращаются следующие поля:

Поле	Значение
name	Имя продукта менеджера баз данных. Например, DB2 for AIX.
flags	TMREGISTER | TMNOMIGRATE Явно устанавливает, что DB2 Universal Database использует динамическую регистрацию и что менеджер транзакций не

должен использовать передачу связей. Неявно устанавливает, что асинхронная работа не поддерживается.

version Должна равняться нулю.

Использование переключателя XA в DB2 Universal Database

Архитектура XA требует, чтобы менеджер ресурсов задал *переключатель*, который бы предоставил менеджеру транзакций XA доступ к программам **xa_** менеджера ресурсов. В переключателе менеджера ресурсов используется структура под названием xa_switch_t. Этот переключатель содержит имя менеджера ресурсов, непустые указатели на точки входа XA менеджера ресурсов, флаг и номер версии.

Системы на основе UNIX и OS/2: Переключатель DB2 UDB можно получить одним из двух способов:

- Через дополнительный уровень косвенной ссылки. В программе на языке C это можно выполнить, определив макрокоманду:

  ```
  #define db2xa_switch (*db2xa_switch)
  ```

  перед использованием *db2xa_switch*.
- С помощью вызова **db2xacic**

  DB2 UDB предоставляет этот API, который возвращает адрес структуры *db2xa_switch*. Прототип этой функции:

  ```
  struct xa_switch_t * SQL_API_FN  db2xacic( )
  ```

При использовании любого из двух методов нужно связать свою программу с библиотекой `libdb2` (в системе на основе UNIX) или `db2api.lib` (в OS/2).

Windows NT: Указатель на структуру *xa_switch*, *db2xa_switch*, экспортируется как данные DLL. Это значит, что программа Windows NT, использующая данную структуру, должна обращаться к ней одним из трех способов:

- Через дополнительный уровень косвенной ссылки. В программе на языке C это можно выполнить, определив макрокоманду:

  ```
  #define db2xa_switch (*db2xa_switch)
  ```

  перед использованием *db2xa_switch*.
- Если используется компилятор Microsoft Visual C++, *db2xa_switch* можно определить так:

  ```
  extern __declspec(dllimport) struct xa_switch_t db2xa_switch
  ```
- С помощью вызова **db2xacic**

  DB2 UDB предоставляет этот API, который возвращает адрес структуры *db2xa_switch*. Прототип этой функции:

  ```
  struct xa_switch_t * SQL_API_FN  db2xacic( )
  ```

При использовании любого из этих методов нужно связать свою программу с библиотекой db2api.lib.

Пример кода на языке C: Следующий код демонстрирует различные способы, с помощью которых к *db2xa_switch* можно обратиться из программы на языке C с любой платформы DB2 UDB. Не забудьте связать свою программу с соответствующей библиотекой.

```
   #include <stdio.h>
   #include <xa.h>

   struct xa_switch_t * SQL_API_FN  db2xacic( );

   #ifdef DECLSPEC_DEFN
   extern __declspec(dllimport) struct xa_switch_t db2xa_switch;
   #else
   #define db2xa_switch (*db2xa_switch)
   extern struct xa_switch_t db2xa_switch;
   #endif
main( )
   {
      struct xa_switch_t *foo;
      printf ( "%s \n", db2xa_switch.name );
      foo = db2xacic();
      printf ( "%s \n", foo->name );
      return ;
   }
```

Диагностика ошибок интерфейса XA

Обнаружив ошибку при обработке требования XA от менеджера транзакций, программа может оказаться не в состоянии получить от менеджера транзакций код ошибки. Если программа завершается аварийно или получает от монитора транзакций или менеджера транзакций непонятный код возврата, следует проверить журнал First Failure Service Log; информация об ошибке XA заносится в него, если действует уровень диагностики 3 или выше. Дополнительную информацию о журнале First Failure Service Log смотрите в руководстве *Troubleshooting Guide*.

Следует также просмотреть сообщение консоли, файл ошибок менеджера транзакций или другую специфическую для продукта информацию о внешних программах обработки транзакций, которые вы используете.

Менеджер баз данных записывает все ошибки XA в журнал First Failure Service Log с SQLCODE -998 (ошибки транзакций или эвристические ошибки) и соответствующими кодами причины. Вот некоторые из наиболее типичных ошибок:

- Недопустимый синтаксис в строке xa_open.
- Ошибка соединения с базой данных, заданной в строке open, по одной из следующих причин:

- База данных не внесена в каталог.
- База данных не была запущена.
- Имя пользователя программы сервера или пароль не прошли авторизацию для соединения с базой данных.

• Ошибка связи.

Вот пример записи в журнале ошибок для ошибки строки xa_open (строка xa_open пропущена), сгенерированной в AIX:

```
Tue Apr  4 15:59:08 1995
toop pid(83378) process (xatest) XA DTP Support      sqlxa_open Probe:101
DIA4701E Database "" could not be opened for distributed transaction
processing.
String Title : XA Interface SQLCA  pid(83378)
SQLCODE = -998  REASON CODE: 4  SUBCODE: 1
Dump File : /u/toop/diagnostics/83378.dmp Data : SQLCA
```

Конфигурирование менеджеров транзакций XA для использования DB2 UDB

Обратите внимание на то, что информация этого раздела гораздо подробнее изложена в соответствующем разделе книги *Руководство администратора: Производительность*.

Ниже описано, как конфигурировать отдельные продукты для использования DB2 в качестве менеджера ресурсов. Можно задать любой из следующих продуктов:

- “Конфигурирование IBM TXSeries CICS”
- “Конфигурирование IBM TXSeries Encina”
- “Конфигурирование BEA Tuxedo” на стр. 210
- “Конфигурирование сервера Microsoft Transaction Server” на стр. 212.

Конфигурирование IBM TXSeries CICS

Информацию о конфигурировании IBM TXSeries CICS для использования DB2 в качестве менеджера ресурсов смотрите в вашем руководстве *IBM TXSeries CICS Administration Guide*. Документацию по TXSeries можно просмотреть на сайте http://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/.

Серверы баз данных хоста и AS/400 могут участвовать в транзакциях, координируемых CICS.

Конфигурирование IBM TXSeries Encina

Ниже приводятся различные API и параметры конфигурации, требуемые для интеграции монитора Encina Monitor с серверами DB2 Universal Database или DB2 for MVS, DB2 for OS/390, DB2 for AS/400 или DB2 for VSE&VM при обращении через DB2 Connect. Документацию по TXSeries можно просмотреть на сайте http://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/.

Серверы баз данных хоста и AS/400 могут участвовать в транзакциях, координируемых Encina.

Конфигурирование DB2

Чтобы сконфигурировать DB2:

1. Каждая база данных должна быть зарегистрирована в каталоге баз данных DB2. Если это отдаленная база данных, для нее должна также быть запись в каталоге узлов. Конфигурирование можно выполнить при помощи графического интерфейса Ассистента конфигурирования клиента (CCA) или процессора командной строки DB2 (CLP). Например:

   ```
   DB2 CATALOG DATABASE inventdb AS inventdb AT NODE host1 AUTH SERVER
   DB2 CATALOG TCPIP NODE host1 REMOTE hostname1 SERVER svcname1
   ```

2. Клиент DB2 может оптимизировать внутреннюю обработку данных для Encina, если он знает, что работает с Encina. Это можно указать, задав для параметра конфигурации менеджера баз данных *tp_mon_name* значение `ENCINA`. По умолчанию специальная оптимизация не производится. Если значение для параметра *tp_mon_name* задано, программа должна обеспечить, чтобы поток, который выполняет единицу работы, немедленно выполнял принятие после ее завершения. Никакие другие единицы работы до этого запускать нельзя. Если вы используете *другую* среду, следует задать для *tp_mon_name* значение `NONE` (или, через CLP значение `NULL`). Этот параметр можно изменить через Центр управления или через процессор командной строки. Команда процессора командной строки выглядит так:

   ```
   db2 update dbm cfg using tp_mon_name ENCINA
   ```

Конфигурирование Encina для каждого менеджера ресурсов

Чтобы сконфигурировать Encina для каждого менеджера ресурсов, администратор должен определить строку открытия, строку закрытия и соглашение о потоке управления для каждой базы данных DB2 как менеджера ресурсов до того, как менеджер ресурсов может быть зарегистрирован для транзакций в прикладной программе. Это можно сделать через полноэкранный интерфейс Enconcole или через интерфейс командной строки Encina. Например:

```
monadmin create rm inventdb -open "db=inventdb,uid=пользователь1,pwd=пароль1"
```

Для каждой базы данных DB2 существует одна конфигурация менеджера ресурсов, и у каждой конфигурации менеджера ресурсов должно быть свое имя rm (″логическое имя менеджера ресурсов″). Для простоты его можно задать таким же, как у базы данных.

Строка xa_open содержит информацию, которая требуется для установления соединения с базой данных. Содержание этой строки определяется конкретным менеджером ресурсов. Строка xa_open для DB2 UDB содержит алиас открываемой базы данных и, необязательно, ID пользователя и пароль, связанные с соединением. Следует отметить, что указанное здесь имя базы данных должно быть зарегистрировано в обычном каталоге баз данных, что

необходимо для любого обращения к базе данных. Информацию о строке xa_open для DB2 смотрите в разделе “Задание базы данных в качестве менеджера ресурсов” на стр. 192.

Строка xa_close в DB2 не используется.

Соглашение о потоке управления определяет, может ли агент прикладной программы обрабатывать несколько транзакций за раз. DB2 UDB поддерживает значение по умолчанию для TMXA_SERIALIZE_ALL_OPERATIONS, то есть поток может вновь использоваться только после завершения транзакции.

При обращении к DB2 for OS/390, DB2 for MVS, DB2 for AS/400 или DB2 for VSE&VM следует использовать менеджер точек синхронизации DB2. Инструкции по конфигурированию смотрите в руководстве *DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings*.

Обращение к базе данных DB2 из прикладной программы Encina

Чтобы обратиться к базе данных DB2 из прикладной программы Encina:

1. При помощи API Encina Scheduling Policy задайте, сколько агентов прикладных программ можно запустить из одного процесса прикладной программы монитора транзакций. Например:

   ```
   rc = mon_SetSchedulingPolicy (MON_EXCLUSIVE)
   ```

   Для DB2 (серверы баз данных DB2 Universal Database, хоста или AS/400) надо использовать значение по умолчанию MON_EXCLUSIVE. Это гарантирует, что:

   - На время существования транзакции процесс прикладной программы блокируется.
   - Программа работает как однопоточная.

   Примечание: При использовании ODBC или интерфейса уровня вызовов DB2 надо отключить поддержку многопоточности. Это можно сделать, задав значение параметра конфигурации интерфейса уровня вызовов DB2 `DISABLEMULTITHREAD = 1` (отключение поддержки многопоточности). По умолчанию для DB2 Universal Database используется `DISABLEMULTITHREAD = 0` (включение поддержки многопоточности). Дополнительную информацию смотрите в справочнике *CLI Guide and Reference*.

2. При помощи API регистрации менеджера ресурсов Encina задайте переключатель XA и логическое имя менеджера ресурсов, используемые Encina при обращении к менеджер ресурсов в процессе прикладной программы. Например:

   ```
   rc = mon_RegisterRmi ( &db2xa_switch,   /* переключатель xa */
                        "inventdb",   /* логическое имя менеджера ресурсов */
              &rmiId );        /* внутренний ID для менеджера ресурсов */
   ```

Переключатель XA содержит адреса подпрограмм XA в менеджере ресурсов, которые может вызвать менеджер транзакций, а также задает функциональность, обеспечиваемую менеджером ресурсов. Переключатель XA для DB2 Universal Database называется `db2xa_switch` и находится в библиотеке клиента DB2 (`db2app.dll` - для операционных систем Windows и OS/2, и `libdb2` - для систем на основе UNIX).

Логическое имя менеджера ресурсов - это имя, которое использует Encina, а не действительное имя базы данных, которое использует прикладная программа SQL, работающая под Encina. Действительное имя базы данных задается в строке xa_open в API регистрации менеджера ресурсов Encina. В данном примере логическое имя менеджера ресурсов совпадает с именем базы данных.

Третий параметр возвращает внутренний идентификатор или хэндл, который используется TM для ссылок на это соединение.

Примечание: При использовании Encina для обработки транзакций с DB2 через интерфейс XA менеджера транзакций учтите, что вложенные транзакции в настоящее время интерфейсом XA DB2 не поддерживаются. Избегайте использовать такие транзакции, если это возможно. Если же избежать их нельзя, постарайтесь обеспечить, чтобы обработка SQL выполнялась только на одном члене семейства транзакций Encina.

Конфигурирование BEA Tuxedo

Чтобы сконфигурировать Tuxedo для использования DB2 в качестве менеджера ресурсов, выполните следующие действия:

1. Установите Tuxedo согласно документации по этому продукту. Надо выполнить все основное конфигурирование для Tuxedo, включая файлы журналов и переменные среды.

   Кроме того, потребуется компилятор и клиент разработки программ DB2. При необходимости установите их.
2. Под ID сервера Tuxedo задайте значение переменной среды DB2INSTANCE - экземпляр, содержащий базу данных, которую вы хотите использовать с Tuxedo. В переменную PATH включите каталоги программ DB2. Подтвердите, что ID сервера Tuxedo может соединяться с базами данных DB2.
3. Измените значение параметра конфигурации менеджера баз данных *tp_mon_name* на `TUXEDO`.
4. Добавьте определение для DB2 в файл определений менеджера ресурсов Tuxedo. В приведенных ниже примерах `UDB_XA` - это заданное локально имя менеджера ресурсов Tuxedo для DB2, а *db2xa_switch* - имя, заданное в DB2 для структуры типа `xa_switch_t`:
   - Для AIX. В файл `${TUXDIR}/udataobj/RM` добавьте определение:

```
# DB2 UDB
   UDB_XA:db2xa_switch:-L${DB2DIR} /lib -ldb2
```

где {TUXDIR} - каталог установки Tuxedo, а {DB2DIR} - каталог экземпляра DB2.

- Для Windows NT. В файл %TUXDIR%\udataobj\rm добавьте определение:

```
# DB2 UDB
   UDB_XA;db2xa_switch;%DB2DIR%\lib\db2api.lib
```

где %TUXDIR% - каталог установки Tuxedo, а %DB2DIR% - каталог экземпляра DB2.

5. Смонтируйте программу сервера монитора транзакций Tuxedo для DB2:
   - В системах AIX:

     ```
     ${TUXDIR}/bin/buildtms -r UDB_XA -o ${TUXDIR}/bin/TMS_UDB
     ```

     где {TUXDIR} - каталог установки Tuxedo.
   - Для Windows NT:

     ```
     %TUXDIR%\bin\buildtms -r UDB_XA -o %TUXDIR%\bin\TMS_UDB
     ```
6. Постройте серверы прикладных программ. В следующих примерах опция -r определяет имя менеджера ресурсов, опция -f (использованная один или несколько раз) определяет файлы, содержащие службы прикладных программ, опция -s задает имена служб прикладных программ для этого сервера, а опция -o задает имя выходного файла сервера:
   - В системах AIX:

     ```
     ${TUXDIR}/bin/buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
        -o UDBserver
     ```

     где {TUXDIR} - каталог установки Tuxedo.
   - Для Windows NT:

     ```
     %TUXDIR%\bin\buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
        -o UDBserver
     ```

     где %TUXDIR% - каталог установки Tuxedo.
7. Задайте в файле конфигурации Tuxedo ссылку на сервер DB2. В раздел *GROUPS файла UDBCONFIG, добавьте запись, подобную следующей:

   ```
   UDB_GRP   LMID=simp GRPNO=3
     TMSNAME=TMS_UDB TMSCOUNT=2
     OPENINFO="UDB_XA:db=пример,uid=пользователь_db2,
     pwd=пароль_пользователя_db2"
   ```

   где параметр TMSNAME задает программу сервера монитора транзакций, которую вы смонтировали ранее, а параметр OPENINFO задает имя менеджера ресурсов. За ним следует имя базы данных, имя пользователя DB2 и пароль, которые используются для аутентификации.

Ссылки на построенные вами ранее серверы находятся в разделе *SERVERS файла конфигурации Tuxedo.

8. Если программа обращается к данным, находящимся в DB2 for OS/390, DB2 for OS/400 или DB2 for VM&VSE, требуется концентратор XA для DB2 Connect. Подробности конфигурирования и ограничения смотрите в книге *DB2 Connect. Руководство пользователя*.
9. Запустите Tuxedo:

   ```
   tmboot -y
   ```

   Когда эта команда будет выполнена, сообщение Tuxedo должно показать, что серверы запущены. Кроме того, если вы введете команду DB2 LIST APPLICATIONS ALL, вы должны увидеть два соединения (в данном случае задаваемые параметром TMSCOUNT в группе UDB файла конфигурации Tuxedo `UDBCONFIG`).

Конфигурирование сервера Microsoft Transaction Server

DB2 UDB, начиная с Версии 5.2, может быть полностью интегрирована с сервером транзакций MTS (Microsoft Transaction Server) Версии 2.0. Программы, запускаемые под MTS в 32-битных операционных системах Windows, могут использовать MTS для координирования двухфазного принятия с несколькими серверами баз данных DB2 UDB, хоста и AS/400, а также с другими совместимыми с MTS менеджерами ресурсов.

Включение поддержки MTS в DB2

Поддержка Microsoft Transaction Server включается автоматически. Вы можете установить для параметра конфигурации *tp_mon_name* менеджера баз данных значение `MTS`, но это необязательно, и заданное значение будет проигнорировано.

Примечание: На Web-сайте IBM можно получить дополнительную техническую информацию, которая поможет вам установить и сконфигурировать поддержку DB2 MTS. Введите адрес http://www.ibm.com/software/data/db2/library/, и поищите DB2 Universal Database ″Technote″ с ключевым словом ″MTS″.

Предварительные требования для программного обеспечения MTS

Для поддержки MTS требуется DB2 Client Application Enabler (CAE) Версии 5.2 или более новой, а сервер транзакций MTS должен быть версии 2.0 с последними исправлениями (выпуск 0772 или новее).

В 32-битных операционных системах Windows при установке драйвера DB2 ODBC в реестр автоматически будет добавлено новое ключевое слово:

```
HKEY_LOCAL_MACHINE\software\ODBC\odbcinit.ini\IBM DB2 ODBC Driver:
Keyword Value Name: CPTimeout
Data Type: REG_SZ
Value: 60
```

Установка и конфигурирование

Ниже приводится сводная информация по установке и настройке конфигурации MTS. Для использования поддержки DB2 для MTS вы должны:

1. Установить MTS и клиент DB2 на том компьютере, где запускается прикладная программа MTS.
2. Если серверы баз данных хоста или AS/400 участвуют в многоузловом изменении:
   a. Установить на локальном или удаленном компьютере DB2 Connect Enterprise Edition. DB2 Connect EE позволяет участвовать серверам баз данных хоста или AS/400 в транзакции многоузлового изменения.
   b. Включить для вашего сервера DB2 Connect EE поддержку многоузлового изменения. Информацию о включении поддержки многоузловых изменений DB2 Connect смотрите в руководстве DB2 Connect Enterprise Edition Quick Beginnings для вашей платформы.

При запуске программ DB2 CLI/ODBC нельзя изменять значения по умолчанию, заданные в файле `db2cli.ini`, для следующих ключевых слов конфигурации:

- Ключевое слово CONNECTYPE (значение по умолчанию 1)
- Ключевое слово MULTICONNECT (значение по умолчанию 1)
- Ключевое слово DISABLEMULTITHREAD (значение по умолчанию 1)
- Ключевое слово CONNECTIONPOOLING (значение по умолчанию 0)
- Ключевое слово KEEPCONNECTION (значение по умолчанию 0)

Программы DB2 CLI, написанные для поддержки MTS, не должны изменять значения атрибутов, соответствующие этим ключевым словам. Кроме того, такие программы не должны изменять значения по умолчанию следующих атрибутов:

- SQL_ATTR_CONNECT_TYPE (по умолчанию SQL_CONCURRENT_TRANS)
- SQL_ATTR_CONNECTON_POOLING (по умолчанию SQL_CP_OFF)

Примечание: На Web-сайте IBM можно получить дополнительную техническую информацию, которая поможет вам установить и сконфигурировать поддержку DB2 MTS. Введите адрес http://www.ibm.com/software/data/db2/library/, и поищите DB2 Universal Database ″Technote″ с ключевым словом ″MTS″.

Проверка правильности установки

1. Сконфигурируйте для вашего клиента DB2 и DB2 Connect EE доступ к серверу баз данных DB2 UDB, хоста или AS/400.
2. Проверьте соединение с компьютера DB2 CAE к серверам баз данных DB2 UDB.
3. Проверьте соединение с компьютера DB2 Connect к серверу баз данных хоста или AS/400 с DB2 CLP и введите несколько запросов.

4. Проверьте соединение с компьютера DB2 CAE через шлюз DB2 Connect к серверу баз данных хоста или AS/400 и введите несколько запросов.

Поддерживаемые серверы баз данных DB2

Многоузловое изменение с использованием транзакций, координируемых MTS, поддерживается следующими серверами:

- DB2 Universal Database Enterprise Edition Версии 5.2
- DB2 Enterprise - Extended Edition Версии 5.2
- DB2 for OS/390
- DB2 for MVS
- DB2 for AS/400
- DB2 for VM&VSE
- DB2 Common Server for SCO Версии 2
- DB2 Universal Database for AIX с исправлениями PTF U453782
- DB2 Universal Database for HP-UX с исправлениями PTF U453784
- DB2 Universal Database Enterprise Edition for OS/2 с исправлениями PTF WR09033
- DB2 Universal Database for SOLARIS с исправлениями PTF U453783
- DB2 Universal Database Enterprise Edition for Windows NT с исправлениями PTF WR09034
- DB2 Universal Database Extended Enterprise Edition для UNIX или Windows NT.

Срок ожидания транзакции MTS и поведение соединения DB2

Значение срока ожидания транзакции можно установить через MTS Explorer. Дополнительную информацию смотрите в электронном руководстве *MTS Administrator Guide*.

Если транзакция продолжается дольше срока ожидания (значение по умолчанию - 60 секунд), MTS асинхронно выдаст команду прекращения работы всех вовлеченных менеджеров ресурсов и вся транзакция будет прервана.

Для соединения с сервером DB2 прекращение транзакции преобразуется в требование отката DB2. Как и любое другое требование базы данных, это требование отката преобразуется при соединении в последовательную форму для гарантии целостности данных на сервере баз данных.

В результате:

- Если подключение незанято, откат будет выполнен немедленно.
- Если продолжается обработка оператора SQL, требование отката ждет завершения этого оператора.

Поддержка пула соединений

Пулы соединений позволяют программам использовать соединение из пула, поэтому отпадает необходимость каждый раз переустанавливать соединение, когда оно используется. Как только соединение создано и помещено в пул, программа может использовать его, не выполняя полный процесс соединения. Когда программа отсоединяется от источника данных ODBC, соединение помещается в пул; оно будет предоставлено для нового соединения с совпадающими атрибутами.

Функция пулов соединений уже использовалась в менеджере драйверов ODBC 2.x. В самом последнем менеджере драйверов ODBC (версия 3.5), поставляемом с MTS, функция пулов соединений отличается некоторыми особенностями конфигурации и новым поведением соединений ODBC для объектов транзакций MTS COM (смотрите раздел “Многократное использование соединений ODBC между объектами COM, участвующих в одной транзакции” на стр. 216).

Менеджер драйверов ODBC версии 3.5 перед тем, как разрешить активацию пула соединений, требует, чтобы драйвер ODBC зарегистрировал в реестре новое ключевое слово. Это ключевое слово:

```
Key Name: SOFTWARE\ODBC\ODBCINST.INI\IBM DB2 ODBC DRIVER
Name: CPTimeout
Type: REG_SZ
Data: 60
```

Драйверы DB2 ODBC Версии 6 и более новых для 32-битных операционных систем Windows полностью поддерживают пулы соединений, и данное ключевое слово регистрируется. Клиенты Версии 5.2 должны установить пакет исправлений 3 (WR09024) или более новый.

Значение по умолчанию - 60 - означает, что соединение будет находиться в пуле в течение 60 секунд до его отключения.

В средах с высокой занятостью лучше увеличить значение CPTimeout срока ожидания для пула соединений (в определенных средах Microsoft иногда предлагает значение 10 минут), чтобы избежать роста числа физических соединений и рассоединений, так как при этом потребляется много системных ресурсов, включая системную память и ресурсы стеков связи.

Кроме того, чтобы то же самое соединение использовалось между объектами в той же транзакции на многопроцессорном компьютере, надо отключить поддержку нескольких пулов на один процессор. Для этого надо скопировать следующее значение реестра в файл под названием `odbcpool.reg`, сохранить его как простой текстовый файл и выполнить команду **odbcpool.reg**. Операционная система Windows импортирует эти значения реестра.

```
REGEDIT4

[HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\ODBC Connection Pooling]
"NumberOfPools"="1"
```

Если не задать для этого ключевого слова значение 1, MTS может поместить соединения в разные пулы, что не позволит повторно использовать то же самое соединение.

Объединение в пул соединений MTS при помощи ADO 2.1 и более новых

Если COM-объекты MTS используют для обращения к базе данных ADO, надо отключить объединение в пул ресурсов OLEDB, чтобы провайдер Microsoft OLEDB для ODBC (MSDASQL) не мешал объединять в пул соединения ODBC. Эта возможность при инициализации отключается в ADO 2.0, но включается в ADO 2.1. Чтобы отключить объединение в пул ресурсов OLEDB, надо скопировать следующее значение реестра в файл под названием `oledb.reg`, сохранить его как простой текстовый файл и выполнить команду **oledb.reg**. Операционная система Windows импортирует это значение реестра.

```
REGEDIT4

[HKEY_CLASSES_ROOT\CLSID\{c8b522cb-5cf3-11ce-ade5-00aa0044773d}]
@="MSDASQL"
"OLEDB_SERVICES"=dword:fffffffc
```

Многократное использование соединений ODBC между объектами COM, участвующих в одной транзакции

В соединениях ODBC в объектах MTS COM используется пул соединений, включаемый автоматически (независимо от участия объектов COM в транзакциях).

Если несколько объектов MTS COM участвуют в одной и той же транзакции, соединение между ними может использоваться многократно следующим образом.

Предположим, что есть два объекта COM - COM1 и COM2, которые соединяются с одним источником данных ODBC и участвуют в одной транзакции.

После того как COM1 соединится и выполнит свою работу, он отсоединяется и соединение помещается в пул. Однако это соединение будет зарезервировано для использования другими объектами COM той же транзакции. Оно станет доступно для других транзакций, только когда закончится текущая.

Когда в этой же транзакции вызывается COM2, предоставляется соединение из пула. MTS гарантирует, что данное соединение будет предоставлено только объектам COM, участвующим в той же транзакции.

С другой стороны, если объект COM1 не отключится явно, он займет это соединение до окончания транзакции. Когда в этой же транзакции будет вызван COM2, ему будет предоставлено отдельное соединение. С этого момента данная транзакция будет занимать два соединения вместо одного.

Такая функция многократного использования соединений для объектов COM, участвующих в одной транзакции, имеет следующие преимущества:

- На клиенте и на сервере используется меньше ресурсов. Необходимо только одно соединение.
- Устраняется возможность того, что два соединения, участвующие в одной транзакции (обращающиеся к одному серверу баз данных и к одним и тем же данным), будут блокировать друг друга, так как серверы DB2 обрабатывают разные соединения из объектов MTS COM как отдельные транзакции.

Настройка связи TCP/IP

Если в среде с высокой рабочей нагрузкой, где одновременно происходит слишком много физических соединений и рассоединений, используется маленькое значение CPTimeout, стек TCP/IP может столкнуться с ограничениями ресурсов.

Для преодоления этой проблемы используйте записи реестра TCP/IP. Они описаны в книге *Windows NT Resource Guide*, Том 1. Значения ключей реестра находятся в HKEY_LOCAL_MACHINE-> SYSTEM-> CurrentControlSet-> Services-> TCPIP-> Parameters.

Значения по умолчанию и предлагаемые параметры:

Имя	**Значение по умолчанию**	**Предлагаемое значение**
KeepAlive time	7200000 (2 часа)	То же
KeepAlive interval	1000 (1 секунда)	10000 (10 секунд)
TcpKeepCnt	120 (2 минуты)	240 (4 минуты)
TcpKeepTries	20 (20 попыток)	То же
TcpMaxConnectAttempts	3	6
TcpMaxConnectRetransmission	3	6
TcpMaxDataRetransmission	5	8
TcpMaxRetransmissionAttempts	7	10
Если значение реестра не задано, создайте его.		

Тестирование DB2 программой примера MTS ″BANK″

Программу примера ″BANK″, поставляемую с MTS, можно использовать для проверки настройки клиентских продуктов и MTS.

Выполните следующие шаги:

1. Измените файл `\Program Files\Common Files\ODBC\Data Sources\MTSSamples.dsn`, чтобы он выглядел так:

   ```
   [ODBC]
   DRIVER=IBM DB2 ODBC DRIVER
   UID=ваш_id_пользователя
   PWD=ваш_пароль
   DSN=алиас_вашей_базы_данных
   Description=MTS Samples
   ```

   где:

   - *ваш_id_пользователя* и *ваш_пароль* - это ID пользователя и пароль, используемые для соединения с хостом.
   - *алиас_вашей_базы_данных* - алиас базы данных, используемый для соединения с сервером баз данных.

2. Откройте Управление ODBC из Панели управления, выберите закладку **System DSN** и добавьте источник данных:
   a. Выберите драйвер IBM ODBC и нажмите кнопку **Готово**.
   b. Когда появится список алиасов базы данных, выберите алиас, заданный ранее.
   c. Нажмите кнопку **OK**.
3. Воспользуйтесь DB2 CLP, чтобы соединиться с базой данных DB2 с указанным выше ID *ваш_id_пользователя*.
   a. Свяжите файл `db2cli.lst`:

      ```
      db2 bind @C:\sqllib\bnd\db2cli.lst blocking all grant public
      ```

   b. Свяжите утилиты.

      Если сервер - сервер хоста DRDA, свяжите `ddcsmvs.lst`, `ddcs400.lst` или `ddcsvm.lst` в зависимости от хоста, с которым соединяетесь (OS/390, AS/400 или VSE&VM). Например:

      ```
      db2 bind @C:\sqllib\bnd\@ddcsmvs.lst blocking all grant public
      ```

      Иначе свяжите файл `db2ubind.lst`:

      ```
      db2 bind @C:\sqllib\bnd\@db2ubind.lst blocking all grant public
      ```

   c. Создайте для программы примера MTS таблицу и данные примера:

      ```
      db2 create table account (accountno int, balance int)
      db2 insert into account values(1, 1)
      ```

4. Убедитесь, что на клиенте DB2 для параметра конфигурации менеджера баз данных *tp_mon_name* задано значение `MTS`.
5. Запустите программу ″BANK″. Нажмите кнопку **Account** и выберите опцию **Visual C++**, затем отправьте требование. Другие опции могут использовать SQL, специфичный для сервера SQL, и они могут не работать.

Глава 11. Проектирование высокой доступности

DB2 Universal Database обеспечивает *поддержку восстановления при отказах для поддержания высокой доступности* на многих платформах. Возможность восстановления при отказах позволяет автоматически передавать рабочую нагрузку с одного процессора на другой при отказах аппаратуры. Например, на AIX восстановление при отказах поддерживается DB2 UDB при помощи возможностей IBM HACMP (High Availability Cluster Multi-Processing - мультиобработка кластера высокой доступности). В этом разделе для ознакомления с понятиями, связанными с высокой доступностью, использованы примеры из AIX.

HACMP обеспечивает высокую доступность при помощи кластеров процессоров, совместно использующих такие ресурсы, как диски или доступ к сети. Если один процессор прекращает работу, его заменяет другой процессор в кластере.

Существуют три режима поддержки восстановления при отказах:

Горячее резервирование
В этом режиме один процессор используется для запуска экземпляра DB2, а второй процессор находится в режиме резервирования, готовый принять на себя обслуживание экземпляра в случае сбоя операционной системы или аппаратного обеспечения, затрагивающего первый процессор.

Взаимная подмена
В этом режиме:

- Оба процессора используются для выполнения отдельных экземпляров DB2, или
- Один процессор используется для выполнения экземпляра DB2, а другой - для выполнения прикладных программ DB2.

Если на одном из процессоров происходит сбой операционной системы или аппаратного обеспечения, другой процессор берет на себя его задания, в конечном счете выполняя работу обоих процессоров.

Одновременный доступ
В этом режиме несколько процессоров совместно обеспечивают работу одного экземпляра базы данных, используя DB2 Universal Database Enterprise - Extended Edition (EEE). Это происходит с помощью модели ″без совместного использования″, где данные разделены так, что на каждом процессоре в кластере обрабатываются один или несколько разделов. Если происходит отказ в работе операционной системы или аппаратного обеспечения на одном из процессоров, другой процессор

берет на себя обработку его разделов. DB2 UDB EEE для дублирования не использует Concurrent Resource Manager. Дублирование обеспечивается при помощи режима горячего резервирования или взаимной подмены. Возможности режима одновременного доступа требуются только для менеджеров баз данных со структурой совместного использования.

Каждую из этих конфигураций можно использовать для восстановления при отказах одного или нескольких разделов многораздельной базы данных. Кроме того, каждая из них обеспечивает восстановление при отказах в одноразделной установке.

Горячее резервирование

Возможность *горячего резервирования* используется для восстановления всего экземпляра одноразделной базы данных или раздела многораздельной базы данных. При отказе одного процессора другой процессор в кластере может подменить его путем автоматической передачи ему экземпляра. Экземпляр базы данных и сама база данных должны быть доступны обоим процессорам в кластере.

- Путь установки DB2 может или использоваться совместно обеими системами, или находиться в файловой системе, не используемой совместно. В последнем случае уровни установки должны быть идентичными.
- Путь экземпляра DB2 может или находиться в совместно используемой файловой системе, или в файловой системе, для которой вручную задано зеркальное отображение.
- База данных и связанные с ней контейнеры должны находиться в файловых системах (или на устройствах), доступных обеим системам.
- Во время восстановления раздела в конфигурации многораздельной базы данных раздел перезапускается на втором процессоре: сценарий восстановления изменяет файл `db2nodes.cfg`, задавая ссылку на этот раздел для нового процессора, и запускает раздел на этом процессоре.
- Когда происходит восстановление, адреса внешних связей для поддерживаемых протоколов связи передаются прозрачно как часть процедуры восстановления.

Подробную информацию о реальных требованиях к установке и о создании экземпляров смотрите в руководстве *HACMP for AIX, Version 4.2: Installation Guide*, SC23-1940.

Примеры

Для каждого из следующих примеров соответствующий сценарий хранится (в DB2 for AIX) в каталоге `sqllib/samples/hacmp`.

Восстановление экземпляра при отказах

В следующем сценарии восстановления с горячим резервированием на двухпроцессорном кластере HACMP выполняется одноразделный экземпляр базы данных (рис. 44). Сведения о конфигурировании кластера HACMP смотрите в разделе “Ресурсы” на стр. 226.

Рисунок 44. Пример конфигурирования восстановления с горячим резервированием

Оба процессора имеют доступ к каталогу установки, каталогу экземпляра и каталогу базы данных. Процессор 1 активен и работает с экземпляром базы данных ″db2inst″. Процессор 2 не активен и используется для горячего резервирования. На процессоре 1 происходит отказ, и экземпляр берет на себя процессор 2. Как только восстановление будет выполнено, как удаленные, так и локальные прикладные программы получают доступ к базе данных в экземпляре ″db2inst″. Тогда базу данных нужно будет перезапустить вручную или же, если разрешен автоматический запуск, перезапуск будет выполнен при первом соединении с базой данных. В показанном сценарии предполагается, что автоматический запуск не разрешен и перезапуск базы данных выполняет сценарий восстановления. Подробную информацию об автоматическом запуске смотрите в разделе ″Обзор восстановления″ в книге *Руководство администратора: Реализация*.

Сценарий примера:

```
hacmp-s1.sh
```

Восстановление раздела

В следующем сценарии восстановления с горячим резервированием отказ происходит на одном разделе, а не на всем экземпляре. Как и в предыдущем примере, в сценарии используется двухпроцессорный кластер HACMP, но на компьютере обрабатывается один из разделов сервера многораздельных баз данных. Процессор 1 обрабатывает один раздел всей конфигурации, а процессор 2 используется для восстановления при отказах. Когда на процессоре 1 происходит сбой, раздел перезапускается на втором процессоре. Процесс восстановления изменяет файл `db2nodes.cfg`, указывая в нем имя хоста и сетевое имя процессора 2, а потом перезапускает раздел на новом процессоре.

Ниже приводится часть файла `db2nodes.cfg` до и после восстановления. В этом примере узел номер 2 работает на процессоре 1 компьютера HACMP, имя хоста и сетевое имя которого - ″node201″. После восстановления узел номер 2 будет работать на процессоре 2 устройства HACMP, имя хоста и сетевое имя которого - ″node202″.

```
До отказа:
        1 node101 0 node101
        2 node201 0 node201    <= HACMP
        3 node301 0 node301

   db2start nodenum 2 restart hostname node202 port 0 netname node202

После:
        1 node101 0 node101
        2 node202 0 node202    <= HACMP
        3 node301 0 node301
```

Сценарий примера:

```
hacmp-s2.sh
```

Восстановление нескольких логических узлов

Более сложный вариант предыдущего примера предполагает восстановление нескольких логических узлов с переносом их с одного процессора на другой. Мы снова используем ту же конфигурацию кластера HACMP с двумя процессорами, как и выше. Однако в этом сценарии на процессоре 1 выполняются три логических раздела. Ситуация та же, что и в сценарии восстановления одного раздела, но в этом случае при отказе процессора 1 каждый логический раздел должен быть запущен на процессоре 2. Каждый логический раздел должен запуститься в порядке, определенном в файле `db2nodes.cfg`: начинать необходимо всегда с логического раздела с номером порта 0.

Ниже приводится часть файла db2nodes.cfg до и после восстановления. В этом примере три логических раздела определены на процессоре 1 двухпроцессорного кластера HACMP.

```
До отказа:
        1 node101 0 node101
        2 node201 0 node201    <= HACMP
        3 node201 1 node201    <= HACMP
        4 node201 2 node201    <= HACMP
        5 node301 0 node301

   db2start nodenum 2 restart hostname node202 port 0 netname node202
   db2start nodenum 3 restart hostname node202 port 1 netname node202
   db2start nodenum 4 restart hostname node202 port 2 netname node202

После:
        1 node101 0 node101
        2 node202 0 node202    <= HACMP
        3 node202 1 node202    <= HACMP
        4 node202 2 node202    <= HACMP
        5 node301 0 node301
```

Сценарий примера:

```
hacmp-s3.sh
```

Взаимная подмена

В режиме *взаимной подмены* один из процессоров может обеспечить восстановление экземпляра одноразделной базы данных или разделов многораздельной базы данных, выполняя другой экземпляр или другие разделы многораздельной базы данных. Как и в конфигурации горячего резервирования, путь установки, каталог экземпляра и база данных должны быть доступны каждому процессору, который используется в процессе восстановления. Пути установки и экземпляров могут быть в совместно используемой файловой системе, либо зеркально отображаются на отдельные файловые системы.

При использовании стратегии взаимной подмены для восстановления экземпляра необходимо так определить экземпляры, чтобы их можно было одновременно выполнять на одном процессоре. Подробную информацию о реальных требованиях к установке и о создании экземпляров смотрите в руководстве *HACMP for AIX, Version 4.2: Installation Guide*, SC23-1940.

Примеры

Для каждого из следующих примеров соответствующий сценарий хранится (в DB2 for AIX) в каталоге sqllib/samples/hacmp.

Восстановление с несколькими экземплярами DB2

В следующем сценарии восстановления после отказа с несколькими экземплярами используется система HACMP с двумя процессорами ″node10″ и ″node20″.

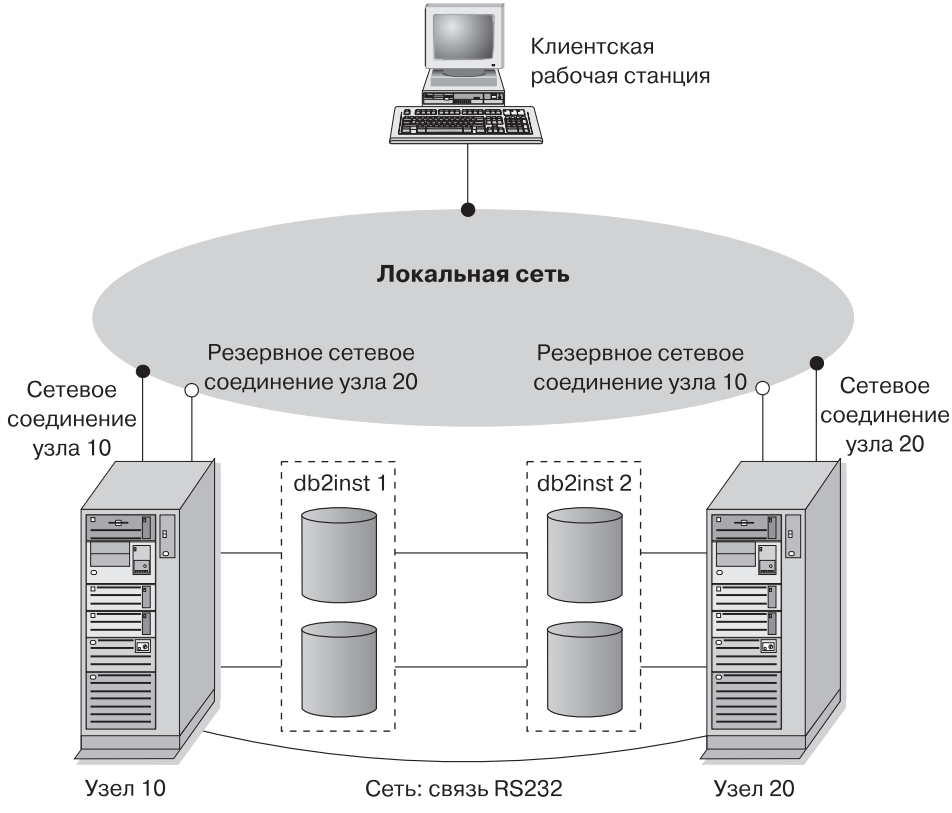


Рисунок 45. Пример восстановления после отказа конфигурации с несколькими экземплярами

Два экземпляра, ″db2inst1″ и ″db2inst2″, созданы из одного пути установки в совместно используемой файловой системе. Экземпляр ″db2inst1″ создается в каталоге `/u/db2inst1`, а экземпляр ″db2inst2″ - в `/u/db2inst2`. Оба этих пути находятся в совместно используемой файловой системе, которая доступна обоим процессорам. У каждого экземпляра есть своя база данных с уникальным путем, который также доступен обоим процессорам.

Оба экземпляра доступны через удаленные клиенты по протоколу TCP/IP: ″db2inst1″ использует имя службы ″db2inst1_port″ (номер порта 5500), а ″db2inst2″ - имя службы ″db2inst2_port″ (номер порта 5550). У удаленных клиентов, получающих доступ к экземпляру ″db2inst1″, этот экземпляр занесен в их каталог узлов под именем хоста ″node10″. У удаленных клиентов,

получающих доступ к экземпляру ″db2inst1″, этот экземпляр занесен в их каталог узлов под именем хоста ″node20″. При нормальных условиях работы ″db2inst1″ работает на ″node10″, а ″db2inst2″ - на ″node20″. При отказе ″node10″ сценарий восстановления запустит ″db2inst1″ на ″node20″, и внешний IP-адрес, связанный с ″node10″, будет переключен в ″node20″. Как только сценарий восстановления запустит экземпляр и перезапустит базу данных, удаленные клиенты смогут соединяться с базой данных этого экземпляра, так же, как если бы он работал на ″node10″.

Сценарий:

```
hacmp-s4.sh
```

Восстановление с несколькими разделами DB2

Восстановление с несколькими разделами в среде сервера многораздельных баз данных требует, чтобы восстановление раздела происходило на логическом узле процессора восстановления. Например, если у вас запущены два раздела сервера многораздельных баз данных на отдельных процессорах двухпроцессорного кластера HACMP, сконфигурированного для взаимной подмены, разделы должны быть восстановлены как логические узлы. Раздел по умолчанию на каждом узле должен быть определен как логический узел 0, то есть когда раздел при восстановлении переносится с одного процессора на другой, он будет запущен как логический узел, у которого не будет принимающих удаленных систем. Такой раздел нельзя использовать как узел координатора.

Другое важное соображение о конфигурировании системы для взаимной подмены относится к пути базы данных локального раздела. Когда база данных создается в среде многораздельных баз данных, она создается на корневом пути, не используемом совместно серверами многораздельных баз данных. Например, рассмотрим такую команду:

```
CREATE DATABASE db_a1 ON /dbpath
```

Эта команда запускается под экземпляром ″db2inst″ и создает базу данных db_a1 в каталоге `/dbpath`. Каждый раздел базы данных создается в своей локальной файловой системе в каталоге `/dbpath/db2inst/node`*xxxx*, где *xxxx* - номер узла. При восстановление HACMP будет пытаться смонтировать файловую систему `/dbpath`, которая уже используется другим процессором. Поэтому сценарий восстановления должен смонтировать файловую систему в другой логической точке и установить символическую связь между этой файловой системой и соответствующим путем `/dpath/db2inst/node`*xxxx*.

Ниже приведена часть файла `db2nodes.cfg` до и после восстановления. В этом примере узел номер 2 работает на процессоре 1 компьютера HACMP, имя хоста и сетевое имя которого - ″node201″. Узел номер 3 работает на процессоре 2 компьютера HACMP, имя хоста и сетевое имя которого - ″node202″.

```
До отказа:
        1 node101 0 node101
        2 node201 0 node201    <= HACMP
        3 node202 0 node202    <= HACMP
        4 node301 0 node301

   db2start nodenum 2 restart hostname node202 port 1 netname node202

После:
        1 node101 0 node101
        2 node202 1 node202    <= HACMP
        3 node202 0 node202    <= HACMP
        4 node301 0 node301
```

Любые удаленные клиенты, пытающиеся после восстановления получить прямой доступ к узлу номер 2 как к координатору, должны перекаталогизировать запись узла для базы данных, указав узел восстановления. Мы не советуем вам использовать сценарий взаимной подмены для узлов координатора. Если для вашего узла координатора требуется дублирование, воспользуйтесь конфигурацией горячего резервирования.

Сценарий:

```
hacmp-s5.sh
```

Соединение после восстановления

Если клиент использует оператор SET CLIENT для соединения с определенным узлом, а этот узел при восстановлении перемещен на другой хост, следующее требование клиента о соединении завершится неудачно. Введите команду **db2stop**, а затем **db2start** *номер_узла* на узле, где был запущен оператор SET CLIENT, а потом перезапустите оператор, чтобы и клиент, и сервер обнаружили новое физическое положение узла назначения.

Ресурсы

Подробную информацию о понятиях HACMP, установке и конфигурировании смотрите в следующих книгах:

- *HACMP for AIX, Version 4.2: Concepts and Facilities*, SC23-1938
- *HACMP for AIX, Version 4.2: Installation Guide*, SC23-1940
- *HACMP for AIX, Version 4.2: Planning Guide*, SC23-1939.

Часть 4. Системы высокой доступности

Глава 12. HACMP ES для AIX (High Availability Cluster Multi-processing, Enhanced Scalability - Мультипроцессорная кластерная обработка с высокой доступностью и улучшенной масштабируемостью)

Улучшенная масштабируемость (ES) - это возможность HACMP для AIX Версии 4.2.2, которая в настоящее время реализована только на узлах RS/6000 SP.

Эта функция обеспечивает такое же восстановление после отказа, как HACMP, и у нее та же структура событий, что у предыдущих версий HACMP (смотрите руководство *HACMP for AIX, V4.2.2, Enhanced Scalability Installation and Administration Guide*). Другие возможности ES:

- Большие кластеры HACMP с возможностью увеличения до 16 узлов на кластер.
- Дополнительные возможности обработки ошибок благодаря *событиям, определяемым пользователем*. Можно следить за событиями, определенными пользователем, которые могут отражать любые ситуации, например, прекращение процесса или момент, когда пространство подкачки приближается к емкости устройства. Такие пред- и постсобытия можно добавить при необходимости в процесс восстановления после отказа. В потоки предсобытий и постсобытий HACMP можно поместить дополнительные функции, зависящие от реализации.

  Файл правил (`/usr/sbin/cluster/events/rules.hacmprd`) содержит события HACMP. К этому файлу добавляются события, определяемые пользователем. В такое определение могут входить файлы сценариев, запускаемых при наступлении событий.

  Дополнительную информацию о событиях, определяемых пользователем, и о файле правил смотрите в разделе “Слежение за событиями HACMP ES и события, определяемые пользователем” на стр. 249.
- Утилиты клиента HACMP для обнаружения изменений статуса и слежения за ними (в одном или нескольких кластерах) из физических узлов AIX, находящихся вне кластера HACMP.

Узлы в кластерах HACMP ES обмениваются сообщениями, называемыми *сигналами работоспособности* или *пакетами активности*, при помощи которых каждый узел сообщает остальным узлам о своей доступности. Если узел перестает отвечать, остальные узлы в кластере инициируют восстановление. Это называется *событием node_down (узел выключен)*; последующий процесс называют также *восстановлением после отказа*. После завершения процесса восстановления производится реинтеграция узла в кластер. Это называется *событием node_up (узел включен)*.

Существует два типа событий: стандартные события, которые ожидаются в рамках операций HACMP ES, и события, определяемые пользователем, которые связаны с наблюдением за параметрами аппаратных и программных компонентов.

Одно из стандартных событий - событие node_down (узел выключен). При планировании действий, необходимых в процессе восстановления, HACMP допускает два варианта восстановления после отказа: горячее резервирование и взаимную подмену.

Конфигурация кластера

В конфигурации *горячего резервирования* узел процессора AIX, который берет на себя функции отказавшего узла, *не* выполняет других работ. В конфигурации *взаимной подмены* узел процессора AIX, который берет на себя функции отказавшего узла, *выполняет* и другие работы.

В общем случае DB2 Universal Database Enterprise - Extended Edition (UDB EEE) запускается в режиме взаимной подмены с разделами на каждом узле. Исключение составляет сценарий, в котором узел каталога является частью конфигурации горячего резервирования.

При планировании крупной установки DB2 в среде RS/6000 SP, использующей HACMP ES, следует решить, разместить ли узлы кластера в пределах одной стойки RS/6000 SP или разделить их между несколькими стойками. Если узел и его резервная копия будут находиться на разных стойках SP, возможна передача нагрузки в случае, когда одна из стоек выходит из строя (то есть отказывает питание стойки или коммутатор). Однако такие отказы должны быть крайне редки, поскольку у каждой стойки есть *N*+1 источник питания, а каждый коммутатор SP, кроме *N*+1 вентилятора и источника питания, имеет резервные пути. При отказе стойки может потребоваться ручное вмешательство для восстановления остальных стоек. Процедура восстановления описана в справочнике SP Administration Guide. HACMP ES обеспечивает восстановление при сбоях узлов SP; восстановление стойки зависит от правильного распределения кластеров в пределах одной или нескольких стоек SP.

При планировании нужно также подумать, как вы будете работать с большими кластерами. Маленьким кластером управлять легче, чем большим; с другой стороны, легче управлять одним большим кластером, чем множеством маленьких. При планировании следует учесть особенности использования прикладных программ в кластерной среде. Если на 16 узлах выполняется одна большая однородная прикладная программа, вероятно, работать с конфигурацией единого кластера будет проще, чем с восемью кластерами по 2 узла. Если же эти 16 узлов содержат много разных прикладных программ с различными сетями, дисками и связями между узлами, вероятно, будет удобнее сгруппировать узлы в меньшие кластеры. Помните, что узлы интегрируются в

кластер HACMP по одному; запуск конфигурации из нескольких кластеров, будет быстрее, чем запуск одного большого кластера. HACMP ES поддерживает как один, так и несколько кластеров, но узел и его резервная копия должны находиться в одном и том же кластере.

В HACMP ES восстановление после отказа поддерживает заранее заданное (другое название - *каскадное*) назначение группы ресурсов для физического узла. Процедура восстановления после отказа допускает также динамическое (или *карусельное*) определение группы ресурсов для физического узла. IP-адреса и группы внешних томов дисков, или файловых систем, или файловые системы NFS и серверы прикладных программ в пределах каждой группы ресурсов задают либо прикладную программу, либо компонент прикладной программы, которыми HACMP ES может управлять на нескольких физических узлах при отработке отказа и реинтеграции. Стратегия при восстановлении после отказа и реинтеграции задается типом созданной группы ресурсов и числом узлов, помещенных в группу ресурсов.

Рассмотрим, например, раздел базы данных DB2 (логический узел). Если его журнал и контейнеры табличных пространств расположены на внешних дисках и с этими дисками связаны другие узлы, эти другие узлы, имея доступ к этим дискам, смогут перезапустить раздел базы данных (на подменяющем узле). Эти действия автоматически выполняются HACMP. HACMP ES также позволяет восстанавливать файловые системы NFS, используемые каталогами главного пользователя экземпляра DB2.

Планируя восстановление при помощи DB2 UDB EEE, внимательно прочитайте документацию по HACMP ES. Нужно прочитать руководства по основным понятиям, планированию, установке и управлению, а затем построить архитектуру восстановления для вашей среды. Для каждой подсистемы, для которой вы задаете восстановление, с учетом известных точек отказов укажите нужные кластеры HACMP, а также узлы восстановления (горячего резервирования или взаимной подмены). Это исходный пункт для заполнения рабочих листов HACMP, приводимых в этой документации.

Настоятельно рекомендуется создать на внешних дисках конфигурации зеркальные копии дисков и адаптеров. Для физических узлов DB2, сконфигурированных для HACMP, следует позаботиться о том, чтобы узлы в группе томов могли переключаться между совместно используемыми внешними дисками. В конфигурации взаимной подмены такая организация требует дополнительного планирования, чтобы парные узлы имели доступ к группам томов друг друга без конфликтов. Для DB2 UDB EEE это означает, что все имена контейнеров должны быть уникальны среди всех баз данных.

Один из способов обеспечить уникальность - включить в состав имени номер раздела. Можно при создании контейнеров SMS или DMS в синтаксис строки контейнера включить выражение, зависящее от узла. Задавая это выражение,

можно включить в имя контейнера номер узла или, если заданы дополнительные аргументы, - результаты вычислений с этими аргументами. Для задания выражения, зависящего от узла, используется аргумент ″ $N″ ([пробел]$N). Аргумент должен находиться в конце строки контейнера и может использоваться только в одной из следующих форм:

Таблица 24. Аргументы для создания контейнеров. Предполагается, что номер узла - пять.

Синтаксис	Пример	Значение
[пробел]$N	″ $N″	5
[пробел]$N+[число]	″ $N+1011″	1016
[пробел]$N%[число]	″ $N%3″	2
[пробел]$N+[число]%[число]	″ $N+12%13″	4
[пробел]$N%[число]+[число]	″ $N%3+20″	22
Примечания: 1. % - остаток от деления. 2. Во всех случаях вычисления выполняются слева направо.		

Примеры создания контейнеров при помощи этого аргумента:

- Создание контейнеров для использования в системе с двумя узлами.

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE USING
   (device '/dev/rcont $N' 20000)
```

  Будут использованы следующие контейнеры:

```
/dev/rcont0   - на Узле 0
/dev/rcont1   - на Узле 1
```

- Создание контейнеров для использования в системе с четырьмя узлами.

```
CREATE TABLESPACE TS2 MANAGED BY DATABASE USING
   (file '/DB2/containers/TS2/container $N+100' 10000)
```

  Будут использованы следующие контейнеры:

```
/DB2/containers/TS2/container100   - на Узле 0
/DB2/containers/TS2/container101   - на Узле 1
/DB2/containers/TS2/container102   - на Узле 2
/DB2/containers/TS2/container103   - на Узле 3
```

- Создание контейнеров для использования в системе с двумя узлами.

```
CREATE TABLESPACE TS3 MANAGED BY SYSTEM USING
   ('/TS3/cont $N%2, '/TS3/cont $N%2+2')
```

  Будут использованы следующие контейнеры:

```
/TS3/cont0   - на Узле 0
/TS3/cont2   - на Узле 0
/TS3/cont1   - на Узле 1
/TS3/cont3   - на Узле 1
```

На рис. 46 и рис. 47 на стр. 234 приведен пример конфигурации подсистемы ввода/вывода DB2 SSA и показано планирование, необходимое для обеспечения высокопроизводительного доступа к внешним дискам и отсутствия конфликтов при обращении ко всем группам томов.

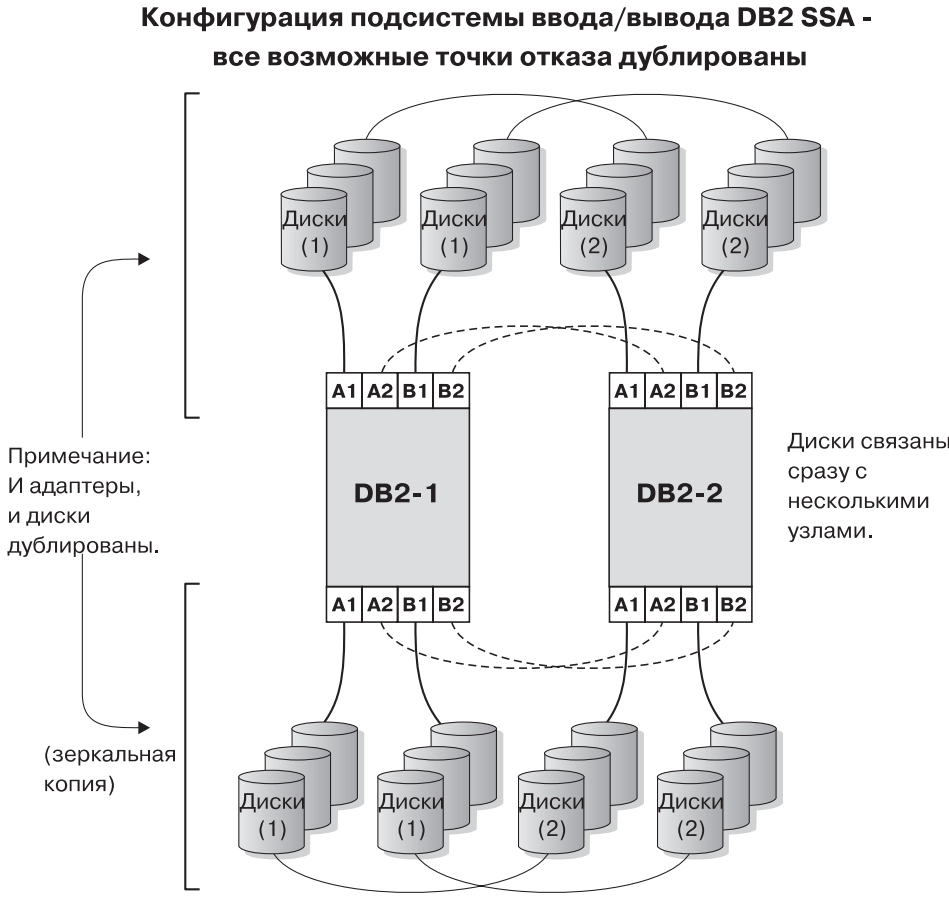


Рисунок 46. Конфигурация, устойчивая к отказу в любой отдельной точке

Конфигурация подсистемы ввода/вывода DB2 SSA - Схема группы томов и логических томов

тестовые данные базы данных db2 на файловой системе/экземпляре под именем powertp

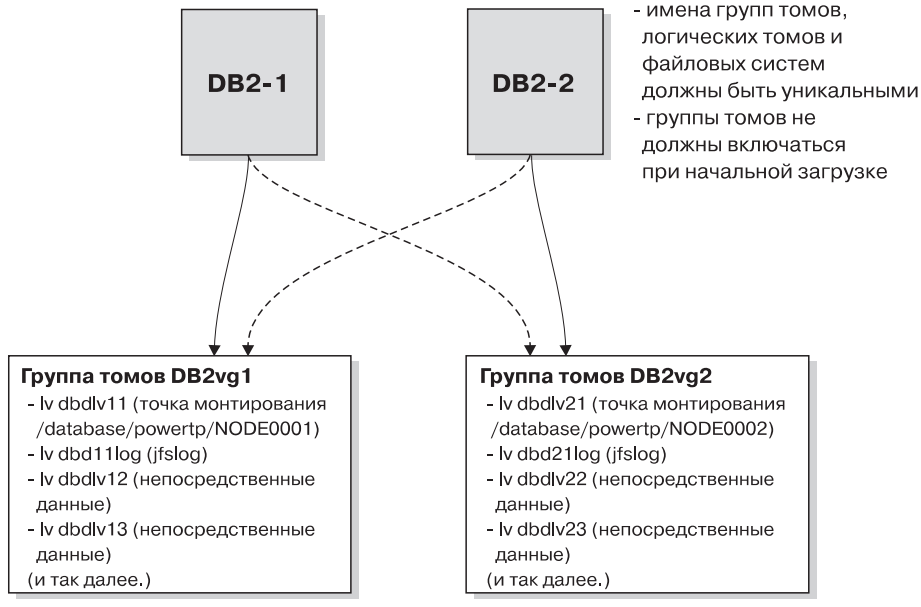


Рисунок 47. Задание группы томов и логического тома

Конфигурирование раздела базы данных DB2

Сконфигурированный HACMP ES поочередно (по одному физическому узлу) запускает все разделы базы данных экземпляра. Если в конфигурации больше четырех узлов, рекомендуется для параллельного запуска DB2 сконфигурировать несколько кластеров. Обратите внимание на то, что в параллельной конфигурации DB2 с 64 узлами быстрее запустить 32 кластера HACMP по 2 узла, чем четыре по 16 узлов.

DB2 UDB EEE создает (и устанавливает на каждом узле в `/usr/bin`) пакет файла сценария `rc.db2pe` для помощи в конфигурировании восстановления после отказа для HACMP ES на узлах горячего резервирования или взаимной подмены. Кроме того, размеры пула буферов DB2 могут настраиваться во время восстановления после отказа в конфигурациях взаимной подмены из `rc.db2pe`. (Когда два раздела базы данных работают на одном физическом узле, требуется настроить размеры пула буферов для обеспечения приемлемой производительности.)

Создавая сервер прикладных программ в конфигурации HACMP раздела базы данных DB2, задайте rc.db2pe как сценарий запуска и остановки:

```
/usr/bin/rc.db2pe <экз> <номер_раздела> <вторич_номер> start <коммутатор>
/usr/bin/rc.db2pe <экз> <номер_раздела> <вторич_номер> stop <коммутатор>
```

где:

```
<экз> - имя экземпляра.
<номер_раздела> - номер раздела базы данных.
<вторич_номер> в  конфигурациях взаимной подмены -
   номер "компаньона" раздела базы данных; в конфигурациях горячего
   резервирования совпадает с <номером_раздела>.
<коммутатор> - обычно пустая строка, что задает
   использование сети коммутатора SP в качестве поля хоста hostname
   в файле db2nodes.cfg (весь трафик DB2 маршрутизируется через коммутатор
   SP); если эта строка не пуста, она используется как имя хоста узла SP.
```

Команда DB2 LIST DATABASE DIRECTORY включена в rc.db2pe, чтобы найти все базы данных, сконфигурированные для этого раздела баз данных. Затем файл сценария ищет файлы /usr/bin/reg.parms.*БАЗА* и /usr/bin/failover.parms.*БАЗА*, где *БАЗА* - каждая из баз данных, сконфигурированных для этого раздела баз данных. В конфигурации взаимной подмены рекомендуется создать файлы параметров reg.parms.*xxx* и failover.parms.*xxx*. В файле параметров failover.parms.*xxx* значения BUFFPAGE, DBHEAP и любые другие, влияющие на пулы буферов, нужно настроить с учетом возможности существования нескольких пулов буферов. Можно использовать предлагаемые примеры файлов параметров reg.parms.SAMPLE и failover.parms.SAMPLE.

Один из важных параметров в этой среде - параметр конфигурации менеджера баз данных *start_stop_time* (время запуска и остановки), его значение по умолчанию - 10 минут. Однако rc.db2pe задает для этого параметра значение 2 минуты. Надо задать в rc.db2pe для этого параметра значение 10 минут или несколько больше. В этом контексте заданный интервал - время между отказом раздела и его восстановлением. Если прикладная программа, выполняемая в разделе, часто выполняет принятие, 10 минут после отказа на разделе базы данных должно хватить для отката непринятых транзакций и возврата к точке согласованности в базе данных на этом разделе. При большой нагрузке или большом числе разделов может понадобиться увеличить этот интервал, чтобы уменьшить вероятность истечения сроков ожидания до завершения операции отката.

Ниже приводится пример конфигурации горячего резервирования и конфигурации взаимной подмены. В обоих примерах группы ресурсов содержат альтернативный адрес коммутатора службы IP. Этот альтернативный адрес коммутатора служит для:

- доступа NFS к файл-серверу для файловых систем владельца экземпляра DB2

- доступа других клиентов, который должен поддерживаться во время восстановления после отказа, соединения TSM (Tivoli Storage Manager, прежнее название - ADSM) и других подобных операций.

Если в данной реализации эти алиасы не требуются, их можно удалить. В случае удаления обязательно задайте для параметра *MOUNT_NFS* в файле сценария `rc.db2pe` значение `NO` .

Пример конфигурации горячего резервирования

В этом примере показана конфигурация горячего резервирования для физических узлов 1 и 2, а экземпляр DB2 называется POWERTP. Раздел базы данных - раздел 1, а база данных называется TESTDATA и находится в файловой системе `/database`.

```
Resource group name: db2_dp_1
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_switch_1  (<<< это альтернативный адрес коммутатора)
Filesystems: /database/powertp/NODE0001
Volume Groups: DB2vg1
Application Servers: db2_dp1_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 1 1 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 1 1 stop
```

Пример конфигурации взаимной подмены

В этом примере показана конфигурация взаимной подмены для физических узлов 1 и 2, а экземпляр DB2 называется POWERTP. Разделы базы данных - разделы 1 и 2, а база данных называется TESTDATA и находится на файловой системе `/database`.

```
Resource group name: db2_dp_1
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_switch_1  (<<< это альтернативный адрес коммутатора)
Filesystems: /database/powertp/NODE0001
Volume Groups: DB2vg1
Application Servers: db2_dp1_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 1 2 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 1 2 stop

Resource group name: db2_pd_2
Node Relationship: cascading
Participating nodenames: node2_eth, node1_eth
Service_IP_label: nfs_switch_2  (<<<  это альтернативный адрес коммутатора)
Filesystems: /database/powertp/NODE0002
Volume Groups: DB2vg2
Application Servers: db2_dp2_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 2 1 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 2 1 stop
```

Конфигурация узла сервера NFS

Сценарий `rc.db2pe` может также использоваться для обеспечения доступа к каталогам, смонтированным NFS в пользовательских каталогах параллельного

экземпляра DB2. Для этого в файле сценария `rc.db2pe` для параметра *MOUNT_NFS* задается значение `YES` и конфигурируется пара серверов NFS для восстановления после отказов:

- Сконфигурируйте домашний каталог и экспортируйте его в качестве ″корневого″ при помощи `/etc/exports` и команды **exportfs** по IP-адресу, используемому на узлах той же подсети, что и IP-адрес сервера NFS. Включите и адрес загрузки HACMP, и адрес служб. IP-адрес сервера NFS совпадает с адресом служб в HACMP и может быть передан на резервный сервер. Домашний каталог владельца экземпляра DB2 должен быть смонтирован NFS явно, а не автоматически. (Сценарии не поддерживают использование автоматического монтирования для домашнего каталога владельца экземпляра DB2.)
- При помощи SMIT или конфигурирования через командную строку создайте отдельную запись `/etc/filesystems` для этой файловой системы, чтобы все узлы в параллельной группировке DB2, включая файл-сервер, могли монтироваться при помощи команды файловой системы NFS.

  Например, файловую систему JFS `/nfshome` можно экспортировать на все узлы как `/dbhome`. Каждый узел создает файловую систему NFS `/dbname` в виде `nfs_server:/nfshome`. Следовательно, для экземпляра ″powertp″ домашним каталогом владельца экземпляра DB2 будет `/dbhome/powertp`.

  Для монтирования в `/etc/filesystems` надо использовать параметры ″hard″, ″bg″, ″intr″ и ″rw″.
- Определения владельца экземпляра DB2, связанные с домашним каталогом `/dbhome/powertp` в `/etc/passwd`, должны быть одинаковы на всех узлах.

  Пользовательские определения в среде SP в типичном случае создаются на управляющей рабочей станции, после чего при помощи ″supper″ и ″pcp″ `/etc/passwd`, `/etc/security/passwd`, `/etc/security/user` и `/etc/security/group` распространяются на все узлы.
- *Не* конфигурируйте ″nfs_filesystems to export″ в группах ресурсов HACMP для экспортируемой группы томов и файловой системы. Вместо этого сконфигурируйте ее для NFS обычным образом. Экспортом этих файловых систем будут управлять сценарии для сервера NFS.
- Главный номер группы томов, где находится файловая система, должен совпадать на первичном узле и узле передачи. Это достигается использованием команды **importvg** с опцией `-V`.
- Убедитесь, что для параметра *MOUNT_NFS* в файле сценария `rc.db2pe` задано значение `YES` и что у каждого узла есть файловая система NFS для монтирования в `/etc/filesystems`. Если это не так, `rc.db2pe` не сможет смонтировать файловую систему и запустить DB2.
- Если владелец экземпляра DB2 уже создан и вы копируете структуру каталога пользователя в создаваемую файловую систему, команду **tar** для этого каталога надо выполнить с опциями (`-cvf`). Это обеспечит сохранение символических связей.

- Не забудьте для создаваемой файловой системы сделать зеркальные копии адаптеров и дисков для логических томов, а также журналов файловой системы.

Пример конфигурации с подменой серверов NFS

В этом примере предполагается, что файловая система сервера NFS /nfshome находится в группе томов nfsvg по IP-адресу ″nfs_server″. Экземпляр DB2 называется POWERTP, а домашний каталог - /dbhome/powertp.

```
Resource group name: nfs_server
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_server     (<<< это альтернативный адрес коммутатора)
Filesystems: /nfshome
Volume Groups: nfsvg
Application Servers: nfs_server_app
Application Server Start Script: /usr/bin/rc.db2pe powertp NFS SERVER start
Application Server Stop Script: /usr/bin/rc.db2pe powertp NFS SERVER stop
```

В этом примере:

- /etc/filesystems на всех узлах будет содержать запись для монтирования /dbhome в точке nfs_server:/nfshome. nfs_server - альтернативный адрес IP-службы коммутатора.
- /etc/exports на узле nfs_server и на резервном узле будут включать адреса загрузки и службы и содержать запись для /nfsfs -root=nfs_switch_1, nfs_switch_2, ...

Особенности конфигурирования коммутатора SP

Реализуя HACMP ES с коммутатором SP, имейте в виду, что:

- На коммутаторе SP есть ″исходные″ адреса и ″альтернативные″ адреса. Исходные адреса - это те, что заданы в SDR SP (System Data Repository - хранилище системных данных) и сконфигурированы rc.switch при начальной загрузке системы. ″Альтернативные″ адреса - это IP-адреса, сконфигурированные в дополнение к исходным адресам в интерфейсе css0 при помощи команды **ifconfig** с атрибутом alias. Например:

  ```
  ifconfig css0 inet alias sw_alias_1 up
  ```

- При конфигурировании файла DB2 db2nodes.cfg ″исходные″ IP-адреса коммутатора SP следует использовать для полей ″hostname″ и ″netname″. ″Альтернативные″ IP-адреса коммутатора используются *только* для поддержания соединений NFS. Восстановление после отказов DB2 достигается перезапуском DB2 при помощи команды **db2start** (RESTART), которая изменяет файл db2nodes.cfg.
- Не следует путать адреса коммутатора с алиасами etc/hosts. И исходные, и альтернативные адреса коммутатора SP - реальные адреса в etc/hosts или DNS. Альтернативный адрес коммутатора - это не другое имя того же адреса коммутатора SP; это просто другой адрес.

- Исходные адреса коммутатора SP всегда присутствуют на узле, когда он в рабочем состоянии. HACMP ES не конфигурирует эти адреса и не перемещает их между узлами.
- Если вы намерены использовать альтернативные адреса коммутатора SP, сконфигурируйте их в HACMP как адреса загрузки и служб для сигналов работоспособности и подмены IP-адресов. Если вы не хотите использовать альтернативные адреса коммутатора SP, сконфигурируйте исходные адреса коммутатора SP в HACMP как адрес служб для использования *только* для сигнала работоспособности (но не для подмены IP-адреса). В любой конфигурации не конфигурируйте одновременно альтернативные адреса *и* исходный адрес коммутатора; такая конфигурация не поддерживается HACMP ES.
- Только альтернативные адреса коммутатора SP (а не исходные адреса коммутатора SP) перемещаются между узлами для конфигурации подмены IP.
- Необходимость в альтернативных адресах коммутатора SP возникает из-за того, что допускается лишь один адаптер коммутатора SP на узел. Использование альтернативных адресов позволяет передать узлу альтернативный адрес вместо IP-адреса коммутатора другого узла, не добавляя еще один адаптер коммутатора. Это полезно на узлах с ограниченным числом гнезд. Дополнительную информацию о восстановлении после сбоев адаптера коммутатора SP смотрите в разделе об отказах сети на “Файлы сценариев HACMP ES” на стр. 253.
- Если вы сконфигурируете коммутатор SP для подмены IP-адреса, нужно будет создать по два лишних алиаса IP-адресов на узел: один как адрес загрузки и один как адрес службы.
- Не забудьте использовать ″HPS″ в определении сетевого имени HACMP ES для исходного IP-адреса коммутатора SP или альтернативного IP-адреса коммутатора SP.
- `rc.cluster` в HACMP автоматически вставляет **ifconfig** в адрес загрузки коммутатора SP при запуске HACMP. Не требуется никакого дополнительного конфигурирования, кроме создания IP-адреса и имени и задания их в HACMP.
- Узел Eprimary коммутатора SP - это сервер, реализующий команды Estart, Efence и Eunfence. Сценарии HACMP пытаются выполнить команды Eunfence или Estart на узле при запуске HACMP и сделать коммутатор доступным, если он определен как одна из ее сетей. Поэтому узел Eprimary должен быть доступен при запуске HACMP. Программа HACMP ожидает до 12 минут, пока завершится отработка отказа Eprimary, прежде чем констатировать ошибку.
- Узел Eprimary коммутатора SP перемещается между узлами при помощи SP Parallel System Support Program (PSSP), а не HACMP. Если узел Eprimary отключается, PSSP автоматически передает функции узла Eprimary резервному узлу. Это изменение не влияет на сеть коммутатора, которая остается в рабочем состоянии.

Примеры конфигурации DB2 HACMP

Следующие примеры иллюстрируют разные конфигурации поддержки восстановления после отказа и показывают, что происходит при сбоях.

Для конфигураций взаимной подмены DB2 HACMP (рис. 48 на стр. 241, рис. 49 на стр. 242 и рис. 50 на стр. 243):

- Адаптеры HACMP определены для сети Ethernet и альтернативных адресов загрузки и служб алиаса коммутатора SP – исходные адреса не используются. Не забывайте в имени сети HACMP использовать строку ″HPS″.
- `NFS_server/nfshome` монтируется как `/dbhome` на всех узлах через алиасы коммутатора.
- Файл `db2nodes.cfg` содержит исходные адреса коммутатора SP. После восстановления после отказа раздела базы данных DB2 (логического узла) команда **db2start** (RESTART) изменяет файл `db2nodes.cfg`.
- Адреса загрузки алиаса коммутатора SP не показаны.
- Узлы могут находиться на разных стойках SP.

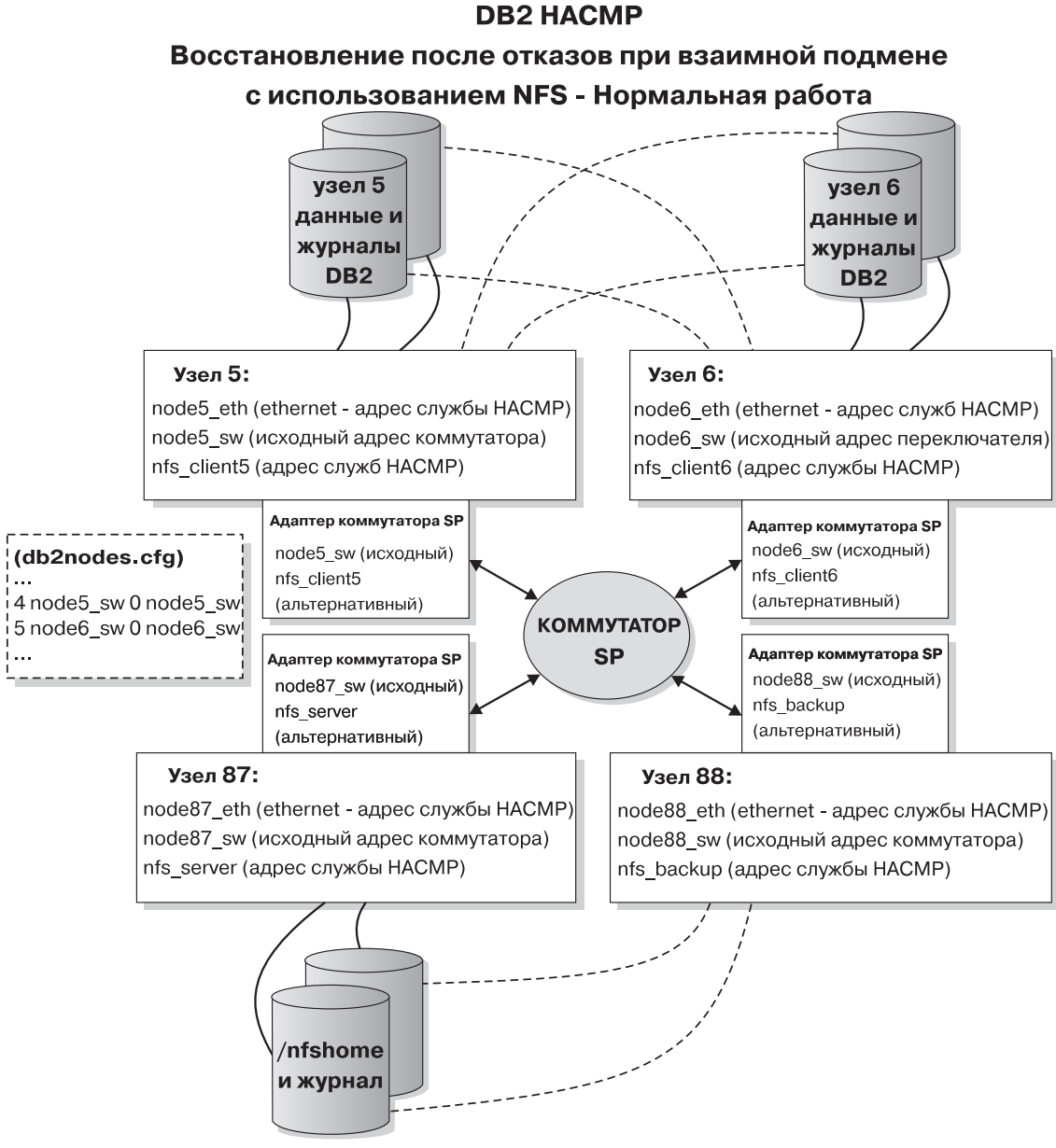


Рисунок 48. Восстановление после отказов при взаимной подмене с использованием NFS - нормальная работа

DB2 HACMP
Восстановление после отказов при взаимной подмене
с использованием NFS - Отказ NFS

- Альтернативный IP-адрес коммутатора SP nfs_server и nfs, монтируемая в точке /nfshome, перемещается с узла 87 на 88.
- Код arp коммутатора SP при этом перемещении может изменить все кэши arp коммутатора.

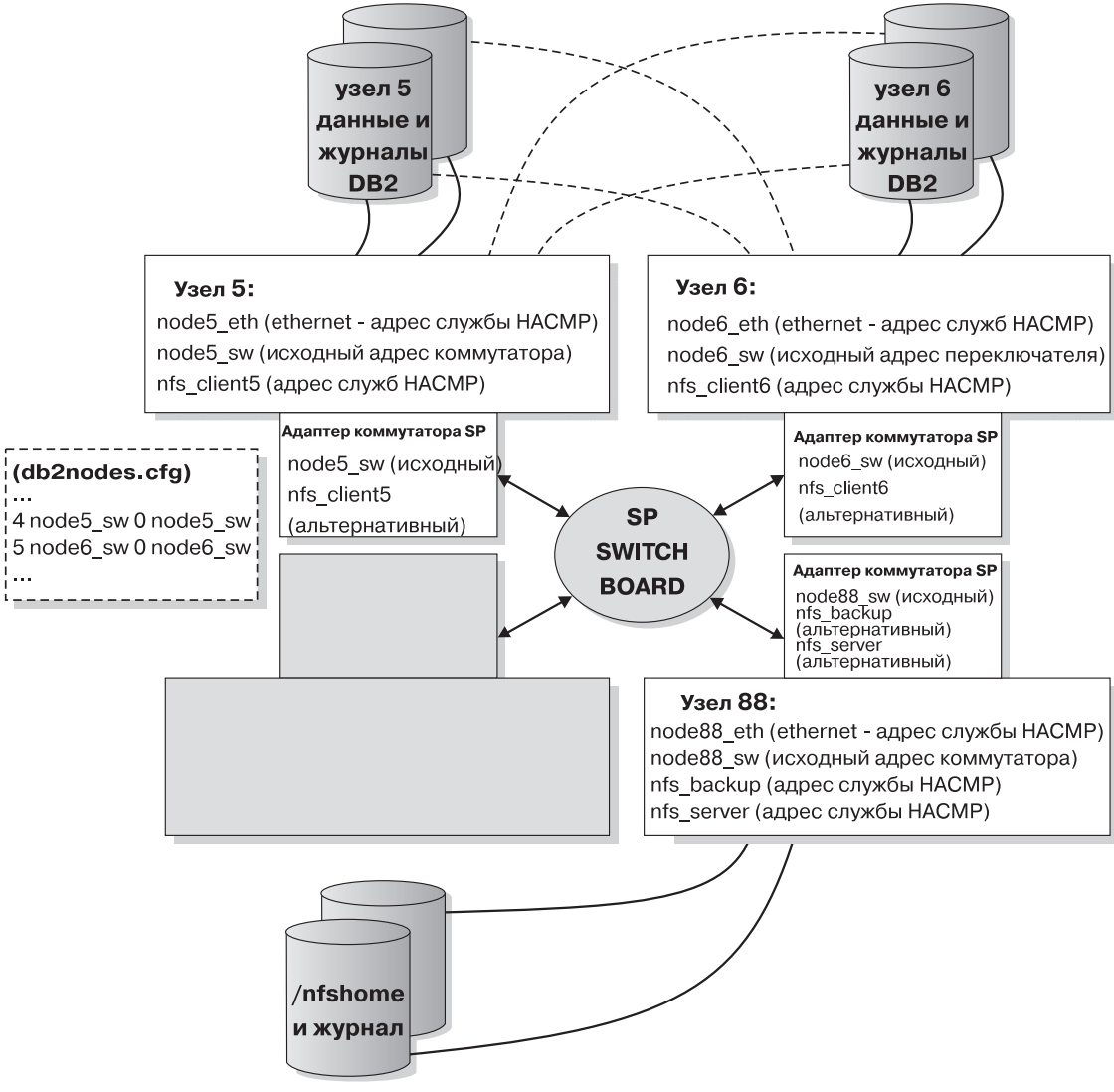


Рисунок 49. Восстановление после отказов при взаимной подмене с использованием NFS - отказ NFS

DB2 HACMP
Восстановление после отказов при взаимной подмене с использованием NFS - Отказ DB2

- подмена IP-адреса коммутатора позволяет другим серверам (типа ADSM) восстановить связь.
- На узле 5 запущены 2 логических узла DB2.

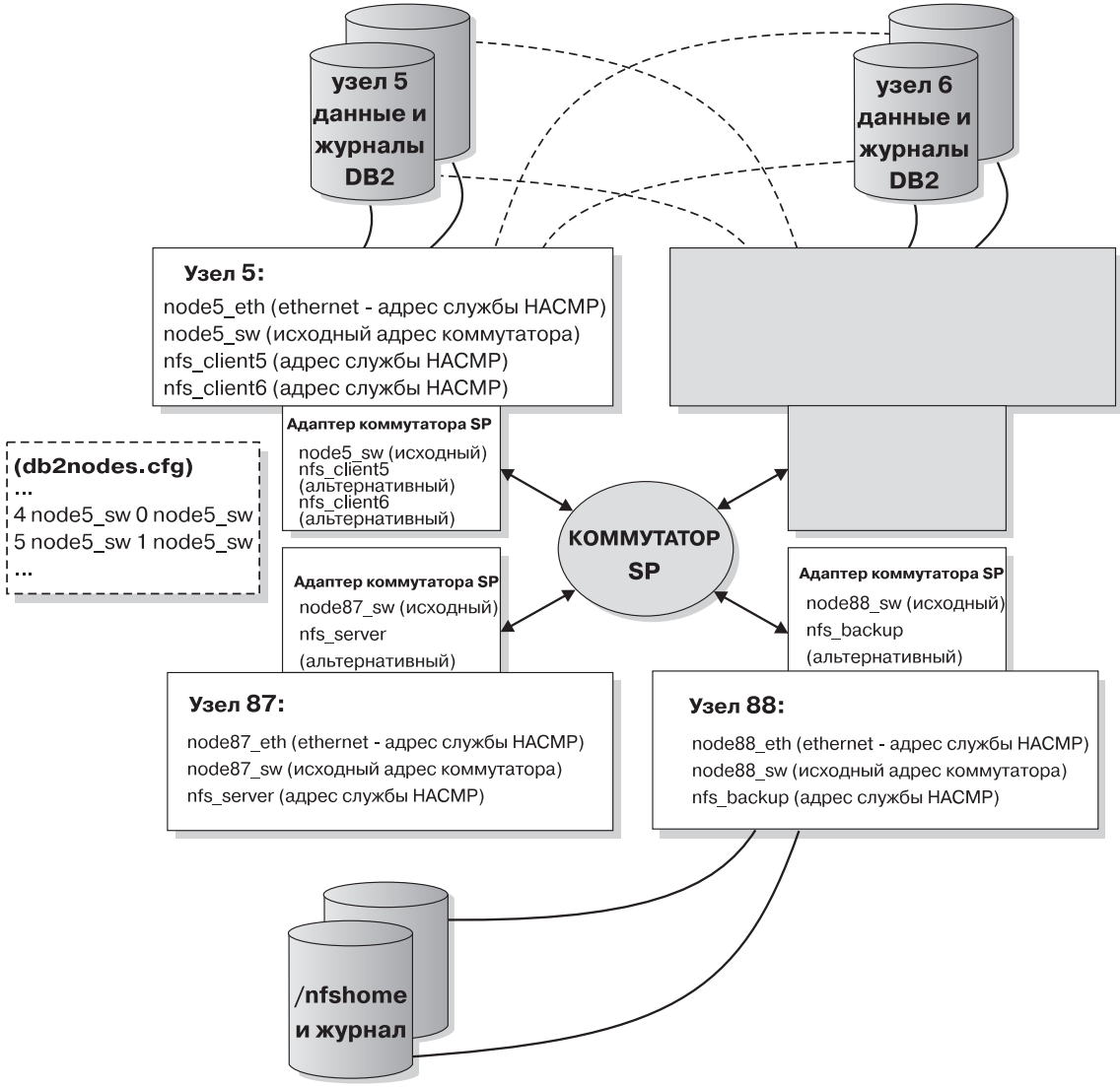


Рисунок 50. Восстановление после отказов при взаимной подмене с использованием NFS - отказ DB2

Для конфигураций горячего резервирования DB2 HACMP (рис. 51 на стр. 245 и рис. 52 на стр. 246):

- Адаптеры HACMP определены для сети Ethernet и альтернативных адресов загрузки и служб алиаса коммутатора SP – исходные адреса не используются. Не забывайте в имени сети HACMP использовать строку ″HPS″.
- `NFS_server/nfshome` монтируется как `/dbhome` на всех узлах через алиасы коммутатора.
- Файл `db2nodes.cfg` содержит исходные адреса коммутатора SP. После восстановления после отказа раздела базы данных DB2 (логического узла) команда **db2start** (RESTART) изменяет файл `db2nodes.cfg`.
- Адреса загрузки алиаса коммутатора SP не показаны.

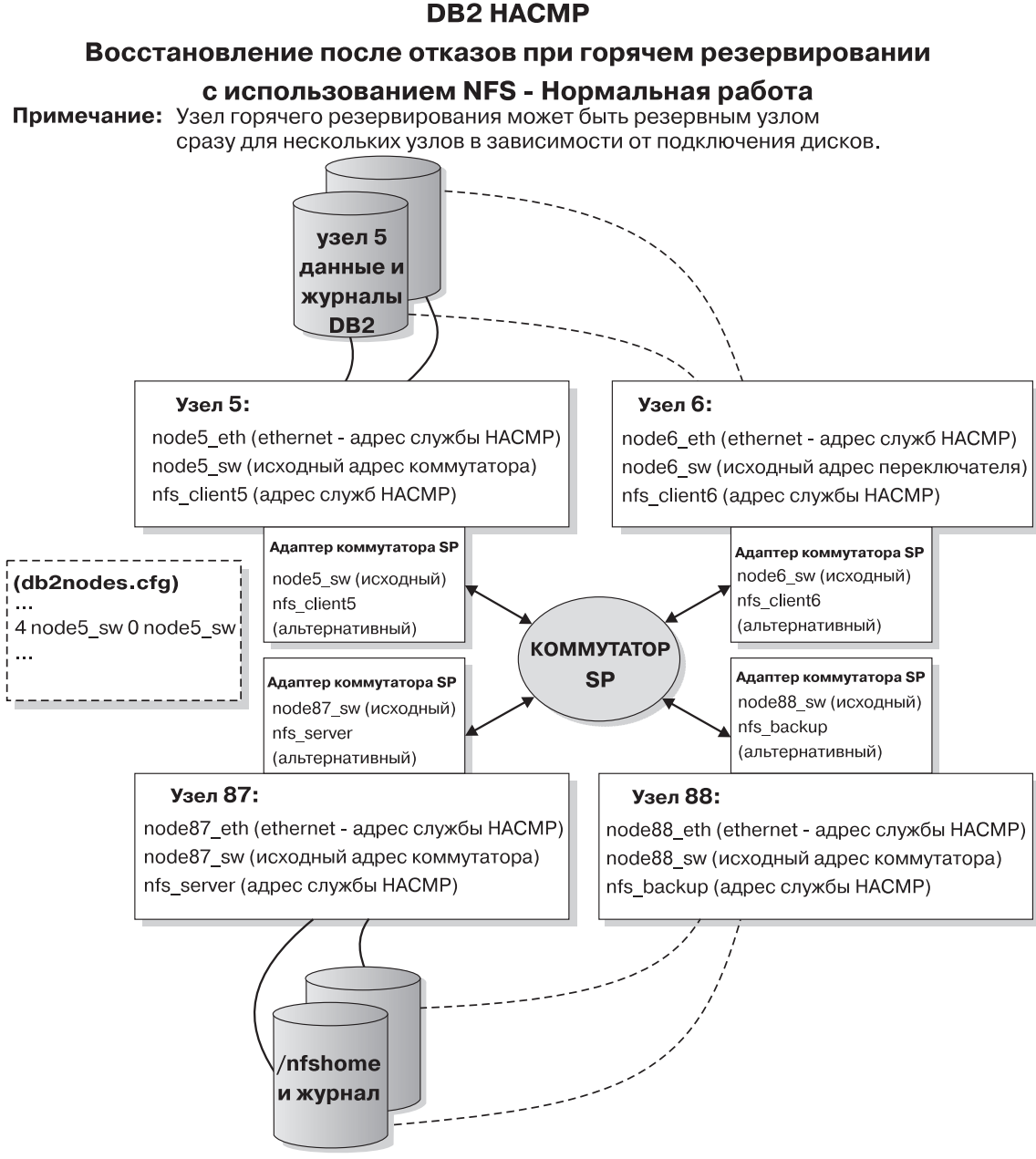


Рисунок 51. Восстановление после отказов при горячем резервировании с использованием NFS - нормальная работа

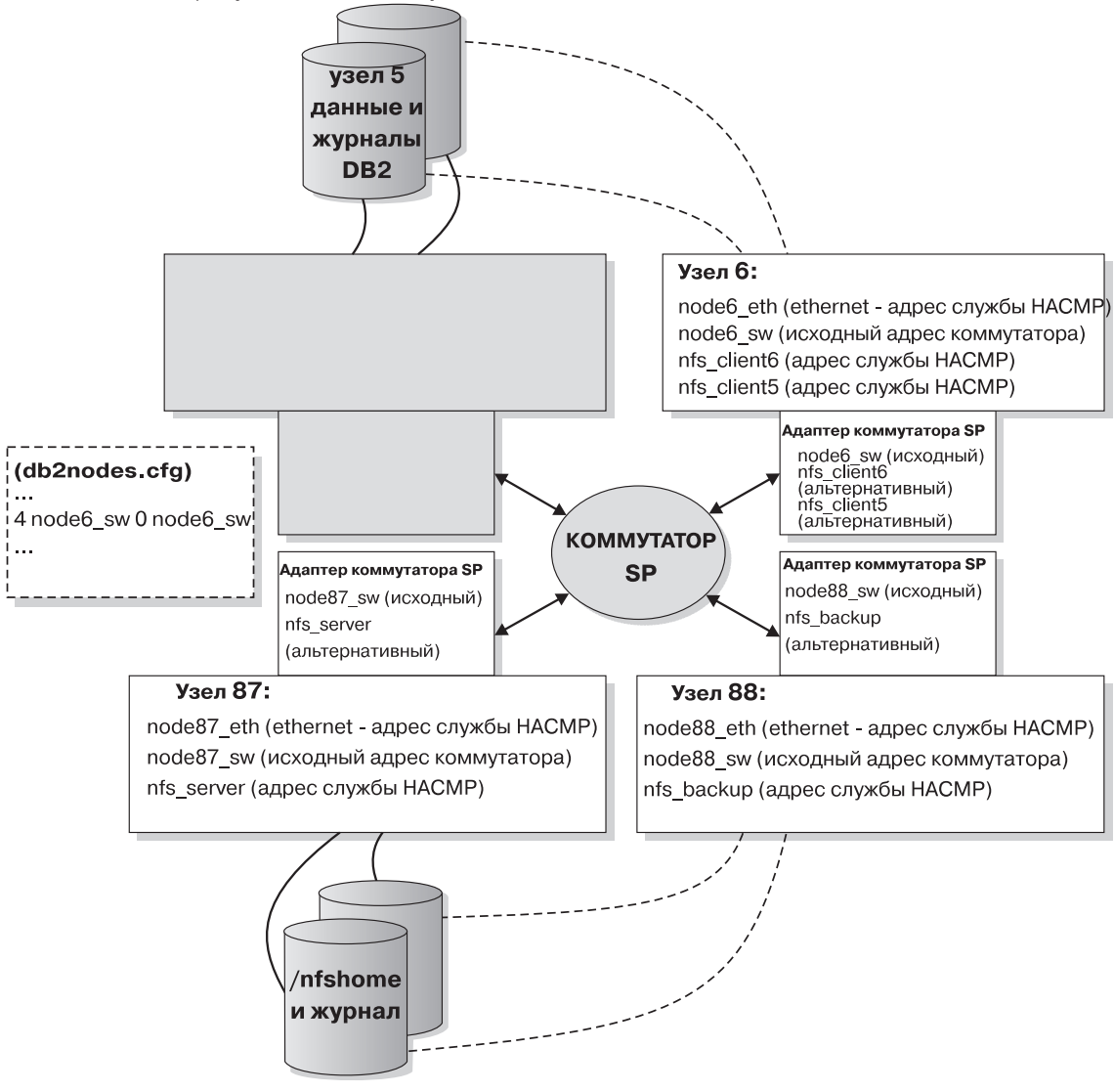


Рисунок 52. Восстановление после отказов при горячем резервировании с использованием NFS - отказ DB2

Для конфигураций взаимной подмены DB2 HACMP без NFS (рис. 53 на стр. 247 и рис. 54 на стр. 248):

- Адаптеры HACMP определены для сети ethernet и исходных адресов SP. Помните, что когда исходные адреса конфигурируются в HACMP как адреса службы, адрес загрузки не используется (только сигналы работоспособности). Не забывайте в имени сети HACMP для коммутатора SP использовать строку ″HPS″.
- Файл `db2nodes.cfg` содержит исходные адреса коммутатора SP. После восстановления после отказа раздела базы данных DB2 (логического узла) команда **db2start** (RESTART) изменяет файл `db2nodes.cfg`.
- Функции восстановления после отказа NFS не показаны.
- Узлы могут находиться на разных стойках SP.

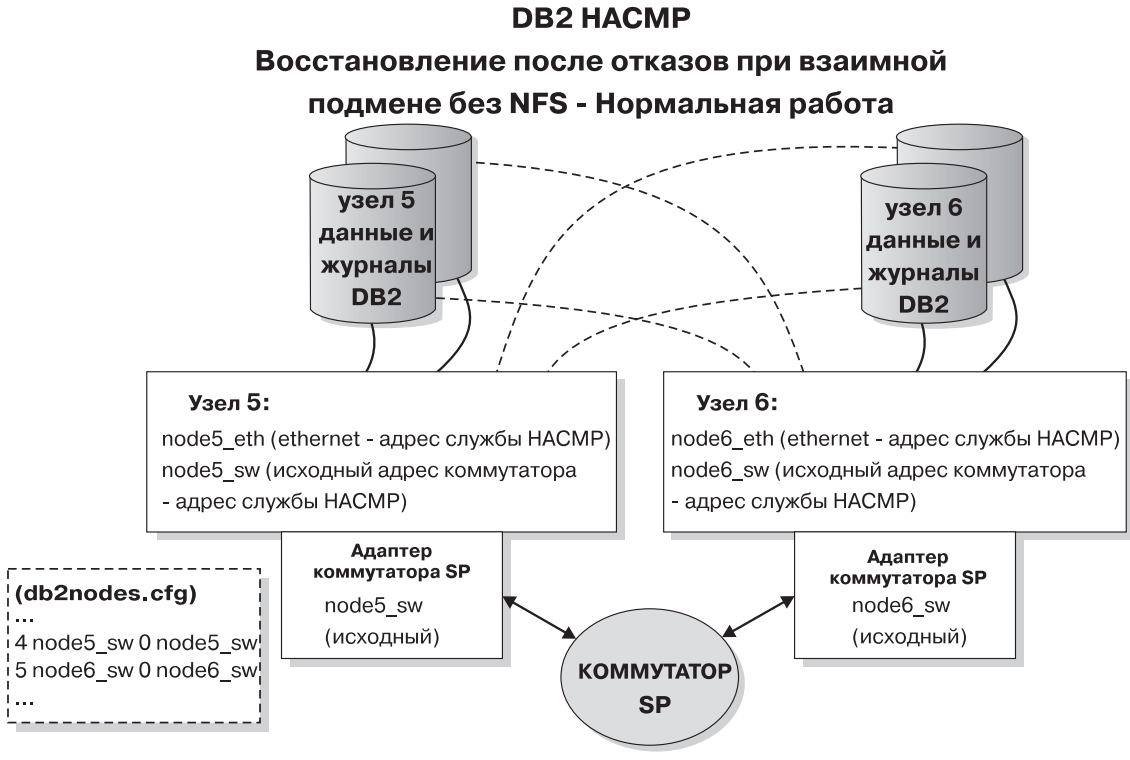


Рисунок 53. Восстановление после отказов при взаимной подмене без NFS - нормальная работа

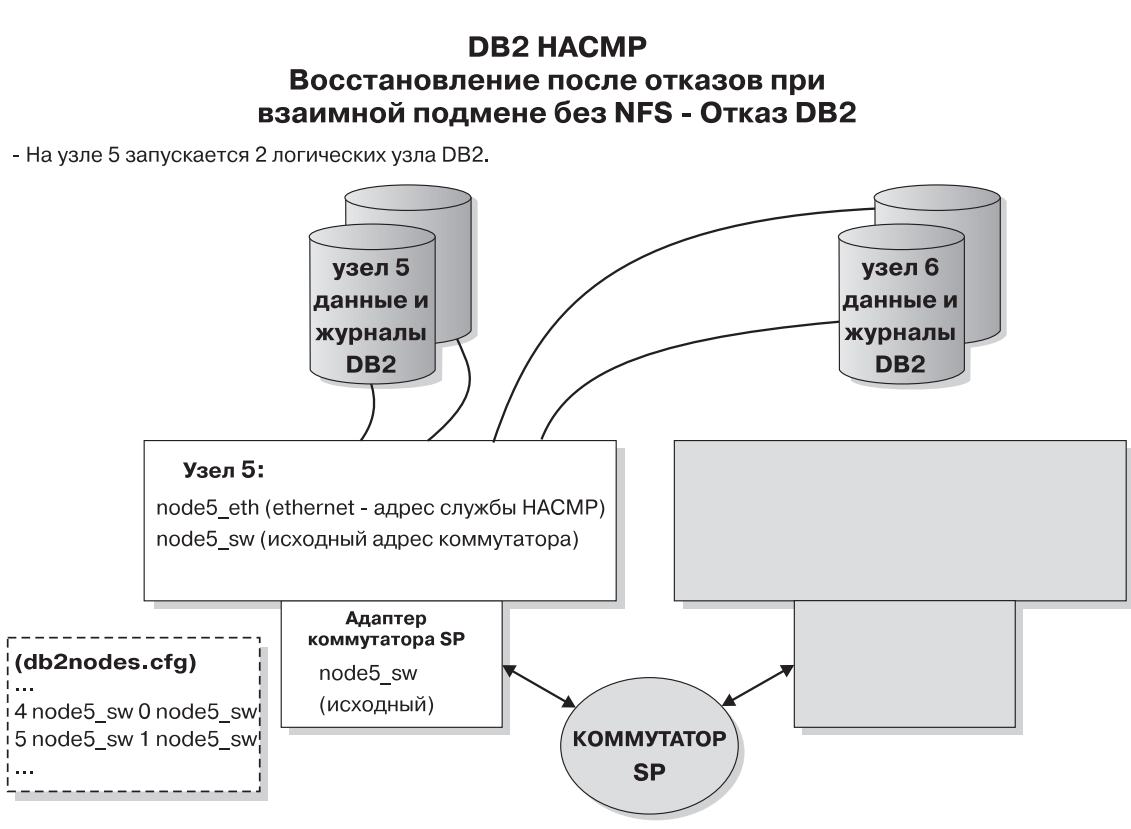


Рисунок 54. Восстановление после отказов при взаимной подмене без NFS - отказ DB2

Рекомендации по запуску DB2 HACMP

Не рекомендуется задавать запуск HACMP при загрузке в `/etc/inittab`. Следует запускать HACMP вручную после загрузки узлов. При этом возможно обслуживание отказавших узлов без прекращения работы.

В качестве примера "ремонта с остановкой" рассмотрим случай, в котором на узле происходит сбой из-за аварии аппаратуры. HACMP автоматически выполняет восстановление после отказа, восстановление происходит успешно. Однако аварийный узел нуждается в ремонте. Если в `/etc/inittab` HACMP предписывается начинать перезагрузку, после загрузки будет предпринята попытка реинтеграции этого узла, что в данном случае нежелательно.

Как пример "ремонта без остановки" рассмотрим ручной запуск HACMP на каждом узле. Здесь сбойные узлы можно отремонтировать и реинтегрировать, не затрагивая остальные узлы. Сценарий `ha_cmd` предназначен для управления командами запуска и остановки HACMP с управляющей рабочей станции.

Примечание: При первом создании экземпляра DB2 к файлу /etc/inittab добавляется запись:

```
rcdb2:2:once:/etc/rc.db2 > /dev/console 2>&1 # Autostart
DB2 Services
```

Если включены HACMP или HACMP ES, отредактируйте файл /etc/inittab, поставив эту строку перед строкой HACMP. Ниже приведен пример строки HACMP в файле /etc/inittab:

```
clinit:a:wait:touch /usr/sbin/cluster/.telinit # HACMP for AIX
```

Эта запись должна быть последней в файле /etc/inittab.

Слежение за событиями HACMP ES и события, определяемые пользователем

Закрытие разделов базы данных DB2 на физическом узле AIX, когда пространство подкачки заполняется на определенный процент, и перезапуск раздела базы данных DB2 или инициирование процедуры восстановления после сбоев, если процесс на данном узле прекратил работу, - два примера событий, определяемых пользователем. Примеры, которые иллюстрируют события, определяемые пользователем, такие как закрытие раздела базы данных и отмена транзакции для освобождения пространства подкачки, можно найти в подкаталоге samples.

Файл правил /user/sbin/cluster/events/rules.hacmprd содержит события HACMP. Каждое описание события в этом файле состоит из девяти компонентов:

- Уникальное имя события.
- Статус или спецификатор события. Имя события и статус - определяющие атрибуты правила. Менеджер кластера HACMP ES инициирует восстановление, только если находит правило с именем и статусом, соответствующим имени и статусу события.
- Путь программы ресурсов - полный путь файла *xxx*.rp, содержащего программу восстановления.
- Тип восстановления. Этот компонент зарезервирован для будущего использования.
- Уровень восстановления. Этот компонент зарезервирован для будущего использования.
- Имя переменной ресурсов, используемое для событий менеджера событий.
- Вектор экземпляра, используемый для событий менеджера событий. Это набор элементов в форме "имя=значение". Значения уникально определяют копию ресурса в системе и, по расширению, копию переменной ресурса.
- Предикат, используемый для событий менеджера событий. Это логическое выражение, сравнивающее переменную ресурса с другими элементами. Когда

это выражение истинно, подсистема менеджера событий генерирует информационное событие для менеджера кластера и соответствующей прикладной программы.

- Предикат реактивации, используемый для событий менеджера событий. Этот предикат служит для генерации события, изменяющего статус первичного предиката. В типичном случае этот предикат - инверсия первичного предиката. Он может также использоваться с предикатом события для задания верхней и нижней границы для рассматриваемого условия.

Каждый объект занимает отдельную строку в определении события, даже если эта строка не используется. Если эти строки удалить, менеджер кластера HACMP ES не сможет правильно проанализировать определение события, что может привести к зависанию системы. Все строки, начинающиеся с "#", считаются комментариями.

Примечание: Файл правил занимает ровно по девять строк на каждое определение события, не считая строк комментариев. При добавлении в конец файла правил события, определяемого пользователем, важно удалить лишние пустые строки в конце файла, иначе узел зависнет.

Пример определения события node_up (узел включен):

```
##### Начало определения события (node_up) узел включен
#
TE_JOIN_NODE
0
/usr/sbin/cluster/events/node_up.rp
2
0
# 6) Переменная ресурса - используется только для событий менеджера
     событий
# 7) Вектор экземпляра - используется только для событий менеджера
     событий
# 8) Предикат - используется только для событий менеджера событий

# 9) Предикат реактивации - используется только для событий менеджера
     событий

###### Конец определения события (node_up) "узел включен"
```

Этот пример - просто одно из определений событий в файле rules.hacmprd. В этом примере в случае наступления события node_up вызывается программа восстановления /usr/sbin/cluster/events/node_up.rp. Заданы значения статуса, типа восстановления и уровня восстановления. Четыре пустых строки соответствуют переменной ресурса, переменной экземпляра, предикату и предикату реактивации.

Можно определить и другие события - реакцию на нестандартные события HACMP ES. Например, чтобы определить такое событие, как заполнение файловой системы `/tmp` более чем на 90 процентов, следует внести изменения в файл `rules.hacmprd`.

Многие события заранее заданы в IBM Parallel System Support Program (PSSP). Эти события можно использовать (в составе событий, определяемых пользователем) так:

1. Остановите кластер.
2. Внесите изменения в файл `rules.hacmprd`. Перед внесением изменений создайте резервную копию. Добавьте заранее заданное событие PSSP вручную. Если нужно синхронизовать точки на всех узлах кластера, используйте в программе восстановления команду **barrier**. (Дополнительные сведения о команде barrier и синхронизации программ восстановления смотрите в руководствах по основным понятиям HACMP, установке и управлению.)
3. Перезапустите кластер. Файл `rules.hacmprd` хранится в памяти во время запуска менеджера кластера. Чтобы гарантировать вступление изменений в силу, перезапустите все кластеры. В кластере не должно быть противоречивых правил.
4. Менеджер кластера использует все события из файла `rules.hacmprd`.

HACMP ES использует детектор событий PSSP для обработки событий, определяемых пользователем. Подсистема менеджера событий PSSP обеспечивает всестороннее обнаружение событий путем наблюдения за различными аппаратными и программными ресурсами.

Состояние ресурсов отражают переменные ресурсов. Условия на ресурсы задают выражения, называемые предикатами.

Менеджер событий получает переменные ресурсов от монитора ресурсов, который наблюдает за состоянием отдельных системных ресурсов и преобразует сведения о состоянии в несколько переменных ресурсов. Эти переменные периодически передаются монитору событий. Монитор событий применяет предикаты, заданные менеджером кластера HACMP ES в `rules.hacmprd`, к каждой переменной ресурсов. Когда предикат оценивается как истинный, событие генерируется и посылается менеджеру кластера. Менеджер кластера инициирует протокол голосования, и запускается (согласно приоритету события) файл программы восстановления (`xxx.rp`) на наборе узлов, заданных как ″наборы узлов″ в программе восстановления.

Файл программы восстановления (*xxx*`.rp`) состоит из одной или нескольких строк программы восстановления. Каждая строка задается в следующем формате:

```
отношение      выполняемая_команда      ожидаемый_статус      NULL
```

Между значениями в строке должен быть хотя бы один пробел. ″Отношение″ - это значение, используемое для выбора запускаемой программы и типа узла. Поддерживаются три типа отношений:

- All. Заданная команда или программа запускается на всех узлах текущего кластера HACMP.
- Event. Заданная команда или программа запускается только на тех узлах, на которых произошло событие.
- Other. Заданная команда или программа запускается на всех узлах, на которых событие *не* произошло.

″Выполняемая_команда″ - это заключенная в кавычки строка, задающая исполняемую программу (с полным путем или без него). Определения с относительным путем можно использовать только для сценариев событий, поставляемых с HACMP. Остальные сценарии или программы должны использовать полный путь, даже если расположены в том же каталоге, что и сценарии событий HACMP.

″Ожидаемый_статус″ - это код возврата данной команды или программы. Это целое значение или ″x″. Если указано ″x″, менеджер кластера не обращает внимания на код возврата. Остальные коды должны быть равны ожидаемому коду возврата; в противном случае менеджер кластера констатирует неудачное завершение события. Обработка этого события ″подвешивает″ процесс до тех пор, пока не произойдет восстановление (из-за ручного вмешательства). Без ручного вмешательства узел не будет синхронизирован с другими узлами. Синхронизация всех узлов необходима, чтобы менеджер кластера управлял всеми узлами.

″NULL″ - поле, зарезервированное для будущего использования. Слово ″NULL″ должно находиться в конце каждой строки, кроме строки barrier. Если задать несколько команд восстановления между двумя командами barrier или до первой такой команды, команды восстановления выполняются параллельно на самом узле и между узлами.

Команда barrier служит для синхронизации всех команд на всех узлах кластера. Когда узел встречает в программе восстановления оператор barrier, менеджер кластера инициирует на этом узле протокол barrier. Поскольку протокол barrier - двухфазный, все узлы уведомляются о завершении обеих фаз, когда все узлы встретили в программе восстановления barrier и ″проголосовали″ за принятие этого протокола.

Кратко процесс можно описать так:

1. Службы группы/ES (для ранее заданных событий) или менеджер событий (для событий, определенных пользователем) уведомляют о событии менеджер кластера HACMP ES.

2. Менеджер кластера читает файл `rules.hacmprd` и определяет программу восстановления, применяемую для этого события.
3. Менеджер кластера запускает программу восстановления, состоящую из последовательности команд восстановления.
4. Программа восстановления выполняет команды восстановления, которые могут быть сценариями оболочки или двоичными командами. (В HACMP для AIX команды восстановления те же, что в сценариях событий HACMP.)
5. Менеджер кластера получает статус возврата от команд восстановления. Статус, отличный от ожидаемого, ″подвешивает″ кластер до ручного вмешательства (при помощи команды `smit cm_rec_aids` или `/usr/sbin/cluster/utilities/clruncmd`).

Файлы сценариев HACMP ES

Приведенные ниже примеры сценариев для восстановления после отказов и событий, определенных пользователем, входят в DB2 UDB EEE. Файлы сценариев находятся в каталоге `$INSTNAME/sqllib/samples/hacmp/es`. Эти сценарии будут работать ″как есть″, но вы можете внести изменения в действия восстановления.

- Сценарий восстановления раздела базы данных DB2 `rc.db2pe`. Это файл сценария, служащий для запуска и остановки конфигурации HACMP на разделе базы данных. Он также работает как сценарий запуска и остановки HACMP для сервера NFS владельца экземпляра DB2.
- Зависящие от DB2 события, определяемые пользователем, для HACMP ES. Включены шесть событий по умолчанию: одно - восстановление процесса, два - для пространства подкачки и три - для восстановления NFS и автоматического монтирования.
- Отработка отказа файл-сервера NFS экземпляра DB2. Этот сценарий обеспечивает восстановление после отказа сервера файловой системы экземпляра DB2 на резервную систему.
- Восстановление после отказа сети. Сценарии `network_up_complete`, `network_back`, `network_down_complete` и `network_down` позволяют выполнить восстановление разделов баз данных SP DB2 после отказа адаптера коммутатора SP.
- Сценарии для задания событий монитора для SP GUI Perspectives. Следить за сбоями и восстановлением, определенным пользователем, можно при помощи средства Event and Hardware Perspectives. Дальнейшие сведения о Perspectives смотрите в документации по PSSP Administration.
- Сценарии установки для установки и удаления основных сценариев и событий на узлах HACMP ES.
- Файлы сценариев для создания и удаления ресурсов менеджера проблем (pman) SP Perspectives для наблюдения за конфигурацией HACMP и DB2.

Сценарии восстановления должны быть установлены на каждом узле, который будет выполнять операции восстановления. Файлы сценариев могут быть централизованно установлены с управляющей рабочей станции SP или другого назначенного узла SP:

1. Скопируйте сценарии из каталога `$INSTNAME/sqllib/samples/hacmp/es` на одну из управляющих рабочих станций SP или другой узел SP, способный выполнять команды **pcp** и **pexec**. Эти команды необходимы для операции установки.
2. Настройте файлы `reg.parms.SAMPLE` и `failover.parms.SAMPLE` для вашей среды, задав ключевые параметры (такие, как BUFFPAGE) для восстановления после отказа. В типичном случае для конфигураций взаимной подмены параметры настройки отказа будут снижены до половины обычных значений или меньше. Используйте копию этих файлов, заменив ее имя (″SAMPLE″) на собственное.
3. Настройте (при необходимости) пять параметров NFS_RETRIES, START_RETRIES, MOUNT_NFS, STOP_RETRIES и FAILOVER в файле `rc.db2pe`. Параметры повтора и восстановления после отказа должны отвечать большинству реализаций. Параметр MOUNT_NFS нужно задать с учетом того, будет ли использоваться пакет для доступа к серверу NFS. Этот параметр следует задать в том случае, если вы хотите, чтобы `rc.db2pe` сама смонтировала и проверила домашний каталог NFS владельца экземпляра DB2. При задании для параметра FAILOVER значения ″YES″ будет вызвана `db2_proc_restart` и начата попытка перезапустить раздел базы данных DB2. Если операция перезапуска завершится неудачно, HACMP закроется и будет выполнено восстановление после отказа.
4. Настройте `db2_paging_action`, `db2_proc_recovery` и `nfs_auto_recovery` в файле событий. Отредактируйте `pwq`, указав в нем владельца экземпляра DB2. Настройте `db2_paging_action`, чтобы задать действие в случае заполнения пространства подкачки более чем на девяносто процентов. (Если это случится, раздел базы данных DB2 останавливается.) Измените сценарий, если при восстановлении требуются дополнительные действия.
5. При помощи `db2_inst_ha` установите сценарии и события на заданных узлах. (Это следует делать только после того, как HACMP ES будет установлена на этих узлах.) Синтаксис `db2_inst_ha`:

   ```
   db2_inst_ha $INSTNAME/sqllib/samples/hacmp/es <список_узлов> <ИМЯБАЗЫ>
   ```

   где

   ```
   $INSTNAME/sqllib/samples/hacmp/es - каталог, в котором
      находятся сценарии и событие
   <список_узлов> - узлы в стиле pcp или rexec; например,
      1-16 или 1,2,3,4
   <ИМЯБАЗЫ> - имя базы данных для обычных файлов и
      файлов параметров восстановления.
   ```

Файлы reg.parms.SAMPLE и failover.parms.SAMPLE будут скопированы на каждый узел под именем reg.parms.ИМЯБАЗЫ. db2_inst_ha копирует файлы на каждый узел в /usr/bin и вносит изменения в файлы событий HACMP:

```
/usr/sbin/cluster/events/rules.hacmprd
/usr/sbin/cluster/events/network_up_complete
/usr/sbin/cluster/events/network_down_complete
```

6. Сконфигурируйте систему и сценарии при помощи HACMP.
7. При помощи команды **create_db2_events** установите слежение за событиями для ресурса менеджера проблем (pman) и SP GUI Perspectives. Нужны дополнительные конфигурирование и настройка в Perspectives. Дополнительную информацию о Perspectives читайте в справочнике PSSP Administration.
8. При помощи команды ha_db2stop закройте разделы базы данных без восстановления после отказа HACMP ES. Для использования этой команды скопируйте файл в домашний каталог пользователя базы данных и убедитесь, что для этого пользователя заданы разрешения и право обладания. Затем, чтобы остановить базу данных без восстановления после отказа, введите от имени этого пользователя:

   ```
   ha_db2stop
   ```

   Примечание: Нужно дождаться ответа от этой команды. Выход с помощью прерывания ctrl-C или отмены процесса может преждевременно снова разрешить восстановление после отказов, и некоторые разделы базы данных могут не остановиться.

Операции сценария восстановления DB2 при помощи HACMP ES

HACMP ES вызывает сценарии восстановления DB2 так:

- node_up_local (запуск узла)

  HACMP выполняет последовательность node_up, запуская группы томов, логические тома, файловые системы и IP-адреса, заданные в группах ресурсов, принадлежащих (при каскадном методе) или назначенных (при карусельном методе) этому узлу.

  При запуске node_up_local_complete определение сервера прикладных программ, содержащее rc.db2pe, инициируется для запуска раздела базы данных, заданного в определениях сервера прикладных программ на этом физическом узле.

  Примечание: rc.db2pe, выполняемая в режиме запуска, настраивает параметры DB2, заданные в reg.parms.DATABASE для каждой базы данных DATABASE в каталоге базы данных, которая соответствует файлу параметров (parms).

Каждый узел выполняет эту последовательность при запуске. Если вы запускаете параллельно несколько кластеров HACMP, несколько узлов реабилитируются одновременно.

- `node_down_remote` (восстановление после отказа)

  HACMP вызывает группы томов, логические тома, файловые системы и IP-адреса, которые заданы в группе ресурсов на назначенном подменяющем узле.

  Когда выполняется `node_down_remote_complete`, HACMP запустит `rc.db2pe` как сценарий запуска сервера прикладных программ, заданный в группе ресурсов для этого раздела базы данных.

  Примечание: `rc.db2pe`, выполняемая в режиме взаимной подмены, останавливает запущенный на ней раздел базы данных DB2, настраивает параметры DB2, заданные в `failover.parms.DATABASE` для каждой базы данных DATABASE в каталоге базы данных, которая соответствует файлу параметров (parms), и затем запускает оба раздела базы данных на физическом подменяющем узле.

- `node_up_remote` (реинтеграция отказавшего узла - каскадная группа ресурса взаимной подмены)

  Когда `node_up_remote` выполняется на подменявшем узле, определение сервера прикладных программ запускает `rc.db2pe` в режиме остановки.

  Примечание: `rc.db2pe`, выполняемая в режиме реинтеграции (взаимная подмена), останавливает оба запущенных на ней раздела базы данных, настраивает параметры DB2, заданные в `reg.parms.DATABASE` для каждой базы данных DATABASE в каталоге базы данных, которая соответствует файлу параметров (parms), и затем запускает только тот раздел базы данных, который должен остаться на этом физическом подменявшем узле.

  Подменявший узел освобождает заданные в группах ресурсов группы томов, логические тома, файловые системы и IP-адреса, которые будут принадлежать реинтегрируемому узлу.

  HACMP вновь запускает заданные в группе ресурсов группы томов, логические тома, файловые системы и IP-адреса, которые теперь принадлежат реинтегрируемому узлу.

  При запуске `node_up_local_complete` определение сервера прикладных программ, содержащее `rc.db2pe`, инициируется для запуска раздела базы данных DB2, заданного в определении сервера прикладных программ на этом реинтегрируемом физическом узле.

Примечание: rc.db2pe, выполняемая в режиме запуска, настраивает параметры DB2, заданные в reg.parms.DATABASE для каждой базы данных DATABASE в каталоге базы данных, которая соответствует файлу параметров (parms).

- node_down_local (остановка узла или остановка с подменой)

  Когда node_down_local выполняется на останавливаемом узле, определение сервера прикладных программ запускает rc.db2pe в режиме остановки.

  Примечание: rc.db2pe, выполняемая в режиме остановки, настраивает параметры DB2, заданные в failover.parms.DATABASE для каждой базы данных DATABASE в каталоге базы данных, которая соответствует файлу параметров (parms), и затем останавливает раздел базы данных DB2 (это делается для подмены).

  HACMP освобождает заданные в группах ресурсов группы томов, логические тома, файловые системы и IP-адреса, которые теперь принадлежат узлу.

- db2_proc_recovery (отмена процесса db2)

  Все узлы выполняют сценарий db2_proc_restart. Узел, на котором произошел отказ, перезапускает нужный раздел базы данных DB2.

- db2_paging_recovery (восстановление пространства подкачки)

  Все узлы выполняют сценарий db2_paging_action. Если на узле заполнено более 70 процентов пространства подкачки, выдается команда wall. Если на узле заполнено более 90 процентов пространства подкачки, разделы базы данных DB2 на этом физическом узле останавливаются и затем перезапускаются.

- nfs_auto_recovery (сбой nfs или процесса автоматического монтирования)

  Все узлы выполняют сценарий rc.db2pe в режиме NFS. Если останавливается процесс NFS, он перезапускается. Аналогичным образом, если останавливается процесс автоматического монтирования, он перезапускается.

- network_down_complete (отказ сети - коммутатор SP)

  Вызывается сценарий net_down. Производится проверка того, что сеть - сеть коммутатора SP, и она выключена. Если это так, сценарий ждет в течение интервала, заданного пользователем. По умолчанию этот интервал составляет 100 секунд.

  Если сеть коммутатора SP включается снова, о чем свидетельствует событие network_up_complete, восстановление не выполняется.

  Если срок истекает, HACMP останавливается с восстановлением после отказа.

Примечание: За всеми событиями можно наблюдать при помощи менеджера проблем SP и SP Perspectives GUI.

Другие утилиты сценариев

Вы можете использовать и другие доступные утилиты сценариев:

- ha_cmd - команда для запуска HACMP на узлах SP с управляющей рабочей станции. Синтаксис:

  ```
  ha_cmd <диапазон_узлов> <START|STOP|TAKE|FORCE>
  ```

  где

  ```
  <диапазон_узлов> - диапазон узлов SP в стиле pcp или рехес.
  Например, "ha_cmd 3-6 START" запустит HACMP на узлах 3,4,5,6.
            "ha_cmd 5 TAKE" закроет HACMP на узле 5 для взаимной подмены.
  ```

- ha_mon - команда для слежения за файлами HACMP hacmp_out с управляющей рабочей станции SP. Синтаксис:

  ```
  ha_mon <узел>
  ```

  где

  ```
  <узел> - узел SP, за которым устанавливается наблюдение.

  ha_mon выполнит "tail -f" для файла /tmp/hacmp.out на заданном узле.
  ```

- db2_turnoff_recov - команда для временного запрета всякого восстановления HACMP (кроме восстановления после отказа), она предназначена для крайне редких случаев. Не инициируется восстановление ни процессов DB2, ни подкачки, ни NFS, ни автоматического монтирования. Эта функция удаляет раздел события для этого восстановления из файла правил HACMP. HACMP следует остановить и перезапустить. Синтаксис:

  ```
  db2_turnoff_recov <список_узлов>
  ```

- db2_turnon_recov - команда для отмены запрета на восстановление HACMP (кроме восстановления после отказа). Команда подается после db2_turnoff_recov для восстановления файлов правил HACMP, чтобы снять запрет на восстановление при событиях, определенных пользователем. HACMP следует остановить и перезапустить. Синтаксис:

  ```
  db2_turnon_recov <список_узлов>
  ```

Мониторинг кластеров HACMP

Есть сценарии для создания событий менеджера проблем SP (pman), служащие для слежения за конфигурацией DB2 HACMP ES в дополнение к утилитам слежения, уже присутствующим в HACMP ES. Чтобы отслеживать статус HACMP с управляющей рабочей станции SP:

- Установите код клиента HACMP на управляющей рабочей станции.
- Отредактируйте файл /usr/sbin/cluster/etc/clhosts, включив IP-адреса сети SP ethernet узлов, за которыми хотите следить.
- Введите команду startsrc -s clinfo, чтобы начать слежение за кластерами.

HACMP поддерживает интерфейс для слежения за кластерами (/usr/sbin/cluster/clstat).

Чтобы использовать мониторинг менеджера проблем с SP Perspectives GUI для HACMP RS и событий, определяемых пользователем:

1. Введите create_db2_events <список_узлов>, где *список_узлов* содержит узлы в стиле pcp или рехес. Этот сценарий создает пять событий pman для слежения при помощи Perspectives.

   Примечание: При создании этих событий используются переменные ресурсов PSSP.pm.User_state12-16. Если эти переменные ресурсов уже используются для другой цели, нужно внести изменения в create_db2_events и update_db2_events, чтобы использовать другие переменные ресурсов.

2. Запустите Perspectives на управляющей рабочей станции. На панели запуска выберите event perspective. Вы увидите пять событий: db2_hacmp_recovery, db2_process_recovery, db2_paging_err, db2_nfs_err и Errlog_PERM_entry.
3. Щелкните дважды по каждому событию. На появившемся экране зарегистрируйте (в таблице определений) условие для события. Нажмите кнопку рядом со стрелкой вниз под названием Name: "unnamed" и выберите то же имя, что для события, заданного вами в условии. Выберите закладку "Response Options". Нажмите кнопку в верхней части дисплея (″Send Message to Perspectives event session″). Можно задать команды, записи errlog, а также перехваты SNMP для этого события. Вывод журнала событий поддерживается только в сеансах Perspective; поэтому вы, возможно, захотите создать записи журнала ошибок AIX для каждого события. Нажмите кнопку **OK** и закройте окно.
4. В панели запуска Perspectives выберите hardware Perspective.
5. Когда появится графический интерфейс кадра устройств, выберите ″View″ и затем ″Monitor″. Появится список событий, за которыми можно следить для вашего SP. Прокрутив список до конца, вы найдете два дополнительных события: одно для восстановления HACMP DB2 (db2_ha_ind), а другое для ошибок PERM узла SP (Errlog_PERM_mon. Выберите те события, за которыми хотите следить. (При наступлении события узел выводится с красным ″X″. Если все отслеживаемые условия в нормальном состоянии, узел изображается зеленым.) В типичном случае используются события host_responds, switch_responds и node_power_LED. Можно также следить за восстановлением DB2 HACMP, а также ошибками PERM на узле.

Примечание: Переменные db2_hacmp_mon и db2_hacmp_recovery для pman и Perspectives не отражают состояние кластера HACMP. Эти переменные отражают состояние операции rc.db2pe, запускающей и останавливающей DB2. ″Настоящее″ состояние HACMP показано в мониторе HACMP clstat и отражает состояние кластера HACMP. Если вы хотите, чтобы db2_hacmp_ind отражал

мониторинг подобно состоянию HACMP, добавьте в файл /etc/inittab следующую строку:

```
haind:2:wait:/usr/bin/db2_update_events HAIND OFF 2>&1
       >/dev/null
```

Если вы планируете использовать в вашей реализации NetView, попробуйте для слежения за конфигурацией использовать HAVIEW (это часть HACMP). Информацию о конфигурировании этого продукта смотрите в документации по NetView.

Установка DB2 SP HACMP ES

Чтобы помочь вам спланировать установку HACMP ES с DB2 Universal Database, ниже приведен пошаговый обзор процессов установки и перенастройки.

Новая установка DB2 SP HACMP ES

Чтобы установить HACMP ES:

1. Установите операционную систему AIX на каждом узле SP (смотрите руководство SP Installation and Administration Guides). Убедитесь, что на управляющей рабочей станции и на каждом узле SP доступно достаточное пространство подкачки. Убедитесь, что продумана и реализована конфигурация коммутатора, а также все остальные изменяемые параметры конфигурации. Поместите на место нужный мониторинг SP (Perspectives). Убедитесь, что работают команды SP dsh, pcp и рехес.
2. Спроектируйте структуру базы данных. Как минимум, определите число используемых узлов, отображение разделов базы данных DB2 на физические узлы, требования к дискам на один узел или раздел и характеристики табличных пространств. Следует также рассмотреть, кто будет главным владельцем экземпляра DB2 и какая авторизация потребуется для этого и других пользователей.
3. Спланируйте конфигурацию внешних дисков SSA, включая избыточные адаптеры, зеркальные копии дисков и двойное подключение дисков.
4. В соответствии со схемой базы данных и конфигурацией SSA заполните рабочие листы HACMP из руководства HACMP Planning, Installation, and Administration Guides.
5. Завершите конфигурирование внешних дисков SSA. Убедитесь, что уровни микрокода одинаковы для всех дисков, и при помощи утилиты Maymap проверьте и заполните все пропуски в рабочих листах.
6. Установите DB2 UDB EEE на каждом узле SP.
7. Установите HACMP ES на каждом узле SP.
8. Установите DB2 UDB EEE HACMP ES на SP Package при помощи команды **db2_inst_ha**.

9. Создайте главного пользователя экземпляра DB2 и дайте ему доступ ко всем узлам. В этот момент он не будет пользователем высокой доступности. Временно это может быть пользователь SP на управляющей станции SP.
10. Создайте свой экземпляр DB2 и базу данных. Убедитесь, что она работает, вызвав **db2start** и затем **db2stop**, прежде чем перейти к следующему шагу.
11. Если вы хотите загрузить базу данных перед добавлением HACMP, сделайте это сейчас.
12. Сконфигурируйте топологию и группы ресурсов HACMP ES на узлах SP согласно рабочим листам HACMP и информации в этом документе.
13. Начиная с серверного узла NFS для главного пользователя DB2, измените этого пользователя (внеся изменения в `/etc/security/user` и `/etc/passwd`) на всех узлах в соответствии с указаниями в этом документе. Этот пользователь станет пользователем высокой доступности NFS; этот узел и его резервный узел внесут изменения в `/etc/exports`. Все узлы смогут монтировать этот каталог при помощи NFS (с записью в `/etc/filesystems` на каждом узле) через алиасы IP-адресов коммутатора.
14. Запакуйте (″tar″) домашний каталог главного пользователя экземпляра и распакуйте его на новом месте.
15. Создайте файловую систему NFS на каждом из узлов SP, чтобы смонтировать домашний каталог нового главного экземпляра.
16. Запустите HACMP на серверном узле NFS. Проверьте успешность этой операции, посмотрев `/tmp/hacmp.out`. При помощи команды **ha_mon** можно следить за этим файлом по мере записи.
17. Включите один за другим остальные узлы, проверяя каждый раз успешность по файлу `/tmp/hacmp.out`. При помощи команды **ha_mon** можно следить за этим файлом по мере записи.
18. Задайте, если хотите, слежение через Perspectives и менеджер проблем.
19. Проверьте работоспособность восстановления после отказа на каждом узле, имитируя ремонтные работы на всех узлах. При помощи команды **ha_cmd** (с опцией TAKE) можно корректно остановить HACMP с подменой. При помощи средств слежения и просмотра `/tmp/hacmp.out` убедитесь, что подмена и реинтеграция выполнены успешно.

Перенастройка DB2 SP HACMP ES

При перенастройке из установки без HACMP в установку с HACMP следуйте приведенным указаниям:

1. Преобразуйте существующие внешние диски в конфигурацию высокой доступности с двойным подключением и зеркальными копиями. Добавляя к конфигурации дополнительные устройства и диски, помните, что имена разных логических томов на разных узлах при двойном подключении *должны* быть уникальны. Это требование распространяется на группы томов, логические тома и файловые системы.

2. Завершите планирование HACMP и заполните соответствующие рабочие листы, включая рабочие листы в этом документе.
3. Завершите внесение изменений в конфигурацию внешних дисков SSA. Убедитесь, что уровни микрокода одинаковы для всех дисков, и при помощи утилиты Maymap проверьте и заполните все пропуски в рабочих листах.

   Примечание: В конфигурации RAID5 поддерживаются диски SSA. Допускается только конфигурация с адаптерами SSA в одном цикле RAID. Для конфигурации HACMP с дисками RAID при двойном подключении поддерживается только по одному адаптеру на узел. В этой конфигурации адаптер является уязвимой точкой для доступа к дискам, и рекомендуется расширить конфигурацию, чтобы обнаруживать выход адаптера из строя и инициировать при этом событие восстановления после отказа HACMP. Сообщение об ошибке AIX - простейший способ сконфигурировать узел для восстановления после отказа адаптера SSA. Дополнительную информацию о сообщениях об ошибках AIX смотрите в руководстве *HACMP for AIX, V4.2.2, Enhanced Scalability Installation and Administration Guide*.
4. Установите HACMP ES на каждом узле SP.
5. Установите DB2 UDB EEE HACMP ES на SP Package при помощи команды **db2_inst_ha**.
6. Сконфигурируйте топологию и группы ресурсов HACMP ES на узлах SP согласно рабочим листам HACMP и информации в этом документе.
7. Начиная с серверного узла NFS для главного пользователя DB2, измените этого пользователя (внеся изменения в `/etc/security/user` и `/etc/passwd`) на всех узлах в соответствии с указаниями в этом документе. Этот пользователь станет пользователем высокой доступности NFS; этот узел и его резервный узел внесут изменения в `/etc/exports`. Все узлы смогут монтировать этот каталог при помощи NFS (с записью в `/etc/filesystems` на каждом узле) через алиасы IP-адресов коммутатора.
8. Запакуйте (″tar″) домашний каталог главного пользователя экземпляра и распакуйте его на новом месте.
9. Создайте файловую систему NFS на каждом из узлов SP, чтобы смонтировать домашний каталог нового главного экземпляра.
10. Запустите HACMP на серверном узле NFS. Проверьте успешность этой операции, посмотрев `/tmp/hacmp.out`. При помощи команды **ha_mon** можно следить за этим файлом по мере записи.
11. Включите один за другим остальные узлы, проверяя каждый раз успешность по файлу `/tmp/hacmp.out`. При помощи команды **ha_mon** можно следить за этим файлом по мере записи.

12. Задайте, если хотите, слежение через Perspectives и менеджер проблем.
13. Проверьте работоспособность восстановления после отказа на каждом узле, имитируя ремонтные работы на всех узлах. При помощи команды **ha_cmd** (с опцией TAKE) можно корректно остановить HACMP с подменой. При помощи средств слежения и просмотра `/tmp/hacmp.out` убедитесь, что подмена и реинтеграция выполнены успешно.

Рабочие листы DB2 SP HACMP ES

Приведенные ниже рабочие листы предназначены для использования с рабочими листами HACMP, которые следует заполнить при подготовке конфигурации внешнего диска SSA (они приведены в руководстве HACMP Planning, Installation, and Administration Guides). В каждом случае приводятся и пример заполнения, и пустая таблица.

Конфигурация базы данных на внешних дисках, описанная в первом примере таблицы, показана на следующем рисунке. Оператор для создания базы данных:

```
db2 create database pwq on /newdata
```

Внешние адаптеры SSA и внешние диски SSA имеют зеркальные копии и двойное подключение, чтобы логические тома были устойчивы к отказу в любой отдельной точке. Конфигурация на схеме аналогична выводу команды **maymap**. Maymap - это утилита (доступная через AIXTOOLS), которая показывает конфигурацию внешнего диска SSA; ее следует использовать при планировании настройки.

Пример 4-узловой конфигурации базы данных DB2 с внешними дисками

- Показаны дублирующие связи для высокой доступности.

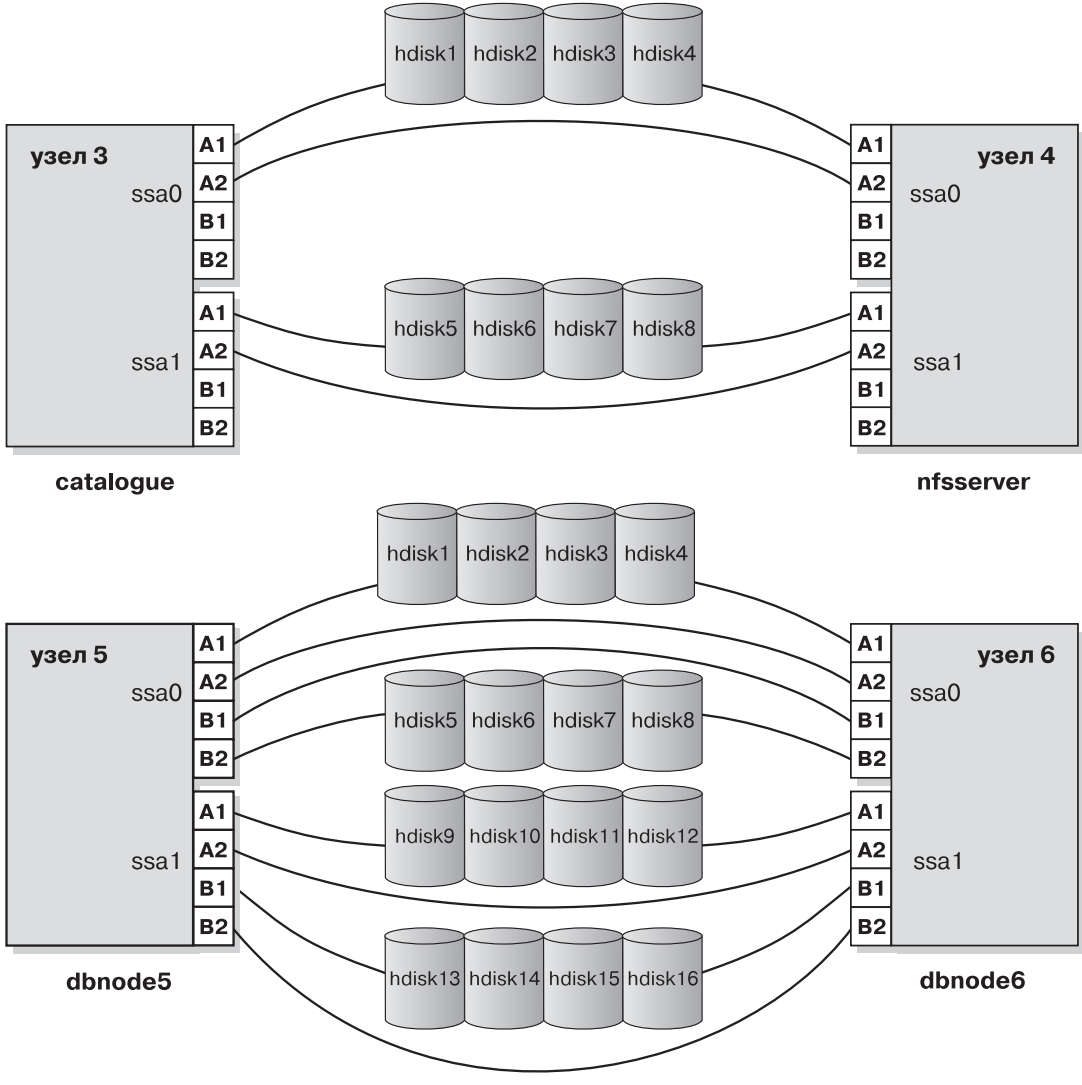


Рисунок 55. Пример настройки внешних дисков базы данных DB2 на 4 узлах

Прежде чем смотреть следующую таблицу, надо прочитать документацию HACMP, в которой рассматривается параметр "quorum" для групп томов и параметры согласования зеркальной записи на логических томах. Эти параметры прямо влияют на доступ и производительность. Обязательно

прочтите об этих параметрах и поймите их роль. В типичных случая и параметр ″quorum″, и параметр ″mirrored write consistency″ имеют значение ″off″.

Таблица 25. Группы томов HACMP, логические тома и файловые системы

Узел SP	Имя группы томов	Размер PP (Мб)	Имя логич. тома	Число PP	Число копий	Список hdisk	Точка монтиров. файловой системы	Логический том журнала файловой системы	Описание узла и резервный узел	Польз. - владелец логич. устр-ва /dev
3	havg3	8	hlv300	10	2	hdisk1 hdisk5	/newdata /pwq /NODE0003	hlog301	Точка монтиров. узла каталога; узел 4	root *
3	havg3	8	hlog301	1	2	hdisk1 hdisk5	нет	нет	Узел каталога jfslog; узел 4	root *
3	havg3	8	hlv301	10	2	hdisk2 hdisk6	нет	нет	Пр-во узла каталога rawtemp; узел 4	pwq **
4	havg4	8	hlv400	10	2	hdisk3 hdisk7	/dbmnt	hlog401	Домашний каталог nfsserver pwq; узел 3	root *
4	havg4	8	hlog401	1	2	hdisk3 hdisk7	нет	нет	jfslog сервера nfs; узел 3	root *
5	havg5	8	hlv500	10	2	hdisk1 hdisk9	/newdata/ pwq/ NODE0005	HLOG501	Точка монтиров. Dbnode5; узел 6	root *
5	havg5	8	hlog501	1	2	hdisk1 hdisk9	нет	нет	Dbnode5 jfslog; узел 6	root *
5	havg5	8	hlv501	10	2	hdisk2 hdisk10	нет	нет	Непосредст. временное простр-во Dbnode5; узел 6	pwq **

Таблица 25. Группы томов HACMP, логические тома и файловые системы (продолжение)

Узел SP	Имя группы томов	Размер PP (Мб)	Имя логич. тома	Число PP	Число копий	Список hdisk	Точка монтиров. файловой системы	Логический том журнала файловой системы	Описание узла и резервный узел	Польз. - владелец логич. устр-ва /dev
5	havg5	8	hlv502	100	2	hdisk2 hdisk10	нет	нет	Непосредст. табл. простр-во Dbnode5; узел 6	pwq **
5	havg5	8	halv503	100	2	hdisk3 hdisk11	нет	нет	Непосредст. табл. пр-во Dbnode5; узел 6	pwq **
5	havg5	8	halv504	100	2	hdisk3 hdisk11	нет	нет	Непосредст. табл. пр-во Dbnode5; узел 6	pwq **
5	havg5	8	halv505	100	2	hdisk4 hdisk12	/dbdata5	hlog501	Системное табл. пр-во Dbnode6; узел 6	root *
6	havg6	8	hlv600	10	2	hdisk5 hdisk13	/newdata/ pwq/ NODE0006	hlog601	Точка монтиров. Dbnode6; узел 5	root *
6	havg6	8	hlog601	1	2	hdisk5 hdisk13	нет	нет	Dbnode6 jfslog; узел 5	root *
6	havg6	8	hlv601	10	2	hdisk6 hdisk14	нет	нет	Непосредст. врем. пр-во Dbnode6; узел 5	pwq **
6	havg6	8	hlv602	100	2	hdisk6 hdisk14	нет	нет	Табл. пр-во Dbnode6; узел 5	pwq **
6	havg6	8	hlv603	100	2	hdisk7 hdisk15	нет	нет	Табл. пр-во Dbnode6; узел 5	pwq **
6	havg6	8	hlv604	100	2	hdisk7 hdisk15	нет	нет	Табл. пр-во Dbnode6; узел 5	pwq **

Таблица 25. Группы томов HACMP, логические тома и файловые системы (продолжение)

Узел SP	Имя группы томов	Размер PP (Мб)	Имя логич. тома	Число PP	Число копий	Список hdisk	Точка монтиров. файловой системы	Логический том журнала файловой системы	Описание узла и резервный узел	Польз. - владелец логич. устр-ва /dev
6	havg6	8	hlv605	100	2	hdisk8 hdisk16	/dbdata6	hlog601	Системн. табл. пр-во Dbnode6; узел 5	root *

Примечания:

1. Логические тома и журналы файловой системы * jfs сохраняют разрешения root.
2. Непосредственные пространства базы данных ** получают разрешения пользователя базы данных для записей непосредственного файла /dev (/dev/rxxxx).

Таблица 26. Группы томов HACMP, логические тома и файловые системы - пустая таблица

Узел SP	Имя группы томов	Размер PP (Мб)	Имя логич. тома	Число PP	Копий	Список hdisk	Точка монтир. файловой системы	Логический том журнала файловой системы	Описание узла и резервный узел	Польз. - владелец логич. устр-ва /dev

Таблица 26. Группы томов HACMP, логические тома и файловые системы - пустая таблица (продолжение)

Узел SP	Имя группы томов	Размер PP (Мб)	Имя логич. тома	Число PP	Копий	Список hdisk	Точка монтир. файловой системы	Логический том журнала файловой системы	Описание узла и резервный узел	Польз. - владелец логич. устр-ва /dev

Таблица 26. Группы томов HACMP, логические тома и файловые системы - пустая таблица (продолжение)

Узел SP	Имя группы томов	Размер PP (Мб)	Имя логич. тома	Число PP	Копий	Список hdisk	Точка монтир. файловой системы	Логический том журнала файловой системы	Описание узла и резервный узел	Польз. - владелец логич. устр-ва /dev

Таблица 27. Планирование сервера HACMP NFS

Узел SP	Внешняя файловая система	Резервный узел	Пары алиасов загрузки и IP-службы коммутатора SP	Монтируемая файловая система (/etc/filesystems)	Файловая система, задаваемая как домашний каталог базы данных	Адреса экспорта файловой системы (/etc/exports)
3	/dbmnt	4	nfs_boot_3 nfs_client_3	nfs_server:/ dbmnt as /dbi	/dbi/pwq	nfs_boot_3 nfs_client_3 nfs_server_boot nfs_server nfs_boot_5 nfs_client_5 nfs_boot_6 nfs_client_6
4	/dbmnt	3	nfs_server_boot nfs_server	nfs_server:/ dbmnt as /dbi	/dbi/pwq	nfs_boot_3 nfs_client_3 nfs_server_boot nfs_server nfs_boot_5 nfs_client_5 nfs_boot_6 nfs_client_6
5	нет	нет	nfs_boot_5 nfs_client_5	nfs_server:/ dbmnt as /dbi	/dbi/pwq	нет
6	нет	нет	nfs_boot_6 nfs_client_6	nfs_server:/ dbmnt as /dbi	/dbi/pwq	нет

Примечания:

1. `/etc/passwd` должен быть одинаков на всех узлах. Эту синхронизацию можно выполнить с управляющей рабочей станции.
2. Владелец экземпляра базы данных должен иметь разрешения на внешнюю файловую систему.
3. `/etc/filesystems` должен иметь параметры монтирования hard, bg, intr и rw.
4. `-root=ip1:ip2:ip3`

   используется для `/etc/exports` только на сервере и его резервном узле.

Таблица 28. Планирование сервера HACMP NFS - пустая таблица

Узел SP	Внешняя файловая система	Узел резерв. копир.	Пары алиасов загрузки и IP-службы коммутатора SP	Монтируемая файловая система (/etc/filesystems)	Файловая система, задаваемая как домашний каталог базы данных	Адреса экспорта файловой системы (/etc/exports)

Таблица 28. Планирование сервера HACMP NFS - пустая таблица (продолжение)

Узел SP	Внешняя файловая система	Узел резерв. копир.	Пары алиасов загрузки и IP-службы коммутатора SP	Монтируемая файловая система (/etc/filesystems)	Файловая система, задаваемая как домашний каталог базы данных	Адреса экспорта файловой системы (/etc/exports)

Глава 13. Высокая доступность в среде Windows NT

Систему баз данных можно сконфигурировать так, чтобы при отказе одного из компьютеров сервер баз данных с этого компьютера мог работать на другом компьютере. Для реализации поддержки восстановления после отказов в Windows NT можно использовать Microsoft Cluster Server (MSCS). Для использования MSCS требуется Windows NT Версии 4.0 Enterprise Edition с установленным MSCS.

MSCS может выполнять обнаружение отказов и перезапуск ресурсов в кластеризованной среде, а также обеспечивать поддержку восстановления после отказов для физических дисков и IP-адресов. (Когда компьютер, на котором ранее произошел отказ, вновь станет активным, ресурсы не будут автоматически перезапущены на нем, если такое поведение не было сконфигурировано заранее. Дополнительную информацию смотрите в разделе “Восстановление работы в исходной системе” на стр. 285.)

Прежде чем включать поддержку восстановления после отказов для экземпляров DB2, выполните следующие шаги планирования:

1. Решите, какие диски будут использоваться для хранения данных. Каждому серверу баз данных должен быть назначен по крайней мере один диск для его собственной работы. Используемый для хранения данных диск должен быть подключен к совместно используемой дисковой подсистеме и сконфигурирован как дисковый ресурс MSCS.
2. Каждый сервер баз данных, используемый для поддержки удаленных запросов, должен иметь свой IP-адрес.

При настройке поддержки восстановления после отказов можно настроить ее для существующего экземпляра или же создать новый экземпляр.

Чтобы включить поддержку восстановления после отказов, выполните следующие действия:

1. Создайте входной файл для утилиты DB2MSCS.
2. Введите команду **db2mscs**.
3. Если используется система многораздельных баз данных, зарегистрируйте отображение дисков базы данных, чтобы сделать возможной работу в режиме взаимной подмены. Смотрите раздел “Регистрация отображения дисков базы данных для конфигураций взаимной подмены в среде многораздельных баз данных” на стр. 285.

После завершения настройки экземпляра для поддержки восстановления после отказов полученная конфигурация будет подобна конфигурации на рис. 56.

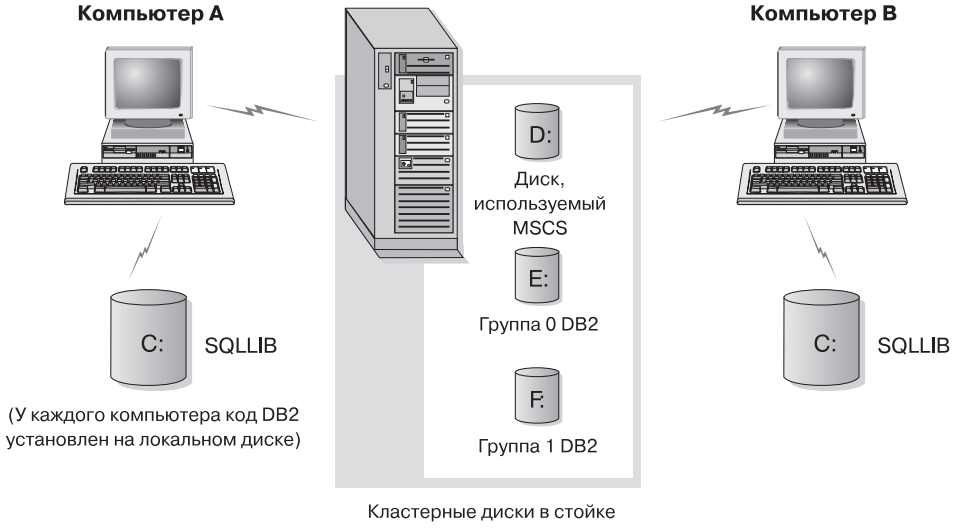


Рисунок 56. Пример конфигурации MSCS

В следующих разделах описываются разные типы поддержки восстановления после отказов и их реализации. Перед выполнением каких-либо из описанных ниже шагов на каждом компьютере, который будет использоваться в кластере MSCS, должно быть установлено программное обеспечение MSCS. Кроме этого, на каждом компьютере должна быть установлена DB2.

Конфигурации восстановления после отказов

Доступны два типа конфигурации:

- Горячее резервирование
- Взаимная подмена

В настоящее время MSCS поддерживает кластеры из двух компьютеров.

В среде многораздельных баз данных кластеры могут иметь разную конфигурацию. На одних кластерах может использоваться конфигурация горячего резервирования, на других - взаимной подмены. Например, если экземпляр DB2 включает пять рабочих станций, можно для двух компьютеров задать конфигурацию горячего резервирования, для двух других - конфигурацию взаимной подмены, а последний компьютер не конфигурировать для поддержки восстановления после отказов.

Конфигурация горячего резервирования

В конфигурации горячего резервирования один компьютер в кластере MSCS используется только для поддержки восстановления после отказов, а второй принимает участие в работе системы баз данных. Если на втором компьютере возникает отказ, его сервер баз данных будет запущен на резервном компьютере. Если в системе многораздельных баз данных на компьютере работают несколько логических узлов, при возникновении отказа на этом компьютере эти логические узлы будут запущены на резервном компьютере. Пример конфигурации горячего резервирования показан на рис. 57.

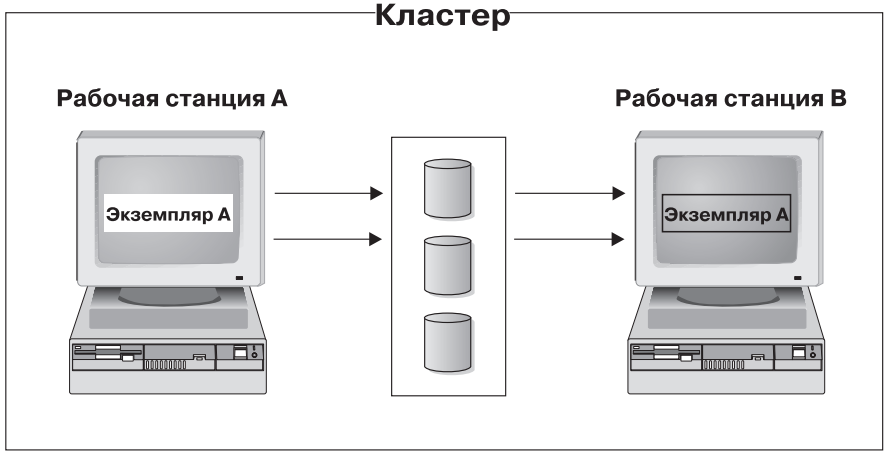


Рисунок 57. Конфигурация горячего резервирования

Конфигурация взаимной подмены

В конфигурации взаимной подмены обе рабочие станции принимают участие в работе системы баз данных (то есть на каждом компьютере работает по крайней мере один сервер баз данных). При отказе на одной из рабочих станций в кластере MSCS ее сервер баз данных будет запущен на втором компьютере. В конфигурации взаимной подмены отказ сервера баз данных на одном компьютере может произойти независимо от работы сервера баз данных на другом компьютере. В каждый момент времени любой сервер баз данных может быть активен на любом компьютере. Пример конфигурации взаимной подмены показан на рис. 58 на стр. 276.

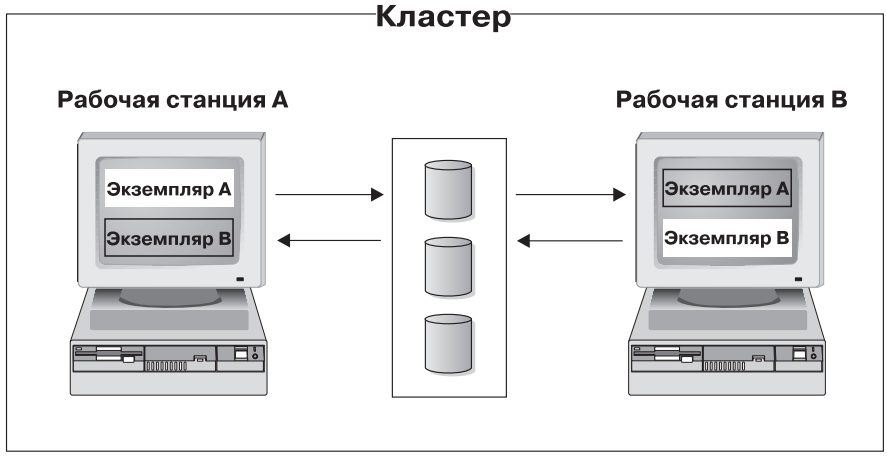


Рисунок 58. Конфигурация взаимной подмены

Использование утилиты DB2MSCS

Утилита DB2MSCS используется для создания инфраструктуры DB2 для восстановления после отказов в среде Windows NT при помощи MSCS. Эту утилиту можно использовать для включения восстановления после отказов в средах одноразделных и многораздельных баз данных.

Для каждого экземпляра один раз выполните команду **db2mscs** на компьютере, владельце экземпляра. Если только один экземпляр DB2 работает на одном компьютере в кластере MSCS, будет задана конфигурация горячего резервирования. Если на каждом компьютере в кластере MSCS работает экземпляр, нужно выполнить утилиту DB2MSCS один раз на каждом компьютере - владельце экземпляра, чтобы задать конфигурацию взаимной подмены.

Утилита DB2MSCS:

1. Считывает требуемые параметры MSCS и DB2 из входного файла с именем `DB2MSCS.CFG`. Информацию о полном наборе входных параметров смотрите в разделе “Задание содержимого файла DB2MSCS.CFG” на стр. 277.
2. Проверяет параметры, прочитанные из входного файла.
3. Регистрирует тип источника DB2.
4. Создает группу (или группы) MSCS, содержащую ресурсы MSCS и DB2.
5. Создает ресурс IP.
6. Создает ресурс сетевого имени.
7. Перемещает диски MSCS в эту группу.

8. Создает ресурс (или ресурсы) DB2.
9. Добавляет все необходимые зависимости для этого ресурса DB2.
10. Преобразует некластеризованный экземпляр DB2 в кластеризованный.
11. Активирует все ресурсы.

Синтаксис команды:

Где:

-f:*входной_файл*
Задает входной файл DB2MSCS.CFG, используемый утилитой MSCS. Если этот параметр не задан, утилита DB2MSCS будет считывать файл DB2MSCS.CFG из текущего каталога.

Задание содержимого файла DB2MSCS.CFG

Файл DB2MSCS.CFG - это текстовый файл ASCII, содержащий параметры, считываемые утилитой DB2MSCS. Каждый входной параметр задается отдельной строкой в следующем формате:
КЛЮЧЕВОЕ_СЛОВО_ПАРАМЕТРА=*значение_параметра*. Например:

```
CLUSTER_NAME=WOLFPACK
GROUP_NAME=DB2 Group
IP_ADDRESS=9.21.22.89
```

Два примера файлов конфигурации приведены в подкаталоге /CFG каталога /SQLLIB. Первый из них, DB2MSCS.EE - это пример для среды одноразделных баз данных. Второй, DB2MSCS.EEE - это пример для среды многоразделных баз данных.

Параметры для файла DB2MSCS.CFG:

DB2_INSTANCE
Имя экземпляра DB2. Если имя экземпляра *не* задано, используется экземпляр по умолчанию (задаваемый переменной среды DB2INSTANCE).

Этот параметр имеет глобальную область действия; он задается в файле DB2MSCS.CFG только один раз.

Это необязательный параметр.

Пример:

```
DB2_INSTANCE=DB2
```

Этот экземпляр должен уже существовать. Информацию о создании экземпляров смотрите в книге *DB2 Enterprise - Extended Edition for Windows Quick Beginnings*.

DB2_LOGON_USERNAME
Имя учетной записи для службы DB2.

Этот параметр имеет глобальную область действия; он задается в файле DB2MSCS.CFG только один раз.

Этот параметр требуется только для экземпляров DB2 Enterprise - Extended Edition.

Пример:

```
DB2_LOGON_USERNAME=db2user
```

DB2_LOGON_PASSWORD
Пароль учетной записи для службы DB2. Если параметр DB2_LOGON_USERNAME задан, а DB2_LOGON_PASSWORD - нет, утилита DB2MSCS просит ввести пароль. При вводе пароля в командной строке он не виден на экране.

Этот параметр имеет глобальную область действия; он задается в файле DB2MSCS.CFG только один раз.

Этот параметр требуется только для экземпляров DB2 Enterprise - Extended Edition.

Пример:

```
DB2_LOGON_PASSWORD=xxxxxx
```

CLUSTER_NAME
Имя кластера MSCS. Все ресурсы, заданные после этой строки и до следующего такого параметра, создаются в этом кластере.

Задайте этот параметр один раз для каждого кластера.

Это необязательный параметр. Если он не задан, используется имя кластера MSCS на локальном компьютере.

Пример:

```
CLUSTER_NAME=WOLFPACK
```

GROUP_NAME
Имя группы MSCS. Если задан этот параметр и эта группа MSCS не существует, создается новая группа. Если эта группа уже существует, она используется в качестве группы назначения. Все ресурсы MSCS, созданные после этой строки и до следующего заданного ключевого слова GROUP_NAME, создаются в этой группе.

Задайте этот параметр один раз для каждой группы.

Это обязательный параметр.

Пример:

```
GROUP_NAME=DB2 Group
```

DB2_NODE

Номер узла сервера раздела базы данных, который нужно включить в текущую группу MSCS. Если на одном компьютере есть несколько логических узлов, для каждого узла требуется отдельное ключевое слово DB2_NODE.

Этот параметр задается после параметра GROUP_NAME, чтобы ресурсы DB2 создавались в правильной группе MSCS.

Этот параметр требуется только для экземпляров DB2 Enterprise - Extended Edition.

Пример:

```
DB2_NODE=0
```

IP_NAME

Имя ресурса IP-адреса. Для IP_NAME можно использовать любое значение, но оно должно быть уникальным. Если задан этот параметр, создается ресурс MSCS типа IP-адрес.

Это обязательный параметр для удаленных соединений TCP/IP. Этот параметр необходимо задать для компьютера - владельца экземпляра в среде многораздельных баз данных. В средах одноразделных баз данных этот параметр необязателен.

Пример:

```
IP_NAME=IP Address for DB2
```

Примечание: Клиенты DB2 должны для внесения в каталог записи узла TCP/IP использовать TCP/IP-адрес этого ресурса IP. Если при отказе сервер баз данных запускается на другом компьютере, клиенты DB2 могут по-прежнему соединяться с этим сервером баз данных, используя IP-адрес MSCS, поскольку этот IP-адрес доступен на резервном компьютере.

Атрибуты ресурса IP:

IP_ADDRESS

TCP/IP-адрес этого ресурса IP. Используйте это ключевое слово, чтобы задать TCP/IP-адрес для указанного ранее ресурса IP.

Это обязательный параметр, если задан параметр IP_NAME.

Пример:

```
IP_ADDRESS=9.21.22.34
```

IP_SUBNET
Имя подсети для указанного ранее ресурса IP.

Это обязательный параметр, если задан параметр IP_NAME.

Пример:

```
IP_SUBNET=255.255.255.0
```

IP_NETWORK
Имя сети MSCS, в которую входит указанный ранее ресурс IP. Если этот параметр не задан, используется первая обнаруженная системой сеть MSCS.

Это необязательный параметр.

Пример:

```
IP_NETWORK=Token Ring
```

NETNAME_NAME
Имя ресурса сетевого имени. Этот параметр используется для создания ресурса сетевого имени.

Это необязательный параметр для сред одноразделных баз данных. Для сред многораздельных баз данных этот параметр обязателен.

Пример:

```
NETNAME_NAME=Network name for DB2
```

Атрибуты ресурса сетевого имени:

NETNAME_VALUE
Значение сетевого имени.

Это обязательный параметр, если задан параметр NETNAME_NAME.

Пример:

```
NETNAME_VALUE=DB2SRV
```

NETNAME_DEPENDENCY
Список зависимостей для ресурса сетевого имени. Каждый ресурс сетевого имени должен иметь зависимость от ресурса IP-адреса. Если этот параметр не задан, ресурс сетевого имени будет иметь зависимость от первого ресурса IP в этой группе.

Это необязательный параметр.

Пример:

```
NETNAME_DEPENDENCY=IP Address for DB2
```

DISK_NAME
Имя ресурсов физических дисков, которые нужно переместить в текущую группу. Задайте столько дисковых ресурсов, сколько нужно.

Примечания:

1. Эти дисковые ресурсы уже должны существовать.
2. Когда утилита DB2MSCS конфигурирует экземпляр DB2 для поддержки MSCS, каталог экземпляра копируется на *первый* диск MSCS в этой группе. Чтобы задать другой диск MSCS для каталога экземпляра, используйте параметр INSTPROF_DISK.

Пример:

```
DISK_NAME=Disk E:
DISK_NAME=Disk F:
```

INSTPROF_DISK

Необязательный параметр, который задает диск MSCS, содержащий каталог экземпляра DB2. Если этот параметр *не* задан, утилита DB2MSCS использует *первый* диск MSCS, входящий в ту же группу, что и каталог экземпляра.

Каталог экземпляра DB2 создается на диске MSCS в каталоге `X:\DB2PROFS` (где X - буква диска MSCS).

Пример:

```
INSTPROF_DISK=Disk E:
```

TARGET_DRVMAP_DISK

Необязательный параметр, задающий диск назначения MSCS для отображения дисков базы данных. Если база данных создана на диске MSCS, не входящем в ту же группу, что и этот узел, раздел базы данных будет находиться на диске назначения отображения дисков. Если этот параметр не задан, отображение дисков базы данных надо зарегистрировать вручную с помощью утилиты DB2DRVMP.

Пример:

```
TARGET_DRVMAP_DISK = Disk E:
```

Настройка восстановления после отказов для системы одноразделной базы данных

При выполнении утилиты DB2MSCS для системы одноразделной базы данных одна группа MSCS содержит ресурсы DB2 и все зависимые ресурсы MSCS (IP-адрес, имя сети и диски). Например, для системы одноразделной базы данных содержимое файла `DB2MSCS.CFG` будет выглядеть так:

```
#
#  DB2MSCS.CFG для системы одноразделной базы данных
#
DB2_INSTANCE=DB2
CLUSTER_NAME=MSCS
GROUP_NAME=DB2 Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
```

```
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk E:
```

Настройка конфигурации взаимной подмены для двух систем одноразделных баз данных

Можно настроить две системы одноразделных баз данных, каждая из которых находится на отдельном компьютере, чтобы при отказе системы баз данных на одном из компьютеров она перезапускалась на другом узле MSCS.

Чтобы настроить поддержку восстановления после отказов для такой конфигурации, нужно выполнить утилиту DB2MSCS один раз для каждого компьютера - владельца экземпляра. Нужно создать файл конфигурации для каждой системы баз данных.

Предположим, что экземпляры DB2 называются DB2A и DB2B. Файл DB2MSCS.CFG для экземпляра DB2A будет выглядеть так:

```
#
#  DB2MSCS.CFG для первой системы одноразделной базы данных
#
DB2_INSTANCE=DB2A
CLUSTER_NAME=MSCS
GROUP_NAME=DB2A Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk E:
```

Файл DB2MSCS.CFG для экземпляра DB2B будет выглядеть так:

```
#
#  DB2MSCS.CFG для второй системы одноразделной базы данных
#
DB2_INSTANCE=DB2B
CLUSTER_NAME=MSCS
GROUP_NAME=DB2B Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk F:
```

Полный пример смотрите в разделе “Пример - Настройка двух одноразделных экземпляров для конфигурации взаимной подмены” на стр. 288.

Настройка нескольких кластеров MSCS для системы многораздельных баз данных

При выполнении утилиты DB2MSCS для системы многораздельных баз данных для каждого физического компьютера, участвующего в этой системе, создается одна группа MSCS. Файл DB2MSCS.CFG должен содержать несколько разделов, каждый из которых должен иметь свои значения для параметра GROUP_NAME и для всех требуемых зависимых ресурсов этой группы.

Кроме этого, необходимо задать параметр DB2_NODE для каждого сервера раздела базы данных в каждой группе MSCS. Если есть несколько логических узлов, для каждого логического узла требуется отдельное ключевое слово DB2_NODE.

Например, предположим, что есть система многораздельных баз данных, состоящая из четырех серверов разделов базы данных на четырех компьютерах, и нужно сконфигурировать два кластера MSCS с использованием конфигурации взаимной подмены. Файл конфигурации DB2MSCS.CFG будет выглядеть так:

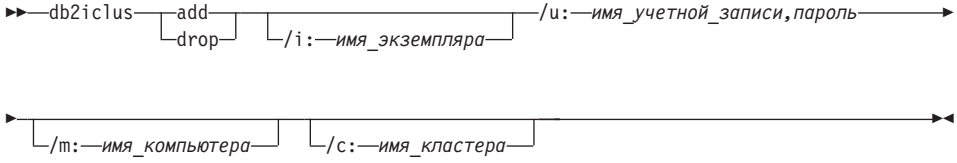
```
#
#  DB2MSCS.CFG для одной системы многораздельных баз данных
#  с несколькими кластерами
DB2_INSTANCE=DB2MPP
DB2_LOGON_USERNAME=db2user
DB2_LOGON_PASSWORD=xxxxxx
CLUSTER_NAME=MSCS1
# Группа 1
GROUP_NAME=DB2 Group 1
DB2_NODE=0
IP_NAME=...

...
# Группа 2
GROUP_NAME=DB2 Group 2
DB2_NODE=1
IP_NAME=...

...

CLUSTER_NAME=MSCS2
# Группа 3
GROUP_NAME=DB2 Group 3
DB2_NODE=2
IP_NAME=...

...
# Группа 4
GROUP_NAME=DB2 Group 4
DB2_NODE=3
IP_NAME=...

...
```

Полный пример смотрите в разделе “Пример - настройка системы многораздельной базы данных с четырьмя узлами для конфигурации взаимной подмены” на стр. 291.

Поддержание системы MSCS

При выполнении утилиты DB2MSCS она создает инфраструктуру для поддержки восстановления после отказов для всех компьютеров в кластере MSCS. Чтобы удалить с компьютера эту поддержку, введите команду **db2iclus** с опцией ″drop″. Чтобы снова включить эту поддержку для компьютера, используйте опцию ″add″.

Синтаксис команды:



Где:

add — Включает поддержку восстановления после отказов на компьютере, добавляя его в кластер MSCS. После этого на этом компьютере может перезапускаться сбойный ресурс DB2 (сервер баз данных).

drop — Удаляет поддержку восстановления после отказов с компьютера, удаляя его из кластера MSCS.

/i: *имя_экземпляра* — Имя экземпляра. (Этот параметр переопределяет значение переменной среды DB2INSTANCE.)

/u: *имя_учетной_записи*, *пароль* — Учетная запись домена, используемая в качестве имени учетной записи регистрации для службы DB2. Например:

```
/u:domainA\db2nt,password
```

Этот параметр требуется только с опцией ″add″.

/m:*имя_компьютера* — Имя компьютера, который нужно добавить в кластер MSCS или удалить из кластера MSCS. Эту опцию нужно задать, если эта команда запущена на компьютере, отличном от

компьютера, для которого нужно изменить поддержку восстановления после отказов.

/c: *имя_кластера* — Имя кластера MSCS, под которым он известен в локальной сети. Это имя задается при создании кластера MSCS.

Восстановление работы в исходной системе

По умолчанию работа групп не будет восстанавливаться на исходном компьютере (на котором был сбой). Если вручную не сконфигурировать группу DB2, чтобы ее работа восстанавливалась на исходном компьютере, после исправления ошибки, вызвавшей отказ, эта группа будет продолжать работать на альтернативном узле MSCS.

Если группа DB2 сконфигурирована для автоматического восстановления работы на исходном компьютере, все ресурсы в этой группе DB2, включая ресурс DB2, будут восстанавливаться на исходном компьютере, как только он станет доступным. Если во время восстановления на исходном компьютере существует соединение с базой данных, ресурс DB2 не может быть переведен в пассивное состояние и операция восстановления на исходном компьютере будет неудачной.

Если нужно, чтобы в процессе восстановления на исходном компьютере принудительно завершались все соединения с базой данных, задайте для переменной реестра DB2_FALLBACK значение ON. Эта переменная задается так:

```
db2set DB2_FALLBACK=ON
```

После задания этой переменной реестра не нужно перезагружать или перезапускать службу кластеров.

Регистрация отображения дисков базы данных для конфигураций взаимной подмены в среде многораздельных баз данных

При создании базы данных в среде многораздельных баз данных можно задать букву диска, на котором должна быть создана эта база данных.

Примечание: Отображение дисков базы данных не задается для сред одноразделных баз данных.

При выполнении команды CREATE DATABASE она ожидает, что заданный диск будет одновременно доступен для всех компьютеров, входящих в экземпляр. Так как это невозможно, DB2 использует отображение дисков базы данных, чтобы назначить для одного диска разные имена для разных компьютеров.

Например, предположим, что экземпляр DB2 с именем DB2 содержит два сервера разделов базы данных:

```
NODE0 активен на компьютере WOLF_NODE_0
NODE1 активен на компьютере WOLF_NODE_1
```

Предположим также, что совместно используемый диск E: входит в ту же группу, что и NODE0, а совместно используемый диск F: входит в ту же группу, что и NODE1.

Чтобы создать базу данных на совместно используемом диске E:

```
db2 create database mppdb on e:
```

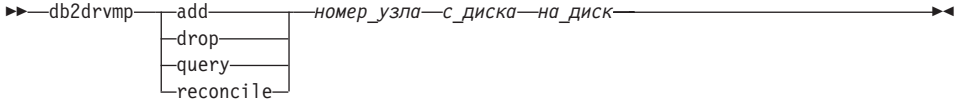
Чтобы эта команда была выполнена успешно, диск E: должен быть доступен на обоих компьютерах. В конфигурации взаимной подмены каждый сервер раздела базы данных может быть активен на разных компьютерах, а диск кластера E: доступен только на одном компьютере. В этой ситуации команда CREATE DATABASE всегда будет неудачной.

Чтобы решить эту проблему, для этого диска базы данных должно быть задано такое отображение:

```
Для NODE0 - отображение диска F: на диск E:
Для NODE1 - отображение диска E: на диск F:
```

Теперь всякое обращение к базе данных для NODE0 на диск F: будет отображаться на диск E:, а всякое обращение к базе данных для NODE1 на диск E: будет отображаться на диск F:. Используя отображение дисков, команда CREATE DATABASE создаст файлы базы данных на диске E: для NODE0 и на диске F: для NODE1.

Для задания отображения дисков используется команда **db2drvmp**. Синтаксис этой команды:



Используются следующие параметры:

add Задать новое отображение дисков базы данных.

drop Удалить существующее отображение дисков базы данных.

query Запросить отображение дисков базы данных.

reconcile Исправить диск отображения дисков базы данных при повреждении содержимого реестра. Дополнительную

информацию смотрите в разделе “Восстановление отображения дисков базы данных”.

номер_узла Этот параметр обязателен для операций добавления и отбрасывания.

с_диска Буква диска, с которого задается отображение. Этот параметр обязателен для операций добавления и отбрасывания.

на_диск Буква диска, на который задается отображение. Этот параметр обязателен для операций добавления. Он не используется для других операций.

Чтобы задать отображение дисков базы данных с F: на E: для узла NODE0, используйте команду:

```
db2drvmp add 0 F E
```

Примечание: Отображение дисков базы данных не применяется для табличных пространств, контейнеров или каких-либо других объектов хранения базы данных.

Подобным образом, чтобы задать отображение дисков базы данных с E: на F: для узла NODE1, используйте команду:

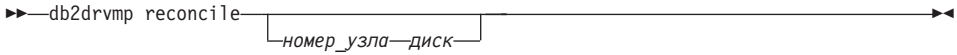
```
db2drvmp add 1 E F
```

Примечание: Задание или изменение отображения дисков базы данных не вступает в силу немедленно. Чтобы активировать отображение дисков базы данных, надо при помощи инструмента администратора кластеров сделать этот ресурс DB2 пассивным и затем вновь активировать его.

Задав ключевое слово TARGET_DRVMAP_DISK в файле DB2MSCS.CFG, можно разрешить автоматическое задание отображения дисков.

Восстановление отображения дисков базы данных

Когда база данных создается на компьютере, на котором действует отображение дисков, информация об отображении сохраняется в скрытом файле. Это делается для предотвращения удаления информации о диске после создания базы данных. Если отображение дисков базы данных было случайно удалено, можно *восстановить* это отображение дисков базы данных. Чтобы восстановить отображение, выполните команду **db2drvmp reconcile** для каждого сервера раздела базы данных, содержащего эту базу данных. Синтаксис команды:



Используются следующие параметры:

номер_узла Номер узла, для которого выполняется исправление. Если *номер_узла* не задан, эта команда исправляет отображения для всех узлов.

диск Диск, для которого выполняется исправление. Если диск не задан, эта команда исправляет отображения для всех дисков.

Команда **db2drvmp** проверяет все диски на компьютере для разделов базы данных, управляемых этим сервером разделов баз данных, и вновь задает в реестре требуемое отображение дисков базы данных.

Пример - Настройка двух одноразделных экземпляров для конфигурации взаимной подмены

Цель - настроить два экземпляра одноразделных баз данных для поддержки восстановления после отказов в конфигурации взаимной подмены. В этом примере четыре сервера конфигурируются в два кластера MSCS. Когда используется конфигурация взаимной подмены, при возникновении отказа на любом из компьютеров сконфигурированный для этого компьютера сервер баз данных будет запущен на другом компьютере (который сконфигурирован для этого с помощью программ MSCS).

Итоговая конфигурация содержит два кластера MSCS. Каждый кластер содержит:

- Два сервера, каждый с 64 Мбайтами памяти и одним локальным диском SCSI объемом 2 Гбайта
- Одну стойку дисков SCSI, содержащую три совместно используемых диска SCSI объемом 2 Гбайта каждый.

Кроме этого, на каждом компьютере установлен один адаптер Ethernet 100X.

На каждом компьютере установлено следующее программное обеспечение:

- Windows NT Версии 4.0 Enterprise Edition с установленным MSCS
- DB2 Universal Database Enterprise Edition Версия 7.

Итоговая конфигурация сети:

Сервер 1:	Сервер 2:
• Имя компьютера: db2test1 • Имя хоста TCP/IP: db2test1 • IP-адрес: 9.9.9.1 (маска подсети: 255.255.255.0) • Имя кластера MSCS: ClusterA	• Имя компьютера: db2test2 • Имя хоста TCP/IP: db2test2 • IP-адрес: 9.9.9.2 (маска подсети: 255.255.255.0) • Имя кластера MSCS: ClusterA

Оба компьютера в сети сконфигурированы для работы с TCP/IP и соединены с собственной локальной сетью с помощью концентратора Ethernet 100 T-base. В отсутствие сервера имен доменов (DNS) все компьютеры имеют локальные файлы хостов TCP/IP `hosts`, содержащие следующие записи:

```
9.9.9.1  db2test1  # для сервера 1
9.9.9.2  db2test2  # для сервера 2
9.9.9.3  ClusterA  # для кластера MSCS ClusterA
9.9.9.4  db2tcp1   # для соединения удаленного клиента DB2 с сервером 1
9.9.9.5  db2tcp2   # для соединения удаленного клиента DB2 с сервером 2
```

Предварительные задачи

Перед выполнением следующих задач подразумевается, что оба компьютера входят в один домен с именем `DB2NTD`:

1. Создайте учетную запись домена для DB2, входящую в группу локальных администраторов на компьютерах, на которых будет работать DB2. Используйте эту учетную запись для выполнения всех задач:
   - Задайте имя пользователя `db2nt`.
   - Задайте пароль `db2nt`.
2. Установите MSCS на компьютерах `db2test1` и `db2test2`:
   - Задайте имя кластера MSCS `ClusterA`.
   - IP-адрес кластера - `9.9.9.3`.
   - Совместно используемый диск D: будет использоваться программным обеспечением MSCS.
   - Совместно используемые диски E: и F: будут использоваться DB2.
3. Установите DB2 Universal Database Enterprise Edition версии 7 на компьютере `db2test1`. Установите это программное обеспечение в каталог `C:\SQLLIB` на локальном диске.
4. Установите DB2 Universal Database Enterprise Edition версии 7 на компьютере `db2test2`. Установите это программное обеспечение в каталог `C:\SQLLIB` на локальном диске.

Следующий шаг - задать содержимое файла `DB2MSCS.CFG` для каждого экземпляра и выполнить утилиту DB2MSCS для каждого экземпляра.

Выполнение утилиты DB2MSCS

Чтобы настроить компьютер db2test1, выполните следующие задачи:

1. На компьютере db2test1 зарегистрируйтесь как пользователь db2nt. Пароль - db2nt.
2. Создайте экземпляр DB2 с именем DB2A, если он еще не существует. Команда создания этого экземпляра:

   ```
   db2icrt DB2A
   ```

3. Задайте содержимое файла DB2MSCS.CFG для экземпляра DB2 на компьютере db2test1:

   ```
   #
   #  DB2MSCS.CFG для системы базы данных
   #  на компьютере db2test1
   DB2_INSTANCE=DB2A
   CLUSTER_NAME=ClusterA
   #
   # Группа 1
   GROUP_NAME=DB2A Group
   IP_NAME=IP Address for DB2A
   IP_ADDRESS=9.9.9.4
   IP_SUBNET=255.255.255.0
   IP_NETWORK=ClusterA
   NETNAME_NAME=Network name for DB2A
   NETNAME_VALUE=DB2SRV1
   NETNAME_DEPENDENCY=IP Address for DB2A
   DISK_NAME=Disk E:
   INSTPROF_DISK=Disk E:
   ```

4. Выполните утилиту DB2MSCS:

   ```
   db2mscs -f:DB2MSCS.CFG
   ```

5. Завершите работу под именем пользователя db2nt.
6. На компьютере db2test2 зарегистрируйтесь как пользователь db2nt, входящий в группу локальных администраторов. Пароль - db2nt.
7. Создайте экземпляр DB2 с именем DB2B, если он еще не существует. Команда создания этого экземпляра:

   ```
   db2icrt DB2B
   ```

8. Задайте содержимое файла DB2MSCS.CFG для экземпляра DB2 на компьютере db2test2:

   ```
   #
   #  DB2MSCS.CFG для системы базы данных
   #  на компьютере db2test2
   DB2_INSTANCE=DB2B
   CLUSTER_NAME=ClusterA
   #
   # Группа 1
   GROUP_NAME=DB2B Group
   IP_NAME=IP Address for DB2B
   IP_ADDRESS=9.9.9.5
   IP_SUBNET=255.255.255.0
   ```

```
IP_NETWORK=ClusterA
NETNAME_NAME=Network name for DB2B
NETNAME_VALUE=DB2SRV2
NETNAME_DEPENDENCY=IP Address for DB2B
DISK_NAME=Disk F:
INSTPROF_DISK=Disk F:
```

9. Выполните утилиту DB2MSCS:

```
db2mscs -f:DB2MSCS.CFG
```

10. Завершите работу под именем пользователя db2nt.

Пример - настройка системы многораздельной базы данных с четырьмя узлами для конфигурации взаимной подмены

Цель - настроить систему многораздельной базы данных с четырьмя узлами для поддержки восстановления после отказов в конфигурации взаимной подмены. В этом примере четыре сервера конфигурируются в два кластера MSCS. Когда используется конфигурация взаимной подмены, при возникновении сбоя на любом из компьютеров сконфигурированные для этого компьютера серверы баз данных будут запущены как логические узлы на другом компьютере (который сконфигурирован для этого с помощью программ MSCS).

Итоговая конфигурация содержит два кластера MSCS. Каждый кластер содержит:

- Два сервера, каждый с 64 Мбайтами памяти и одним локальным диском SCSI объемом 2 Гбайта
- Одну стойку дисков SCSI, содержащую три совместно используемых диска SCSI объемом 2 Гбайта каждый.

Кроме этого, на каждом компьютере установлен один адаптер Ethernet 100X.

На каждом компьютере установлено следующее программное обеспечение:

- Windows NT Версии 4.0 Enterprise Edition с установленным MSCS
- DB2 Universal Database Extended Enterprise Edition версии 7.

Итоговая конфигурация сети:

Сервер 1:	Сервер 2:
• Имя компьютера: db2test1	• Имя компьютера: db2test2
• Имя хоста TCP/IP: db2test1	• Имя хоста TCP/IP: db2test2
• IP-адрес: 9.9.9.1 (маска подсети: 255.255.255.0)	• IP-адрес: 9.9.9.2 (маска подсети: 255.255.255.0)
• Имя кластера MSCS: ClusterA	• Имя кластера MSCS: ClusterA

Сервер 3:	Сервер 4:
• Имя компьютера: db2test3 • Имя хоста TCP/IP: db2test3 • IP-адрес: 9.9.9.3 (маска подсети: 255.255.255.0) • Имя кластера MSCS: ClusterB	• Имя компьютера: db2test4 • Имя хоста TCP/IP: db2test4 • IP-адрес: 9.9.9.4 (маска подсети: 255.255.255.0) • Имя кластера MSCS: ClusterB

Все компьютеры в сети сконфигурированы для работы с TCP/IP и соединены с собственной локальной сетью с помощью концентратора Ethernet 100 T-base. В отсутствие сервера имен доменов (DNS) все компьютеры имеют локальные файлы хостов TCP/IP `hosts`, содержащие следующие записи:

```
9.9.9.1       db2test1       # для сервера 1
9.9.9.2       db2test2       # для сервера 2
9.9.9.3       db2test3       # для сервера 3
9.9.9.4       db2test4       # для сервера 4
9.9.9.5       ClusterA       # для кластера MSCS 1
9.9.9.6       ClusterB       # для кластера MSCS 2
9.9.9.7       db2tcp         # для соединения удаленного клиента DB2
```

Предварительные задачи

Перед выполнением следующих задач подразумевается, что все четыре компьютера входят в один домен с именем `DB2NTD`:

1. Создайте учетную запись домена для DB2, входящую в группу локальных администраторов на компьютерах, на которых будет работать DB2. Используйте эту учетную запись для выполнения всех задач:
   - Задайте имя пользователя `db2nt`.
   - Задайте пароль `db2nt`.
2. Создайте вторую учетную запись домена с опцией *"пароль с неограниченным сроком действия"*. Эта учетная запись будет связана со службами DB2:
   - Задайте имя пользователя `db2mpp`.
   - Задайте пароль `db2mpp`.
3. Установите MSCS на компьютерах `db2test1` и `db2test2`:
   - Задайте имя кластера MSCS `ClusterA`.
   - IP-адрес кластера - `9.9.9.5`.
   - Совместно используемый диск D: будет использоваться программным обеспечением MSCS.
   - Совместно используемые диски E: и F: будут использоваться DB2.
4. Установите MSCS на компьютерах `db2test3` и `db2test4`:
   - Задайте имя кластера MSCS `ClusterB`.
   - IP-адрес кластера - `9.9.9.6`.

- Совместно используемый диск D: будет использоваться программным обеспечением MSCS.
- Совместно используемые диски E: и F: будут использоваться DB2.

5. Установите DB2 Enterprise - Extended Edition на компьютере db2test1:
   - Выберите опцию *"Этот компьютер будет сервером раздела баз данных - владельцем экземпляра"*.
   - Учетная запись для службы DB2 - db2mpp. Пароль - db2mpp.
   - Установите это программное обеспечение в каталог C:\SQLLIB на локальном диске.
6. Установите DB2 Enterprise - Extended Edition на компьютерах db2test2, db2test3 и db2test4:
   - Выберите опцию *"Этот компьютер будет новым узлом существующей системы многораздельных баз данных"*.
   - Задайте db2test1 в качестве компьютера, владельца экземпляра.
   - Учетная запись для службы DB2 - db2mpp. Пароль - db2mpp.
   - Установите это программное обеспечение в каталог C:\SQLLIB на локальном диске.

Следующий шаг - задать содержимое файла DB2MSCS.CFG и выполнить утилиту DB2MSCS.

Выполнение утилиты DB2MSCS

Чтобы настроить компьютер db2test1, выполните следующие задачи:

1. Зарегистрируйтесь как пользователь db2nt, входящий в группу локальных администраторов. Пароль - db2nt.
2. Задайте содержимое файла DB2MSCS.CFG:

```
#
#  DB2MSCS.CFG для одной системы многораздельной базы данных
#  с несколькими кластерами MSCS
DB2_INSTANCE=DB2MPP
CLUSTER_NAME=ClusterA
DB2_LOGON_USERNAME=db2mpp
DB2_LOGON_PASSWORD=db2mpp
# Группа 1
# для узла DB2 0
GROUP_NAME=DB2NODE0
DB2_NODE=0
IP_NAME=IP Address for DB2
IP_ADDRESS=9.9.9.7
IP_SUBNET=255.255.255.0
IP_NETWORK=Ethernet
NETNAME_NAME=Network name for DB2
NETNAME_VALUE=DB2WOLF
```

```
NETNAME_DEPENDENCY=IP Address for DB2
DISK_NAME=Disk E:
INSTPROF_DISK=Disk E:
#
# Группа 2
# для узла DB2 1
GROUP_NAME=DB2NODE1
DB2_NODE=1
DISK_NAME=Disk F:
#
CLUSTER_NAME=ClusterB
# Группа 3
# для узла DB2 2
GROUP_NAME=DB2NODE2
DB2_NODE=2
DISK_NAME=Disk E:
#
# Группа 4
# для узла DB2 3
GROUP_NAME=DB2NODE3
DB2_NODE=3
DISK_NAME=Disk F:
```

3. Выполните утилиту DB2MSCS:

   ```
   db2mscs -f:DB2MSCS.CFG
   ```

4. Завершите работу под именем пользователя db2nt.

Заключительные шаги - зарегистрировать отображение дисков базы данных для этих двух кластеров MSCS.

Регистрация отображения дисков базы данных для кластера ClusterA

Чтобы зарегистрировать отображение дисков базы данных для кластера MSCS ClusterA, выполните следующие задачи:

1. На компьютере db2test1 зарегистрируйтесь как пользователь db2mpp (это имя учетной записи связано со службами DB2). Пароль - db2mpp.
2. Чтобы зарегистрировать отображение дисков базы данных, введите следующие команды:

   ```
   db2drvmp add 0 F E

   db2drvmp add 1 E F
   ```

3. Сделайте пассивными все ресурсы DB2, затем вновь активируйте их.

Регистрация отображения дисков базы данных для кластера ClusterB

Чтобы зарегистрировать отображение дисков базы данных для кластера MSCS ClusterB, выполните следующие задачи:

1. На компьютере db2test3 зарегистрируйтесь как пользователь db2mpp (это имя учетной записи связано со службами DB2). Пароль - db2mpp.

2. Чтобы зарегистрировать отображение дисков базы данных, введите следующие команды:

```
db2drvmp add 2 F E
db2drvmp add 3 E F
```

3. Сделайте пассивными все ресурсы DB2, затем вновь активируйте их.

Управление DB2 в среде MSCS

При использовании кластеров MSCS для экземпляра DB2 требуется дополнительное планирование ежедневных операций, размещения базы данных и конфигурации базы данных. Для обеспечения прозрачной работы DB2 на узле MSCS нужно выполнить дополнительные задачи администратора. Все связанные с DB2 ресурсы операционной системы должны быть доступны на всех узлах MSCS. Некоторые из этих ресурсов операционной системы находятся вне области видимости MSCS. Это значит, что их нельзя определить как ресурсы MSCS. Каждая система должна быть сконфигурирована так, чтобы на всех узлах MSCS были доступны одни и те же ресурсы операционной системы. В следующих разделах описываются дополнительные действия, которые нужно выполнить.

Запуск и остановка ресурсов DB2

Для запуска и остановки ресурсов DB2 нужно использовать инструмент администратора кластеров. Есть разные способы запуска экземпляра DB2, например, команда **db2start** и опция **Службы** на Панели управления. Однако, если DB2 запущена не из администратора кластеров, программное обеспечение MSCS не будет знать состояние экземпляра DB2. Если экземпляр DB2 запущен с помощью администратора кластеров, а остановлен с помощью команды **db2stop**, программное обеспечение MSCS будет считать команду **db2stop** программным сбоем и попытается перезапустить DB2. (Существующие в настоящее время интерфейсы MSCS не поддерживают уведомления о *состоянии ресурсов*.)

Аналогично, если для запуска экземпляра DB2 используется команда **db2start**, MSCS не может определить, что этот ресурс запущен; при возникновении сбоя сервера баз данных MSCS не перезапустит этот ресурс DB2 на резервном компьютере кластера.

Для экземпляра DB2 могут применяться три операции:

Online (подключение)
Эта операция эквивалентна использованию команды **db2start**. Если DB2 уже активна, эту операцию можно использовать просто для того, чтобы сообщить MSCS, что DB2 активна. Сообщения об ошибках, возникших во время этой операции, будут записаны в журнал событий Windows NT.

Offline (отключение)
Эта операция эквивалентна использованию команды **db2stop**. Если есть

активные подключения к экземпляру, эта операция завершится неудачно. Это соответствует поведению команды **db2stop**.

Fail resource (Объявить неработающим)
Эта операция эквивалентна использованию команды **db2stop** с заданной опцией **force**. DB2 отсоединит от этой системы DB2 все прикладные программы и остановит все серверы базы данных.

Выполнение сценариев

Сценарии можно выполнить как до, так и после активации ресурса DB2. Эти сценарии *должны* находиться в каталоге профиля экземпляра, заданном в переменной среды DB2INSTPROF. Этот каталог задается параметром ″-p″ команды **db2icrt**. Узнать это значение можно с помощью команды:

```
db2set -i:имя_экземпляра DB2INSTPROF
```

Этот каталог экземпляра должен находиться на кластеризованном диске, чтобы он был доступен для всех узлов кластера.

Эти файлы сценариев не являются обязательными и они выполняются, только если они найдены в каталоге экземпляра. Служба кластера MSCS запускает их в фоновом режиме. В файлах сценариев должна перехватываться вся выходная информация, направляемая командами в этих файлах на стандартное устройство вывода. Эти выходные данные не выводятся на экран.

По умолчанию в среде многораздельной базы данных для каждого сервера раздела базы данных в экземпляре будет использован один и тот же сценарий. Если нужно различать разные серверы разделов баз данных экземпляра, используйте значение переменной среды DB2NODE, которой на разных узлах присваиваются различные номера узлов (например, используйте оператор IF в файлах `db2cpre.bat` и `db2cpost.bat`).

Выполнение сценариев перед активацией ресурсов DB2

Если нужно выполнять сценарий *перед* активацией ресурса DB2, этот сценарий *должен* быть назван `db2cpre.bat`. DB2 вызывает функции, которые запустят этот пакетный файл из процессора командной строки (CLP) Windows NT, и прежде, чем активировать ресурс DB2, ожидает, пока процессор командной строки завершит выполнение этих функций. Этот пакетный файл можно использовать для таких задач, как изменение конфигурации менеджера баз данных DB2. Можно изменить значения некоторых параметров менеджера баз данных, если на резервной системе не хватает ресурсов и нужно уменьшить объем ресурсов, используемых DB2.

Команды, помещенные в сценарий `db2cpre.bat`, должны выполняться синхронно. В противном случае ресурс DB2 может стать активным до того, как будут выполнены все команды сценария, что может вызвать непредвиденное поведение. В частности, в сценарии `db2cpre.bat` нельзя использовать команду

db2cmd, поскольку она, в свою очередь, запустит другой процессор командной строки, который будет выполнять команды DB2 асинхронно с программой **db2cmd**.

Если в файле db2cpre.bat нужно использовать команды процессора командной строки DB2, эти команды нужно поместить в отдельный файл, который будет выполняться как пакетный файл процессора командной строки из программы, которая инициализирует среду DB2 для процессора командной строки DB2 и затем ожидает завершения работы процессора командной строки DB2. Например:

```
#include <windows.h>

int WINAPI DB2SetCLPEnv_api(DWORD pid);

void main ( int argc, char *argv [ ] )
{
      STARTUPINFO         startInfo   = {0};
      PROCESS_INFORMATION pidInfo     = {0};
      char     title  [32]   = "Выполняем синхронно";
      char     runCmd [64]  =
                             "DB2 -z c:\\run.out -tvf c:\\run.clp";
/* Вызов функции API для задания среды CLP */
      if ( DB2SetCLPEnv_api (GetCurrentProcessId ()) == 0 ) 1
      {
         startInfo.cb          = sizeof(STARTUPINFO);
         startInfo.lpReserved  = NULL;
         startInfo.lpTitle     = title;
         startInfo.lpDesktop   = NULL;
         startInfo.dwX         = 0;
         startInfo.dwY         = 0;
         startInfo.dwXSize     = 0;
         startInfo.dwYSize     = 0;
         startInfo.dwFlags     = 0L;
         startInfo.wShowWindow = SW_HIDE;
         startInfo.lpReserved2 = NULL;
         startInfo.cbReserved2 = 0;
               if ( CreateProcessA( NULL,
                              runCmd, 2
                              NULL,
                              NULL,
                              FALSE,
                              NORMAL_PRIORITY_CLASS CREATE_NEW_CONSOLE,
                              NULL,
                              NULL,
                              &startInfo,
                              &pidInfo ) )
         {
            WaitForSingleObject (pidInfo.hProcess, INFINITE);
            CloseHandle (pidInfo.hProcess);
            CloseHandle (pidInfo.hThread);
```

```
        }
      }
   return;
}
```

1 Функция API DB2SetCLPEnv_api находится в библиотеке DB2API.LIB. Эта функция API задает среду, в которой можно вызывать команды процессора командной строки. Если эта программа вызывается из сценария db2cpre.bat, процессор командной строки будет ожидать завершения команд процессора командной строки.

2 runCmd - имя файла сценария, содержащего команды процессора командной строки DB2.

Пример программы с именем db2clpex.exe можно найти в подкаталоге MISC каталога установки DB2. Этот выполняемый файл подобен приведенному выше примеру программы, но он получает команду процессора командной строки DB2 в качестве аргумента своей командной строки. Если нужно использовать этот пример программы, скопируйте ее в подкаталог BIN. Этот выполняемый файл можно использовать в сценарии db2cpre.bat так (INSTHOME - каталог экземпляра):

```
db2clpex "DB2 -Z INSTHOME\pre.log -tvf INSTHOME\pre.clp"
```

Во всех командах DB2 ATTACH или операторах CONNECT нужно явно задавать пользователя; в противном случае они будут выполняться под учетной записью пользователя, связанной со службой кластеров. Сценарии процессора командной строки должны также завершаться командой TERMINATE, которая прекращает фоновый процесс процессора командной строки.

Пример файла db2cpre.bat:

```
db2cpre.bat : 1
-----------------------
db2clpex "db2 -z INSTHOME\pre-%DB2NODE%.log -tvf INSTHOME\pre.clp" 2 - 5
-----------------------

PRE.CLP 6
-----------------------
update dbm cfg using MAXAGENTS 200;
get dbm cfg;
terminate;
-----------------------
```

1 Сценарий db2cpre.bat выполняется под учетной записью пользователя, связанной со службой кластеров. Если требуется выполнение операций DB2, учетная запись пользователя, связанная со службой кластеров, должна быть правильным идентификатором SQL, удовлетворяющим требованиям DB2.

2 INSTHOME - каталог экземпляра.

3 На всех узлах файлы журналов должны иметь разные имена, чтобы избежать конфликтов при одновременной активации обоих логических узлов.

4 `db2clpex.exe` - пример программы, аргумент командной строки которой используется для задания команды процессора командной строки, которую нужно выполнить.

5 Программа `db2clpex.exe` должна быть доступной на всех узлах кластера MSCS.

6 Команды процессора командной строки в этом примере задают ограничение на число агентов.

Выполнение сценариев после активации ресурсов DB2

Если нужно выполнять сценарий *после* активации ресурса DB2, этот сценарий *должен* быть назван `db2cpost.bat`. Этот сценарий будет выполняться асинхронно по отношению к MSCS после успешной активации ресурса DB2. В этом сценарии можно использовать команду **db2cmd** для выполнения файлов сценариев процессора командной строки DB2. Используйте параметр ″-c″ команды **db2cmd**, чтобы задать, что эта утилита должна закрывать все окна по завершении задания. Например:

```
db2cmd -c db2 -tvf mycmds.clp
```

Параметр ″-c″ должен быть первым аргументом команды **db2cmd**, так как он предотвращает выполнение в фоновом режиме командных процессоров после завершения вызвавшего их процесса.

Сценарий `db2cpost.bat` удобен, если после возникновения сбоя и перезапуска ресурса DB2 на резервном компьютере нужно сразу выполнить операции с базами данных. Например, можно перезапустить или активировать базы данных этого экземпляра, чтобы сделать их доступными для доступа пользователей.

Пример сценария `db2cpost.bat`:

```
db2cpost.bat 1
------------------------
db2cmd -c db2 -z INSTHOME\post-%DB2NODE%.log -tvf INSTHOME\post.clp 2 - 4
------------------------

POST.CLP 5
------------------------
restart database SAMPLE;
connect reset;
activate database SAMPLE;
terminate;
------------------------
```

1 Сценарий `db2cpost.bat` выполняется под учетной записью пользователя, связанной со службой кластеров. Если требуется выполнение операций DB2, учетная запись пользователя, связанная со

службой кластеров, должна быть правильным идентификатором SQL, удовлетворяющим требованиям DB2.

2 INSTHOME - каталог экземпляра.

3 На всех узлах файлы журналов должны иметь разные имена, чтобы избежать конфликтов при одновременной активации обоих логических узлов.

4 Можно использовать команду **db2cmd**, так как сценарий `db2cpost.bat` может выполняться асинхронно. Необходимо использовать параметр ″-c″, вызывающий завершение работы процессора командной строки.

5 Сценарий процессора командной строки в этом примере содержит команды для перезапуска и активации базы данных. Этот сценарий опять переводит базу данных в активное состояние сразу после запуска менеджера баз данных. В системе многораздельных баз данных нужно удалить команду ACTIVATE DATABASE, поскольку одновременно активируются несколько ресурсов DB2. Команда RESTART DATABASE может завершиться неудачно, если другой узел активирует эту базу данных. Если это произошло, снова выполните этот сценарий, чтобы запустить базу данных правильно.

База данных

При создании базы данных надо расположить ее на совместно используемом диске. Это позволит базе данных быть видимой со всех узлов MSCS. Для успешного восстановления после отказов DB2 все журналы и другие файлы базы данных также должны располагаться на кластеризованных дисках. Если это не сделано, возникнет системная ошибка DB2, поскольку DB2 решит, что эти файлы были удалены или недоступны.

Параметры конфигурации базы данных и менеджера баз данных надо задать так, чтобы используемый DB2 объем системных ресурсов был доступен на всех узлах MSCS. Параметр конфигурации базы данных *autorestart* должен иметь значение `ON`, чтобы первое соединение с базой данных при восстановлении после отказа переводило базу данных в совместимое состояние. (По умолчанию для *autorestart* используется значение `ON`.) Базу данных можно также перевести в состояние готовности, используя сценарий `db2cpost.bat` для перезапуска и активации базы данных. Это предпочтительный метод, поскольку он не зависит от значения параметра *autorestart* и база данных будет переведена в состояние готовности независимо от пользовательского запроса соединения.

Поддержка пользователей и групп

DB2 использует средства Windows NT для аутентификации пользователей и поддержки групп. Чтобы экземпляр DB2 мог в случае отказа на одном узле MSCS нормально запускаться на другом узле, все узлы MSCS должны иметь доступ к одним и тем же базам данных защиты Windows NT. Для этого можно использовать защиту доменов Windows NT.

Определите всех пользователей и все группы DB2 в базе данных защиты домена. Узлы MSCS должны быть членами этого домена или этот домен должен быть доверенным доменом. Затем DB2 будет использовать базу данных защиты доменов для поддержки аутентификации и групп, независимо от того, на каком узле MSCS работает DB2.

Если используются локальные учетные записи, их надо скопировать на все узлы MSCS. Такой подход использовать не рекомендуется, поскольку он увеличивает вероятность ошибок и требует более сложного обслуживания.

В качестве режима аутентификации поддерживается также защита DCE, если все узлы MSCS являются клиентами в одной ячейке DCE.

Для службы MSCS нужно задать учетную запись пользователя, соответствующую правилам именования DB2. Это позволит службе MSCS выполнять действия с DB2, которые могут потребоваться в сценариях `db2cpre.bat` и `db2cpost.bat`.

Дополнительную информацию о поддержке пользователей и групп в Windows NT смотрите в главе ″Аутентификация пользователей при работе с DB2 for Windows NT″ книги *Руководство администратора: Реализация*.

Связь

В среде MSCS DB2 поддерживает два протокола локальной сети:

- TCP/IP
- NetBIOS

Протокол TCP/IP поддерживается, поскольку он является поддерживаемым типом ресурса кластера. Чтобы разрешить DB2 использовать TCP/IP в качестве протокола связи для системы многораздельной базы данных, создайте ресурс IP-адреса и поместите его в ту же группу, что и ресурс DB2, представляющий сервер раздела базы данных, который должен использоваться в качестве узла координатора для удаленных прикладных программ. Затем с помощью инструмента администратора кластеров создайте зависимость, чтобы этот ресурс IP активировался до запуска ресурса DB2. После этого клиенты DB2 могут внести в каталог записи каталога узлов TCP/IP для использования этого адреса TCP/IP.

Порт TCP/IP, связанный с параметром конфигурации менеджера баз данных *svcename*, должен быть зарезервирован для использования экземпляром DB2 на всех компьютерах, входящих в этот экземпляр. Кроме этого, в файле `services` на всех компьютерах с этим номером порта должно быть связано одно и то же имя службы.

Хотя протокол NetBIOS и не является поддерживаемым ресурсом кластера, его можно использовать в качестве протокола локальной сети, поскольку этот

протокол обеспечивает уникальность имен NetBIOS в локальной сети. Когда DB2 регистрирует имя NetBIOS, NetBIOS гарантирует, что это имя еще не используется в сети. В сценарии восстановления после отказов при перемещении DB2 с одной системы на другую для используемого DB2 имени *nname* будет отменена регистрация на одном компьютере партнера в кластере MSCS и затем оно будет зарегистрировано на другом компьютере.

Поддержка DB2 NetBIOS использует NetBIOS Frames (NBF). Этот стек протокола может быть связан с различными номерами логических адаптеров (LANA). Для обеспечения согласованного доступа NetBIOS к серверу на всех узлах кластера со стеком протокола NBF должен быть связан один и тот же LANA. Такую конфигурацию можно задать с помощью опции **Сеть** в Панели управления. Свяжите NBF с LANA 0, поскольку это значение по умолчанию, ожидаемое DB2.

Системное время

Для отметки времени определенных операций DB2 использует системное время. На всех узлах MSCS, участвующих в восстановлении после отказов DB2, должны быть синхронизированы системное время и временной пояс, чтобы работа DB2 была согласована на всех компьютерах.

Для задания системного часового пояса используйте опцию **Дата/Время** Панели управления. В MSCS есть служба времени, синхронизирующая дату и время при объединении узлов MSCS в кластер. Однако эта служба времени выполняет синхронизацию только раз в 12 часов, что может вызвать ошибки, если в одной из систем было изменено время и отказ DB2 возник до синхронизации выполнения времени.

Если время изменено на одном из узлов кластера MSCS, следует вручную синхронизировать время на других узлах кластера с помощью команды:

```
net time /set /y \\удаленный_узел
```

Где *удаленный_узел* - имя компьютера узла кластера.

Сервер администратора и Центр управления в среде многораздельных баз данных

При установке DB2 Universal Database может (необязательно) создаваться сервер администратора DB2. Это так для системы одноразделных баз данных. Центр управления использует службы, поставляемые сервером администратора, для управления экземплярами и базами данных DB2.

В системе многораздельных баз данных экземпляр DB2 может располагаться на нескольких узлах MSCS. Это означает, что экземпляр DB2 должен быть внесен в каталог на нескольких системах под Центром управления DB2, чтобы этот экземпляр оставался доступен независимо от того, на каком узле MSCS активен этот экземпляр DB2.

Каталог экземпляра сервера администратора не используется совместно. На всех узлах MSCS необходимо создать зеркальную копию всех пользовательских файлов из каталога сервера администратора, чтобы обеспечить один и тот же уровень управления для всех узлов MSCS. В частности, на всех узлах должны быть доступны пользовательские сценарии и внесенные в расписание выполняемые файлы. Кроме этого, на всех компьютерах в кластере MSCS в расписание должны быть внесены одни и те же операции.

Вместо того, чтобы дублировать сервер администратора на всех компьютерах, можно использовать сервер администратора в режиме восстановления после отказов. Для следующего примера предположим, что в кластере есть два узла MSCS с именами MACH0 и MACH1. Узел MACH0 имеет доступ к диску кластера, который будет использоваться сервером администратора. Предположим также, что на обоих узлах MACH0 и MACH1 есть сервер администратора. Для обеспечения высокой доступности сервера администратора выполните следующие шаги:

1. Остановите сервер администратора на обоих компьютерах, введя на каждом компьютере команду **db2admin stop**.
2. На всех компьютерах клиентов администратора с помощью команды UNCATALOG NODE удалите из каталога все ссылки на серверы администратора на узлах MACH0 и MACH1. (Чтобы проверить наличие ссылок на сервер администратора, можно использовать команду LIST NODE DIRECTORY на компьютере клиента.)
3. Отбросьте сервер администратора с узла MACH1, введя на узле MACH1 команду **db2admin drop**. (Этот шаг нужно выполнить, только если сервер администратора есть на обоих компьютерах.)
4. Узнайте имя сервера администратора, введя на узле MACH0 команду **db2admin**. (Имя по умолчанию - DB2DAS00.)
5. Используйте утилиту DB2MSCS, чтобы настроить поддержку восстановления после отказов для сервера администратора. При этом в MSCS будет создан ресурс DB2 с именем DB2DAS00, имеющий зависимости от ресурсов IP и дисковых ресурсов. (В конфигурации взаимной подмены этот ресурс нужно поместить в группу, в которую входит ресурс DB2 для NODE0.) Этот ресурс будет использоваться в качестве ресурса MSCS, поддерживающего сервер администратора. Файл DB2MSCS.ADMIN должен выглядеть так:

```
#
# db2mscs.admin для сервера администратора
# run db2mscs -f:db2mscs.admin
#
DB2_INSTANCE=DB2DAS00
CLUSTER_NAME=CLUSTERA
DB2_LOGON_USERNAME=db2admin
DB2_LOGON_PASSWORD=db2admin
# помещаем сервер администратора в ту же группу, что и узел 0 DB2
GROUP_NAME=DB2NODE0  1
```

```
DISK_NAME=DISK E:
INSTPROF_DISK=DISK E:
IP_NAME= IP Address for Administration Server
IP_ADDRESS=9.9.9.8
IP_SUBNET=255.255.255.0
IP_NETWORK=Ethernet
```

1 Эта группа может совпадать с существующей группой. При этом не требуется дополнительный диск для каталога профиля экземпляра.

6. На узле MACH1 введите следующую команду, чтобы задать DB2DAS00 в качестве сервера администратора:

   ```
   db2set -g db2adminserver=DB2DAS00
   ```

7. На узле MACH0 с помощью программы Службы измените свойства запуска DB2DAS00, чтобы он запускался вручную, а не автоматически, поскольку теперь DB2DAS00 контролируется MSCS.

Если сервер администратора работает в режиме восстановления после отказов, все удаленные обращения к серверу администратора должны для связи использовать ресурс IP MSCS. Теперь сервер администратора будет иметь следующие свойства:

- При отказах вместе с сервером администратора на другой узел будет переноситься каталог экземпляров сервера администратора.
- На клиентах для связи с сервером администратора в каталог будет вноситься только один узел, независимо от того, на каком узле MSCS активен сервер администратора.
- Задания для сервера администратора нужно будет внести в расписание только один раз.
- Сервер администратора можно будет использовать для управления локальными экземплярами, только когда сервер администратора будет активен на том же узле, что и локальный экземпляр.
- Сервер администратора недоступен, если неактивна служба кластеров.

Ограничения

При работе DB2 в среде MSCS:

- Нельзя использовать физический ввод-вывод для совместно используемых дисков (если только эти совместно используемые диски не имеют одни и те же номера физических дисков на обоих узлах MSCS). Можно использовать логический ввод-вывод, поскольку при этом для обращения к диску используется идентификатор раздела.
- Все ресурсы DB2 должны быть сконфигурированы для поддержки MSCS. Если это не сделано, во время выполнения DB2 будут возникать ошибки (DB2 не сможет правильно работать при отсутствии системных ресурсов). Например, если журналы базы данных находятся не на совместно используемом диске MSCS, DB2 не сможет перезапустить базу данных.

- Для управления экземпляром DB2 необходимо использовать инструмент администратора кластеров. MSCS будет рассматривать применение других способов запуска и остановки менеджера баз данных как программные сбои. Например, если использовать MSCS для запуска DB2 и команду **db2stop** для остановки DB2, MSCS воспримет эту операцию как программный сбой и перезапустит этот экземпляр. Это означает также, что для запуска и остановки DB2 нельзя использовать Центр управления.
- Для деинсталляции DB2 необходимо сначала остановить MSCS.

Глава 14. DB2 и высокая доступность в Sun Cluster 2.2

Эта глава подробно описывает, как достигается высокая доступность при работе DB2 с Sun Cluster 2.x (SC2.x); она содержит описание агента высокой доступности, действующего как посредник между двумя программными продуктами (смотрите рис. 59).

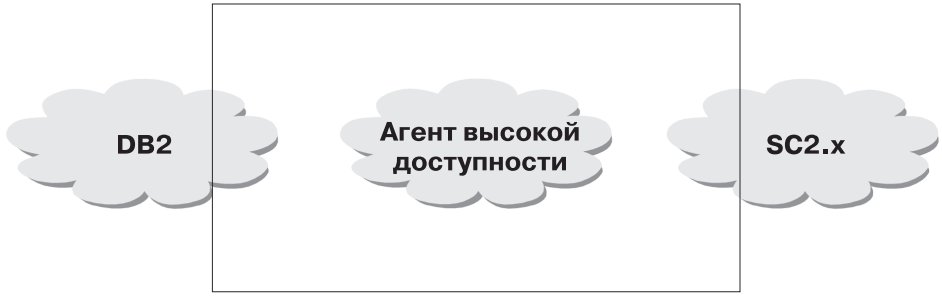


Рисунок 59. DB2, Sun Cluster 2.x и высокая доступность

Системы высокой доступности

Компьютерные системы, где размещаются службы данных, содержат множество различных компонентов, каждый из которых характеризуется своей ″средней наработкой на отказ″ (mean time before failure, MTBF). Наработка на отказ - это среднее время, в течение которого данный компонент остается исправным. Для качественного жесткого диска MTBF составляет порядка миллиона часов (приблизительно 114 лет). Хотя этот срок может показаться долгим, из каждых 200 дисков один, скорее всего, откажет уже через 6 месяцев.

Для увеличения доступности службы данных существуют разные методы, из которых чаще всего применяется метод кластеров высокой доступности. Кластер, обеспечивающий высокую доступность, состоит из двух или более компьютеров, набора интерфейсов собственной сети, одного или нескольких интерфейсов общедоступной сети и нескольких совместно используемых дисков. Эта особая конфигурация позволяет перемещать службы данных с одного компьютера на другой. Благодаря такой возможности служба данных сохраняет возможность обращаться к своим данным. Перемещение службы данных с одного компьютера на другой называется *восстановлением после сбоев*, как показано на рис. 60 на стр. 308.

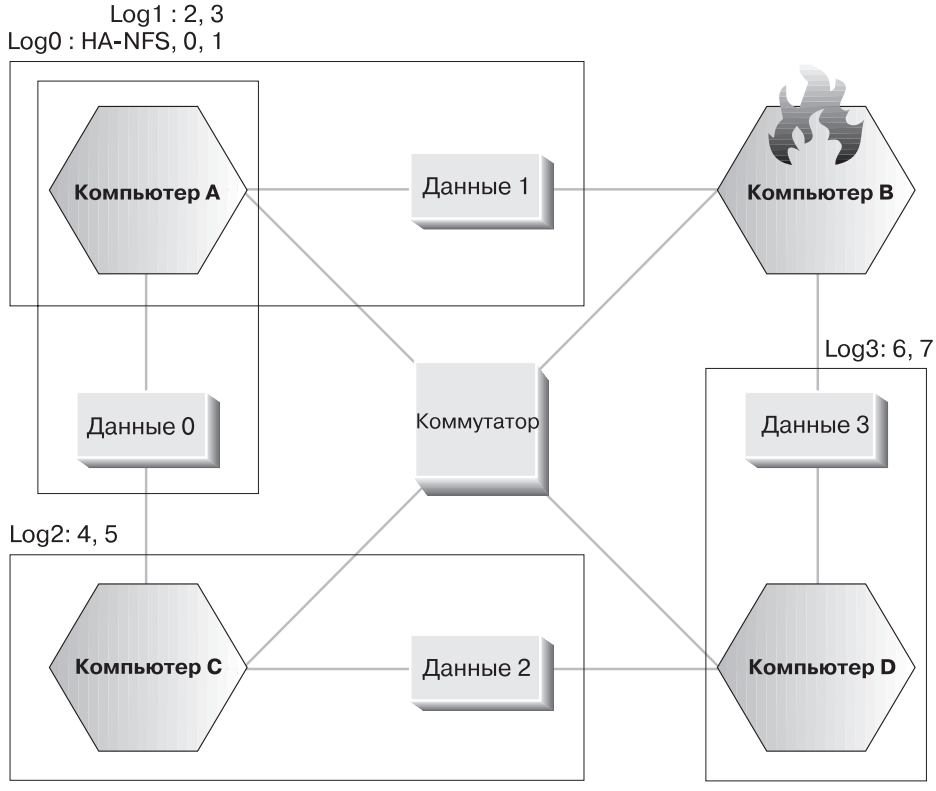


Рисунок 60. Восстановление после сбоев

Интерфейсы собственной сети служат для обмена сообщениями *работоспособности*, а также управляющими сообщениями между компьютерами в кластере. Интерфейсы общедоступной сети используются для непосредственной связи с клиентами кластера высокой доступности. Диски в кластере высокой доступности соединены с двумя или более компьютерами кластера, поэтому в случае отказа одного из компьютеров другие компьютеры сохраняют к ним доступ.

Служба данных, запущенная в кластере высокой доступности, имеет один или несколько логических интерфейсов общедоступной сети и набор связанных с ними дисков. Клиенты службы данных высокой доступности соединяются через протокол TCP/IP только с логическими сетевыми интерфейсами службы данных. В случае сбоя служба данных вместе со своими логическими сетевыми интерфейсами и набором дисков перемещается на другой компьютер.

Одно из преимуществ кластера высокой доступности состоит в том, что службы данных можно восстанавливать без помощи персонала поддержки, причем в любое время. Другое преимущество заключается в дублировании. Все части

кластера, включая сами компьютеры, должны быть продублированы. Кластер должен быть рассчитан на успешное преодоление ошибки в любой отдельной точке.

Как бы ни различались службы данных высокой доступности по своей природе, они подчиняются некоторым общим требованиям. Клиенты службы данных высокой доступности исходят из неизменности ее сетевого адреса и имени хоста и рассчитывают, что они будут посылать свои требования в неизменном виде, на каком бы компьютере ни находилась эта служба данных.

Рассмотрим Web-браузер, обращающийся к высокодоступному Web-серверу. При этом требование посылается с URL, который содержит как имя хоста, так и путь к файлу на Web-сервере. Браузер ожидает, что имя хоста и путь к файлу не изменятся после отказа на Web-сервере. Если при загрузке файла с Web-сервера происходит сбой, браузеру придется послать требование повторно.

Доступность служб данных измеряется количеством времени, в течение которого служба доступна пользователям. Обычно единицей измерения доступности служит процент времени доступности, который часто выражают количеством девяток:

```
99,99% => служба недоступна (максимум) 52,6 минут в год
99,999% => служба недоступна (максимум) 5,26 минут в год
99,9999% => служба недоступна (максимум) 31,5 секунд в год
```

При проектировании и тестировании кластера высокой доступности:

1. Администратор кластера должен быть знаком с системой и с процессом восстановления после отказа.
2. Убедитесь, что каждая часть кластера надежно продублирована и может быть быстро заменена в случае ее отказа.
3. Организуйте принудительные отказы на тестовой системе в контролируемой среде и проверьте, правильно ли происходит каждый раз восстановление.
4. Сохраняйте при восстановлении информацию о причинах каждого отказа. Хотя частой необходимости в этом не ожидается, важно обращать внимание на любые возможные источники нестабильной работы кластера. Например, если некоторый элемент кластера вызывал пятикратное восстановление после отказа в течение месяца, найдите причину этого и устраните ее.
5. Персонал поддержки данного кластера должен знать о произошедшем отказе.
6. Не перегружайте кластер. Убедитесь, что оставшиеся системы после отказа и восстановления в состоянии выдерживать приемлемый уровень рабочей нагрузки.
7. Часто проверяйте компоненты, склонные к отказам (например, диски), для своевременной их замены.

Отказоустойчивость и непрерывная доступность

Другой способ повышения доступности службы данных - увеличение устойчивости к отказам. *Отказоустойчивый компьютер* имеет встроенные резервные элементы и должен переносить отказ любой части, включая центральный процессор и память. Отказоустойчивые компьютеры часто используются там, где это необходимо и, как правило, очень дороги. Кластер высокой доступности, компьютеры которого размещены в разных географических точках, обладает дополнительными преимуществами, поскольку он способен к восстановлению после локальных аварий и стихийных бедствий.

Постоянная доступность - шаг вперед по сравнению с высокой доступностью. Она ограждает клиентов как от запланированных, так и от незапланированных простоев. При конфигурации постоянной доступности на работу клиента никак не влияет отказ одного из компьютеров, на котором располагаются службы данных, или его выключение для обслуживания. Конфигурации постоянной доступности сложны и дороги.

Кластер высокой доступности - наиболее общее решение для повышения доступности, так как он масштабируем, прост в использовании и относительно недорог.

Sun Cluster 2.2

Sun Cluster 2.2 (SC2.2) - это продукт компании Sun Microsystems для обеспечения кластеризации и высокой доступности. SC2.2 поддерживает до четырех компьютеров в одном кластере. Если используется 4 компьютера Ultra Enterprise 10000, кластер может иметь до 256 процессоров и 256 Гбайт оперативной памяти.

Поддерживаемые системы

Система	UltraSPARC	Емкость памяти	Параллелизм ввода/вывода
Ultra Enterprise 1	1	64 Мбайта - 1 Гбайт	3 SBus
Ultra Enterprise 2	1-2	64 Мбайта - 2 Гбайта	4 SBus
Ultra Enterprise 450	1-4	32 Мбайта - 4 Гбайта	10 PCI
Ultra Enterprise 3000	1-6	64 Мбайта - 6 Гбайт	9 SBus
Ultra Enterprise 4000	1-14	64 Мбайта - 14 Гбайт	21 SBus
Ultra Enterprise 5000	1-14	64 Мбайта - 14 Гбайт	21 SBus

Ultra Enterprise 6000	1-30	64 Мбайта - 30 Гбайт	45 SBus
Ultra Enterprise 10000	1-64	512 Мб - 64 Гб	64 SBus

Агенты

Программное обеспечение Sun Cluster включает ряд агентов высокой доступности, которые поддерживаются продуктом SC2.2 и поставляются вместе с ним. Другие агенты высокой доступности, например агент для DB2, разрабатываются не компанией Sun и не поставляются вместе с программным обеспечением Sun Cluster. Агент высокой доступности для DB2 поставляется вместе с DB2 и поддерживается IBM.

Программное обеспечение Sun Cluster работает с высокодоступными службами данных, обеспечивая возможность регистрации методов (сценариев или программ), соответствующих компонентам программного обеспечения Sun Cluster. При помощи этих методов программное обеспечение SC2.2 способно управлять службой данных, не зная подробностей ее работы. В число этих методов входят:

START
Используется для запуска частей службы данных до подключения логических сетевых интерфейсов.

START_NET
Используется для запуска частей службы данных после подключения логических сетевых интерфейсов.

STOP Используется для остановки частей службы данных после отключения логических сетевых интерфейсов.

STOP_NET
Используется для остановки частей службы данных до отключения логических сетевых интерфейсов.

ABORT
Подобен методу STOP, но запускается непосредственно перед выключением компьютера программными средствами кластера. В этом случае под вопросом ″здоровье″ компьютера, и службе данных может понадобиться выполнить требования ″последнего желания″ до выключения компьютера. Запускается после отключения логических сетевых интерфейсов.

ABORT_NET
Подобен методу ABORT, но запускается до отключения логических сетевых интерфейсов.

FM_INIT
Используется для инициализации мониторов отказов.

FM_START
Используется для запуска мониторов отказов.

FM_STOP
Используется для остановки мониторов отказов.

FM_CHECK
Вызывается командой **hactl**. Возвращает текущее состояние соответствующей службы данных.

Агент DB2 состоит из следующих сценариев: START_NET, STOP_NET, FM_START и FM_STOP. Следующие сценарии не запускаются во время повторного конфигурирования кластера: ABORT, ABORT_NET и FM_CHECK.

Агент высокой доступности состоит из одного или нескольких перечисленных методов. Методы регистрируются с SC2.2 при помощи команды **hareg**. После регистрации программные средства Sun Cluster уже могут вызывать соответствующий метод для управления службой данных.

Важно помнить, что методы службы ABORT и STOP вызывать нельзя. Эти методы предназначены для управляемого закрытия службы данных, и служба данных должна быть способна к восстановлению при отказе компьютера без их вызова.

Дополнительную информацию смотрите в документации по Sun Cluster.

Логические хосты

В программном обеспечении SC2.2 реализована идея логического хоста. *Логический хост* состоит из набора дисков и одного или нескольких логических интерфейсов общедоступной сети. Высокодоступная служба данных связана с логическим хостом и требует дисков, входящих в группу дисков логического хоста. Логические хосты могут размещаться на разных компьютерах кластера и ″заимствовать″ процессоры и память компьютера, на котором они запущены.

Логические сетевые интерфейсы

Как и другие операционные системы на основе UNIX, кроме основного адреса для сетевого интерфейса Solaris способен иметь дополнительные IP-адреса. Дополнительные IP-адреса связаны с логическим интерфейсом так же, как первичный IP-адрес - с физическим сетевым интерфейсом. Ниже приведен пример логических интерфейсов на двух компьютерах в кластере. Есть два логических хоста, которые в настоящий момент оба находятся на компьютере под именем ″thrash″.

```
scadmin@crackle(202)# netstat -in
Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis Queue
lo0 8232 127.0.0.0 127.0.0.1 289966 0 289966 0 0 0
hme0 1500 9.21.55.0 9.21.55.98 121657 6098 764122 0 0 0
scid0 16321 204.152.65.0 204.152.65.1 489307 0 476479 0 0 0
scid0:1 16321 204.152.65.32 204.152.65.33 0 0 0 0 0 0
```

```
scid1 16321 204.152.65.16 204.152.65.17 347317 0 348073 0 0 0
```

1. lo0 - интерфейс обратной связи
2. hme0 - интерфейс общедоступной сети (ethernet)
3. scid0 - первый интерфейс собственной сети (SCI - Scalable Coherent Interface)
4. scid0:1 - логический сетевой интерфейс, который программные средства Sun Cluster используют как внутренний
5. scid1 - второй интерфейс собственной сети

```
scadmin@thrash(203)# netstat -in
Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis Queue
lo0 8232 127.0.0.0 127.0.0.1 1128780 0 118780 0 0 0
hme0 1500 9.21.55.0 9.21.55.92 1741422 5692 757127 0 0 0
hme0:1 1500 9.21.55.0 9.21.55.109 0 0 0 0 0 0
hme0:2 1500 9.21.55.0 9.21.55.110 0 0 0 0 0 0
scid0 16321 204.152.65.0 204.152.65.2 476641 0 489476 0 0 0
scid0:1 16321 204.152.65.32 204.152.65.34 0 0 0 0 0 0
scid1 16321 204.152.65.16 204.152.65.18 348199 0 347444 0 0 0
```

1. hme0:1 - логический сетевой интерфейс для логического хоста
2. hme0:2 - логический сетевой интерфейс для другого логического хоста

С логическим хостом может быть связан один или несколько логических интерфейсов. Эти логические интерфейсы перемещаются вместе с логическим хостом с одного компьютера на другой и используются для обращения к службе данных, связанной с логическим хостом. Поскольку эти логические интерфейсы перемещаются вместе с логическими хостами, клиенты могут обращаться к службе данных независимо от компьютера, на котором она находится.

Высокодоступная служба данных должна быть связана с адресом TCP/IP INADDR_ANY. Благодаря этому каждый IP-адрес в системе может принимать соединения для службы данных. Если вместо этого служба данных связывается с конкретным IP-адресом, ей необходимо связать логический интерфейс, ассоциированный с логическим хостом, где размещена эта служба данных. Связывание с INADDR_ANY также устраняет необходимость связываться с новым IP-адресом, если он появляется в системе, которая требуется службе данных.

Примечание: Клиенты экземпляра высокой доступности должны каталогизировать базу данных, используя имя хоста для логического IP-адреса логического хоста. Ни в коем случае нельзя использовать для обозначения компьютера имя первичного хоста, так как нет никакой гарантии, что DB2 будет запущена именно на этом компьютере.

Группы дисков и файловые системы

Диски для службы данных составляют связанные с логическим хостом группы (или наборы). Если в кластере запущен Sun StorEdge Volume Manager (Veritas), программные средства Sun Cluster используют для импорта групп дисков для

каждого логического хоста и их удаления утилиту Veritas ″vxdg″. Ниже приводится пример групп дисков для двух логических хостов ″log0″ и ″log1″ на компьютере ″thrash″. На компьютере ″crackle″ в данный момент нет логических хостов.

```
scadmin@crackle(206)# vxdg list
NAME STATE ID
rootdg enabled 899825206.1025.crackle

scadmin@thrash(205)# vxdg list
NAME STATE ID
rootdg enabled 924176206.1025.thrash
data0 enabled 925142028.1157.crackle=
data1 enabled 899826248.1108.crackle
```

Группы дисков ″data0″ и ″data1″ соответствуют логическим хостам ″log0″ и ″log1″. Группу дисков ″data0″ можно удалить с компьютера ″thrash″ с помощью команды

```
vxdg deport data0
```

и импортировать на компьютер ″crackle″ с помощью команды

```
vxdg import data1
```

Программные средства Sun Cluster выполняют это автоматически, делать это вручную в активном кластере не следует.

Каждая группа дисков содержит некоторое количество дисков, которые могут быть поделены между двумя или несколькими компьютерами в кластере. Логический хост можно переместить только на компьютер, имеющий физический доступ к дискам из относящихся к нему групп дисков.

Управление файловыми системами для каждого логического хоста осуществляется с помощью двух файлов:

```
/etc/opt/SUNWcluster/conf/hanfs/vfstab.<логический_хост>
/etc/opt/SUNWcluster/conf/hanfs/dfstab.<логический_хост>
```

где *логический_хост* - имя связанного с системой логического хоста.

Файл `vfstab` подобен файлу `/etc/vfstab`, но содержит, в отличие от него, записи о файловых системах, которые надо смонтировать после импорта групп дисков для логического хоста. Файл `dfstab` подобен файлу `/etc/dfs/dfstab`, но содержит, в отличие от него, записи о файловых системах, которые надо экспортировать посредством HA-NFS для логического хоста. На каждом компьютере есть собственная копия этих файлов, и необходимо, чтобы их содержимое на разных компьютерах, входящих в кластер, совпадало.

Примечание: Пути к файлам `vfstab` и `dfstab` логического хоста не соответствуют действительности, так как содержат каталог `hanfs`.

Для HA-NFS используется только файл dfstab логического хоста. Файл vfstab используется, даже если HA-NFS не конфигурирована.

Ниже приведены примеры для кластера, где запущена DB2 Universal Database Enterprise - Extended Edition (EEE) в конфигурации взаимной подмены:

```
scadmin@thrash(217)# ls -l /etc/opt/SUNWcluster/conf/hanfs
total 8
-rw-r--r-- 1 root build 173 Apr 14 15:01 dfstab.log0
-rw-r--r-- 1 root build 316 Apr 26 12:07 vfstab.log0
-rw-r--r-- 1 root build 389 Apr 13 21:04 vfstab.log1

scadmin@thrash(218)# cat dfstab.log0
share -F nfs -o root=crackle:thrash:\
jolt:bump:crackle.torolab.ibm.com:thrash.torolab.ibm.com:\
jolt.torolab.ibm.com:bump.torolab.ibm.com /log0/home
```

Хосты, получившие разрешение монтировать файловую систему /log0/home, могут относиться ко всем сетевым интерфейсам (логическим и физическим) на каждом компьютере кластера. Файловые системы экспортируются с разрешениями root.

```
scadmin@thrash(220)# cat vfstab.log0
#монтируемое устр-во        устр-во для fsck      точка     тип прох. монт.опц.
#                                                 монтир.    FS fsck  при
                                                                      загр.

/dev/vx/dsk/data0/data1-stat                     /log0       ufs  2    no   -
                      /dev/vx/rdsk/data0/data1-stat
/dev/vx/dsk/data0/vol01 /dev/vx/rdsk/data0/vol01 /log0/home  ufs  2    no   -
/dev/vx/dsk/data0/vol02 /dev/vx/rdsk/data0/vol02 /log0/data  ufs  2    no   -

scadmin@thrash(221)# cat vfstab.log1
#монтируемое устр-во        устр-во для fsck      точка     тип прох. монт.опц.
#                                                 монтир.   FS  fsck  при
                                                                      загр.

/dev/vx/dsk/data1/data1-stat                     /log1       ufs  2    no   -
                      /dev/vx/rdsk/data1/data1-stat
/dev/vx/dsk/data1/vol01 /dev/vx/rdsk/data1/vol01 /log1/home  ufs  2    no   -
/dev/vx/dsk/data1/vol02 /dev/vx/rdsk/data1/vol02 /log1/data  ufs  2    no   -
/dev/vx/dsk/data1/vol03 /dev/vx/rdsk/data1/vol03 /log1/data1 ufs  2    no   -
```

Файл vfstab.log0 содержит три допустимых записи для файловых систем каталога /log0. Обратите внимание на то, что файловые системы для логического хоста log0 используют устройства логического тома, которые входят в группу дисков data0, связанную с логическим хостом.

Файловые системы в файлах vfstab монтируются в порядке следования сверху вниз, поэтому важно, чтобы они были перечислены в правильном порядке. Файловые системы, которые монтируются под конкретной файловой системой,

должны стоять ниже ее в списке. Реальные файловые системы, которые требуются логическому хосту, зависят от потребностей службы данных и могут значительно отличаться от приведенных примеров.

Во время восстановления после сбоя программные средства SC2.2 отвечают за то, чтобы группы дисков и логические интерфейсы, связанные с логическим хостом, следовали за ним по всему кластеру от компьютера к компьютеру. Ожидается, что после восстановления высокодоступная служба данных будет располагать по крайней мере этими ресурсами в новой системе. Фактически многие службы данных даже не "знают" о своей высокой доступности, поэтому эти ресурсы должны быть в точности такими же, как перед восстановлением после сбоя.

Методы управления

Методы управления регистрируются с помощью команды

```
hareg(1m)
```

Когда служба высокой доступности зарегистрирована, SC2.2 начинает отвечать за вызов методов, которые были зарегистрированы для службы высокой доступности в соответствующие моменты во время повторного конфигурирования кластера или восстановления после сбоя.

Во время повторного конфигурирования кластера (управляемого восстановления) выполняются следующие действия (в приведенном порядке). Действия, предшествующие шагу 5c, не предпринимаются в случае отказа компьютера. (Дополнительную информацию о повторном конфигурировании кластера смотрите в документации по SC2.2.)

```
 1. Запускается метод FM_STOP.
 2. Запускается метод STOP_NET.
 3. Отключаются логические интерфейсы для логического хоста.
      - ifconfig hme0:1 0.0.0.0 down
 4. Запускается метод STOP.
 5. Перемещаются группы дисков и файловые системы.
      a. Демонтируются файловые системы логического хоста.
      b. vxdg удаляет группы дисков с одного из компьютеров.

    - - При отказе компьютера выполняются только шаги, указанные ниже - -

      c. vxdg импортирует группы дисков на другой компьютер.
      d. Производится проверка (fsck) файловых систем логического хоста.
      e. Монтируются файловые системы логического хоста.
 6. Запускается метод START.
 7. Подключаются логические интерфейсы для логического хоста.
      - ifconfig hme0:1 <ip-адрес> up
 8. Запускается метод START_NET.
 9. Запускается метод FM_INIT.
10. Запускается метод FM_START.
```

Методы управления запускаются с помощью следующих аргументов командной строки:

```
METHOD <установленные логические хосты> <неустановленные логические хосты>
       <срок ожидания>
```

Первый аргумент - это список через запятую установленных на данный момент логических хостов, а второй - список через запятую неустановленных логических хостов. Последний аргумент - это время ожидания для данного метода, то есть интервал времени, в течение которого этому методу разрешается работать до того, как его выполнение будет прекращено программными средствами SC2.2.

Конфигурация дисков и файловой системы

SC2.2 поддерживает два менеджера томов: Sun StorEdge Volume Manager (Veritas) и Solstice Disk Suite. Хотя оба менеджера работают хорошо, StorEdge Volume Manager имеет некоторые преимущества в кластерной среде. При некоторых конфигурациях кластера число контроллеров для оболочки диска на каждом из компьютеров кластера может быть различным. При разном числе контроллеров пути к дисковым устройствам для контроллера также будут различаться. Поскольку Disk Suite работает непосредственно с путями дисковых устройств, в этой ситуации он не будет хорошо работать. StorEdge Volume Manager работает с самими дисками независимо от числа контроллеров и не реагирует на различия числа контроллеров.

Поскольку цель высокой доступности состоит в увеличении доступности службы данных, важно убедиться, что для всех файловых систем и дисковых устройств используются зеркальные копии или конфигурация RAID. Это предотвратит восстановления, вызванные отказами дисков, и увеличит стабильность кластера.

HA-NFS

Если экземпляр конфигурируется на нескольких компьютерах, для DB2 UDB EEE требуется совместно используемая файловая система. В типичной конфигурации DB2 UDB EEE есть домашний каталог, экспортированный с одного из компьютеров через NFS и смонтированный на всех компьютерах экземпляра EEE. В случае конфигурации взаимной подмены DB2 UDB EEE обеспечивает совместно используемую высокодоступную файловую систему при помощи HA-NFS. Один из логических хостов экспортирует файловую систему с помощью HA-NFS, а каждый компьютер в кластере монтирует затем эту файловую систему как домашний каталог экземпляра EEE. Дополнительную информацию о HA-NFS смотрите в документации по Sun Cluster.

Утилиты cconsole и ctelnet

Вместе с SC2.2 поставляются две полезные утилиты - `cconsole` и `ctelnet`. Эти утилиты можно использовать для отправки одной команды одновременно нескольким компьютерам в кластере. Редактирование файла конфигурации при помощи этих утилит обеспечивает его идентичность на всех компьютерах, входящих в кластер. Данные утилиты можно также использовать для установки

программного обеспечения на каждый компьютер одинаковым образом. Дополнительную информацию об этих утилитах смотрите в документации по Sun Cluster.

Микрорайонная кластеризация и межрегиональная кластеризация

Кластер называется *микрорайонным кластером*, если его компьютеры расположены в разных зданиях. Микрорайонный кластер позволяет исключить само здание как единую точку ошибки. Например, если компьютеры кластера находятся в одном и том же здании, в случае пожара пострадает весь кластер. Когда же компьютеры расположены в разных зданиях, даже если одно из них сгорит, кластер сохранится.

Межрегиональный кластер - это кластер, компьютеры которого размещены в разных городах. В этом случае преследуется уже цель исключить целый географический регион как единую точку ошибки. Этот тип кластеров обеспечивает устойчивость к таким катастрофическим событиям, как наводнения или землетрясения.

В настоящее время Sun Cluster в состоянии поддерживать работу компьютеров, удаленных друг от друга на 10 км. Это делает микрорайонную кластеризацию практичным решением для тех, кому требуется высокоскоростное соединение между двумя удаленными точками. Кластер требует наличия двух собственных межсоединений и нескольких волоконно-оптических кабелей для совместно используемых дисков. Затраты на высокоскоростное соединение между разными городами могут и не окупиться.

Общие проблемы

Чтобы обеспечить единое для всего кластера место хранения его конфигурации, программные средства SC2.2 используют базу данных конфигурации кластера Cluster Configuration Database или CCD(4). Эта CCD имеет собственный API и хранится в каталоге `/etc/opt/SUNWcluster/conf`. В редких случаях CCD может выйти из синхронизации и потребовать устранения неисправности. Лучший способ восстановления CCD в этой ситуации - восстановление ее из резервной копии.

Чтобы создать резервную копию CCD, закройте программные средства кластера на всех его компьютерах, запакуйте каталог `/etc/opt/SUNWcluster/conf` и сохраните tar-файл в безопасном месте. Если работа программных средств кластера не прекращалась при создании резервной копии CCD, у вас могут возникнуть трудности с ее восстановлением. Обеспечьте наличие свежей резервной копии, обновляя ее каждый раз при изменении конфигурации кластера. Для восстановления CCD закройте программные средства кластера на всех его компьютерах, переместите каталог `conf` в `conf.old` и распакуйте резервную копию. После этого можно запустить кластер с новой CCD.

Особенности DB2

В этом разделе обсуждаются следующие темы:

Прикладные программы, соединяющиеся с экземпляром высокой доступности

Прикладные программы, использующие высокодоступный экземпляр DB2, должны быть способны к повторному соединению в случае восстановления после отказа. Поскольку имя и IP-адрес логического хоста не изменяются, соединяться с другим именем хоста или повторно каталогизировать базу данных нет необходимости.

Рассмотрим кластер с двумя компьютерами и одним экземпляром DB2 Universal Database Enterprise Edition (EE). Экземпляр EE обычно находится на одном из компьютеров кластера. Клиенты экземпляра высокой доступности соединяются с логическим IP-адресом (или именем хоста) логического хоста, связанного с экземпляром высокой доступности.

С точки зрения клиента высокой доступности, существует два типа восстановления после отказов. Один имеет место при аварии компьютера, на котором установлен экземпляр высокой доступности. При восстановлении второго типа экземпляр высокой доступности имеет возможность корректно завершить работу.

Если экземпляр высокой доступности прекращает работу из-за аварии компьютера, как существующие, так и новые соединения ″зависнут″. Зависание соединений происходит потому, что в сети нет компьютеров с IP-адресом, который клиенты использовали для базы данных. Если работа базы данных прекращена корректно, `db2stop force` разрывает существующие соединения с базой данных и возвращается сообщение об ошибке.

Во время восстановления логический IP-адрес, связанный с базой данных, отключен - либо программными средствами SC2.2, либо из-за аварии компьютера, где находился логический хост. В этот момент любые новые соединения с базой данных на короткое время ″зависнут″.

Логический IP-адрес, связанный с базой данных, в конечном счете переносится на другой компьютер до запуска DB2. На этом этапе соединение с базой данных не ″зависнет″, но, так как DB2 в системе еще не запущена, будет получено сообщение об ошибке связи. Клиенты DB2, которые оставались подключенными к базе данных, также начнут получать сообщения об ошибках связи. Хотя клиенты по-прежнему считают себя подключенными, компьютер, на котором теперь находится логический IP-адрес, не имеет никаких сведений о существующих соединениях. Соединения просто сброшены, и клиент DB2 получает сообщение об ошибке связи. Через короткое время на этом компьютере будет запущена DB2, и можно будет успешно соединиться с базой данных. В этой точке база данных может находиться в неустойчивом состоянии, и клиентам может понадобиться подождать, пока она не будет восстановлена.

При проектировании прикладных программ для среды высокой доступности не требуется писать специальный код для этапов, на которых соединения с базой данных ″зависают″. Соединения ″зависают″ только на короткое время, пока программные средства Sun Cluster перемещают логический IP-адрес. Любая служба данных при работе на Sun Cluster столкнется на этом этапе с таким же ″зависанием″ соединений. В любом варианте потери активности базы данных клиенты получат сообщение об ошибке и должны будут повторять попытки соединиться с ней до установления соединения. С точки зрения клиента это выглядит так, как будто экземпляр высокой доступности прекратил работу, но затем был возвращен на тот же самый компьютер. При управляемом восстановлении после сбоя клиенту кажется, что он был принудительно отключен и позже смог восстановить соединение с базой данных на том же компьютере. При неуправляемом восстановлении клиенту кажется, что сервер баз данных отказал, но вскоре вновь был запущен на том же компьютере.

Структура диска для экземпляров EE и EEE

DB2 ожидает, что требуемые дисковые устройства или файловые системы будут одинаковыми на каждом компьютере кластера. Для этого требуемые диски или файловые системы должны быть сконфигурированы так же, как на логическом хосте, связанном с экземпляром высокой доступности, и иметь одинаковые имена каталогов на каждом компьютере кластера.

В среде высокой доступности поддерживаются как табличные пространства DMS, так и SMS. Контейнеры устройств для табличных пространств DMS должны использовать непосредственные устройства, созданные менеджером томов, которые либо имеют зеркальные копии, либо используют конфигурацию RAID. Обычные дисковые устройства, например, `/dev/rdsk/c20t0d0s0`, использовать нельзя, поскольку:

- Это увеличивает вероятность записи на диск одновременно с нескольких компьютеров.
- На другом компьютере может быть другое число контроллеров.

Если восстановление DB2 после сбоя происходит в этой ситуации, требуемые дисковые устройства не будут восприниматься так же, как на другом компьютере, и запуска DB2 не произойдет. Контейнеры файлов для табличных пространств DMS и контейнеры для табличных пространств SMS должны находиться в смонтированных файловых системах. Файловые системы для логического хоста монтируются автоматически, если они включены в файл `vfstab` для логического хоста.

Путь к файлу `vfstab` для логического хоста:

```
/etc/opt/SUNWcluster/conf/hanfs/vfstab.<логический_хост>
```

где *логический_хост* - имя логического хоста, связанного с файлом `vfstab`.

У каждого логического хоста есть свой файл `vfstab`, содержащий файловые системы, которые надо монтировать после переноса групп дисков для логического хоста на текущий компьютер, но до запуска служб высокой доступности. Программные средства Sun Cluster пытаются смонтировать любую правильно определенную файловую систему после запуска команды **fsck** (проверка файловой системы), которая позволяет убедиться, что файловая система в порядке. Если команда **fsck** выполняется неудачно, файловая система не будет смонтирована и в журнал заносится сообщение об ошибке.

Примечание: Если в некотором процессе есть открытый файл или текущий рабочий каталог под точкой монтирования, монтирование завершится неудачно. Чтобы избежать этого, убедитесь, что под точками монтирования, содержащимися в файле логического хоста `vfstab`, не осталось никаких процессов.

Если используются табличные пространства SMS, для структуры файловой системы экземпляра EEE можно использовать любые соглашения. Ниже приводится соглашение для утилиты hadb2_setup:

```
scadmin@crackle(190)# pwd
/export/ha_home/db2eee/db2eee
scadmin@crackle(191)# ls -l
всего 18
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0000 ->
                                        /log0/disks/db2eee/NODE0000
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0001 ->
                                        /log0/disks/db2eee/NODE0001
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0002 ->
                                        /log0/disks/db2eee/NODE0002
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0003 ->
                                        /log0/disks/db2eee/NODE0003
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0004 ->
                                        /log0/disks/db2eee/NODE0004
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0005 ->
                                        /log1/disks/db2eee/NODE0005
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0006 ->
```

```
                                        /log1/disks/db2eee/NODE0006
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0007 ->
                                        /log1/disks/db2eee/NODE0007
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0008 ->
                                        /log1/disks/db2eee/NODE0008
scadmin@crackle(192)#
```

Владелец экземпляра - db2eee, а каталог базы данных по умолчанию для экземпляра db2eee - /export/ha_home/db2eee. На логическом хосте log0 находятся разделы базы данных 0, 1, 2 и 3, а на логическом хосте log1 - разделы 4, 5, 6, 7 и 8.

Для каждого раздела базы данных есть соответствующий каталог NODE*xxxx*. Узловые каталоги для разделов базы данных указывают на каталог в связанной с ними файловой системе логического хоста.

При выборе соглашения о путях убедитесь, что:

1. Диски для файловой системы входят в группу дисков логического хоста, ответственного за разделы базы данных, которым требуются эти диски.
2. Файловые системы, содержащие контейнеры, монтируются при помощи файла vfstab логического хоста.

Структура домашнего каталога для экземпляров EE и EEE

Для экземпляра EE домашним каталогом должна быть файловая система, определенная в файле vfstab для логического хоста. Этот каталог доступен до запуска DB2 и перемещается вместе с DB2 в любое место кластера вслед за перемещениями логического хоста. На каждом из компьютеров есть собственная копия файла vfstab, и необходимо, чтобы ее содержимое на разных компьютерах совпадало. Ниже приведен пример домашнего каталога для экземпляра EE.

```
/log0/home/db2ee
```

где /log0 - файловая система для логического хоста log0, а db2ee - имя экземпляра DB2. Этот путь домашнего каталога должен быть помещен в файл /etc/passwd на каждом компьютере в кластере, где может находиться экземпляр ″db2ee″.

Для экземпляра EEE есть два способа установки домашнего каталога. При конфигурации горячего резервирования домашний каталог можно устанавливать так же, как и для экземпляра EE. При конфигурации взаимной подмены для домашнего каталога необходимо использовать HA-NFS, правильно сконфигурировав ее *до* настройки экземпляра EEE.

Один из компьютеров в кластере должен экспортировать файловую систему для экземпляра EEE с помощью файла dfstab для выбранного логического хоста. Файл dfstab содержит файловые системы, которые следует экспортировать

посредством NFS, когда на компьютере находится логический хост. На каждом из компьютеров есть собственная копия файла `dfstab`, и необходимо, чтобы ее содержимое на разных компьютерах совпадало.

Информация для файловой системы HA-NFS помещается в файл `hadb2tab` (с помощью программы `hadb2_setup`). Когда агент высокой доступности читает информацию об экземпляре, он автоматически монтирует файловую систему HA-NFS для этого экземпляра (смотрите раздел “Файл hadb2tab” на стр. 326).

Точкой монтирования для файловой системы HA-NFS обычно является `/export/ha_home`. На каждом компьютере кластера это будет NFS, смонтированная с логического хоста, который экспортирует каталог HA-NFS. Домашний каталог владельца экземпляра EEE находится в этом каталоге и называется:

```
/export/ha_home/<экземпляр>
```

где *экземпляр* - имя владельца экземпляра.

Домашний каталог для экземпляра можно завести на всех компьютерах, чтобы избежать необходимости монтировать и демонтировать его. Это требует дополнительных затрат на управление, обеспечивающих идентичность домашних каталогов на каждом компьютере. Если это не сделано, DB2 может не запуститься или запуститься с другой конфигурацией. Такая возможность *не* поддерживается.

Логические хосты и DB2 UDB EEE

Логический хост обычно служит для размещения одного или более разделов базы данных, а также для экспорта файловой системы HA-NFS. Например, если в базе данных четыре раздела, а в кластере два компьютера, на каждом из них будет по логическому хосту (рис. 61 на стр. 324). Один логический хост можно использовать для размещения двух разделов базы данных и экспорта файловой системы HA-NFS, а другой - для размещения остальных двух разделов базы данных.

По умолчанию экземпляр DB2 UDB EEE выделяет достаточно ресурсов для успешного добавления до двух разделов базы данных на компьютер, где уже есть один или несколько активных разделов базы данных для этого экземпляра. Например, если на один экземпляр в кластере приходится четыре раздела базы данных, на логическом хосте может быть один раздел базы данных или же сразу три раздела. В любом случае можно восстановить три раздела базы данных на компьютере, где уже есть какой-либо раздел базы данных для того же самого экземпляра.

С помощью переменной реестра DB2_NUM_FAILOVER_NODES можно увеличить количество ресурсов, оставляемых под восстанавливаемые разделы базы данных.

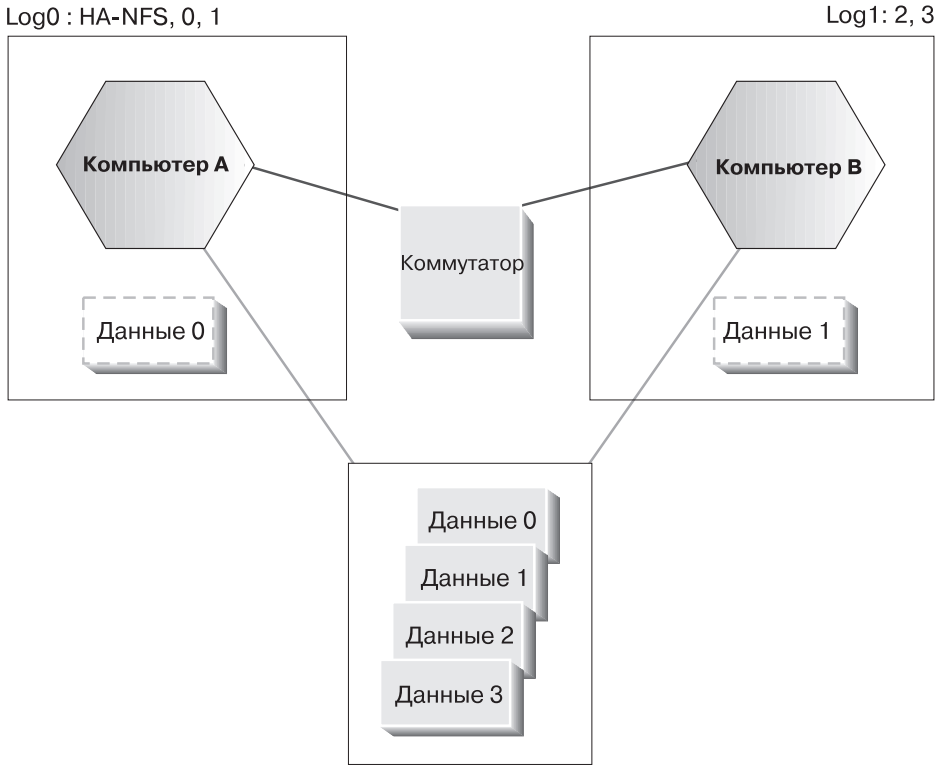


Рисунок 61. По одному логическому хосту на каждом компьютере

Положение и опции установки DB2

Файловая система, на которой установлена DB2, должна иметь зеркальную копию или же находиться в конфигурации RAID. Если DB2 установлена на обычных дисках, вероятность ошибки диска выше; восстановление в итоге может быть не выполнено и устойчивость кластера будет ниже.

DB2 нельзя установить на диски в группе дисков для логического хоста, так как агенту высокой доступности постоянно требуется доступ к библиотекам DB2. Если агенты высокой доступности не имеют доступа к библиотекам DB2, они не смогут продолжать работу. DB2 должна быть правильно установлена на каждом компьютере кластера.

Параметры конфигурации базы данных и менеджера баз данных

Параметры конфигурации менеджера баз данных можно изменить после отказа и восстановления и до запуска DB2 с помощью сценария `pre_db2start`

(смотрите раздел “Пользовательские сценарии” на стр. 328). Этот выполняемый сценарий (если он существует) запускается в подкаталоге `sqllib/ha` домашнего каталога владельца экземпляра. Как следует из его имени, он запускается непосредственно перед командой **db2start**. Если речь не идет об экземпляре EEE, сценарию `pre_db2start` передаются те же аргументы, что и методам управления. Для экземпляра EEE сценарию `pre_db2start` передается также номер узла для команды **db2start**.

Восстановление после аварии

Восстановление после аварии в среде высокой доступности происходит так же, как в обычной среде. Даже если экземпляр высокой доступности перенесен с одного из компьютеров, на котором произошла авария, на другой, файлы и дисковые устройства для экземпляра будут выглядеть так же и действия, необходимые для восстановления базы данных, останутся неизменными. Дополнительную информацию о восстановлении после аварии и других формах восстановления базы данных смотрите в разделе ″Восстановление базы данных″ книги *Руководство администратора: Реализация*.

Хотя базу данных можно перезапустить вручную (или с помощью одного из пользовательских сценариев), рекомендуется задать значение параметра конфигурации базы данных *autorestart* `ON`, особенно для экземпляра EEE. Это сократит до минимума время нахождения базы данных в несогласованном состоянии.

Высокая доступность через репликацию данных

Доступность данных можно увеличить также через их репликацию. Одна из форм высокой доступности достигается через репликацию данных между двумя серверами. При отказе одного из серверов другой должен быть способен взять на себя его функции по обеспечению службы данных.

Однако из-за асинхронности репликации ко времени отказа сервера некоторые изменения могут еще не быть реплицированы на другой сервер.

Агент высокой доступности DB2

Агент высокой доступности DB2 действует как посредник между DB2 и SC2.x. Он обеспечивает программным средствам Sun Cluster 2.2 возможность управлять DB2 в кластеризованной среде, не зная подробностей работы DB2. Для экземпляров EE и EEE используется один и тот же агент. Этот агент поддерживает как экземпляры управления, так и экземпляры баз данных.

Регистрация службы hadb2

Для работы с SC2.2 агент высокой доступности DB2 необходимо зарегистрировать. При регистрации служба данных сообщает SC2.2, какие методы управления доступны, и в каком каталоге они находятся. Специальный сценарий `hadb2_reg`, поставляемый вместе с агентом высокой доступности,

может зарегистрировать службу hadb2 для экземпляров как EE, так и EEE. Сценарий hadb2_reg требуется запускать только один раз для всего кластера.

Хотя для агента высокой доступности DB2 существует только один набор методов управления, способ их регистрации зависит от того, будет ли использован экземпляр EEE в конфигурации взаимной подмены. Для экземпляра EE или экземпляра EEE в конфигурации горячего резервирования HA-NFS не используется; поэтому переключатель "-d nfs", сообщающий программным средствам SC2.2, что служба hadb2 зависит от HA-NFS, не требуется.

Сама команда, которую hadb2_reg использует для регистрации методов управления DB2 Версии 7.1 для экземпляра EEE, выглядит так:

```
hareg -r hadb2 -b /opt/IBMdb2/V7.1/ha -m
START=hadb2_start,START_NET=hadb2_startnet,STOP_NET=hadb2_stopnet,
FM_START=hadb2_fmstart,FM_STOP=hadb2_fmstop
-t START_NET=$TIMEOUT,STOP_NET=$TIMEOUT -d nfs
```

Переключатель -b заставляет SC2.x обращаться за всеми методами управления к каталогу opt/IBMdb2/V7.1/ha. Переключатель -m определяет фактические методы управления для службы hadb2. Переключатель -t задает срок ожидания для методов управления START_NET и STOP_NET. Подробное описание каждого из методов управления смотрите в документации по Sun Cluster.

Сценарий hadb2_unreg можно использовать для отмены регистрации службы hadb2; подобно сценарию hadb2_reg, он должен запускаться для кластера только один раз.

Файл hadb2tab

Файл hadb2tab - главный файл конфигурации для агента высокой доступности DB2. Каждый метод управления обращается к этому файлу, чтобы узнать, какой из экземпляров является высокодоступным. Файл hadb2tab находится в каталоге /var/db2/v71/ для DB2 UDB Версии 7.1. Этот файл поддерживает несколько экземпляров, причем каждая строка (кроме комментариев) представляет отдельный экземпляр высокой доступности. Ниже приводится пример файла hadb2tab:

```
<scadmin@thrash(203)# cat hadb2tab
EEE DATA db2eee jolt ON /export/ha_home /log0/home #Добавлено программным
                                                   #обеспечением HA DB2
EE ADMIN db2ee log1 ON  -               -          #Добавлено программным
                                                   #обеспечением HA DB2
```

Первое поле указывает агенту высокой доступности DB2, является ли данный экземпляр экземпляром EE или EEE. Второе поле указывает, является ли данный экземпляр экземпляром данных или экземпляром управления. Третье поле содержит имя пользователя экземпляра высокой доступности. Четвертое поле указывает на логический хост или хост HA-NFS для экземпляра, в зависимости

от того, идет ли речь об экземпляре EE или EEE. Пятое поле указывает, включен или выключен мониторинг отказов для экземпляра. Два последних поля соответствуют локальной точке монтирования и удаленному каталогу HA-NFS. Для этих полей, если они не используются, следует задать значение `-`; их следует использовать только в конфигурации взаимной подмены EEE. Комментарии в файле `hadb2tab` разрешены, если информация в строке перед маркером ″#″ либо имеет нулевую длину, либо является допустимым определением экземпляра.

Методы управления

Методы управления для агентов SC2.2 могут представлять собой наборы сценариев или программ. Агент для DB2 в Solaris - это набор программ, включающий следующие методы:

START_NET
: hadb2_startnet, используется для запуска DB2

STOP_NET
: hadb2_stopnet, используется для остановки DB2

FM_START
: hadb2_fmstart, используется для запуска монитора отказов для DB2

FM_STOP
: hadb2_fmstop, используется для остановки монитора отказов для DB2

Дополнительную информацию об этих методах управления смотрите в документации по Sun Cluster.

Для экземпляров EE логический хост, связанный с экземпляром, определяется именно в файле `hadb2tab`. Для экземпляров EEE, однако, метод управления должен также обратиться к:

```
~<экземпляр>/sqllib/ha/hadb2-eee.cfg
```

где `~<экземпляр>` - домашний каталог владельца экземпляра. Этот файл содержит по одной строке для каждого раздела базы данных и служит для связывания разделов базы данных с логическими хостами. Пример допустимого файла `hadb2-eee.cfg`:

```
crackle % cat hadb2-eee.cfg
NODE:log0 0
NODE:log0 1
NODE:log1 2
NODE:log1 3
```

Экземпляр или разделы базы данных следуют за соответствующим логическим хостом по всему кластеру. Логический хост может перемещаться на любой компьютер в кластере, поддерживаемый базовым оборудованием и SC2.2. При верно заданной конфигурации DB2 будет поддерживать любую топологию, поддерживаемую программными средствами SC2.2.

После прочтения всей информации для экземпляра метод управления узнает, какие логические хосты связаны с этим экземпляром. После синтаксического анализа аргументов командной строки метод управления также узнает, какие логические хосты размещены на данном компьютере, а какие - нет.

В приводимой ниже таблице показаны действия, предпринимаемые в зависимости от того, какой запущен метод управления, и размещены ли на данном компьютере логические хосты, связанные с разделом базы данных или экземпляром.

Метод управления	Связанные логические хосты размещены	Связанные логические хосты не размещены
START_NET	Запустить экземпляр DB2 или разделы базы данных	Действие отсутствует
STOP_NET	Действие отсутствует	Остановить экземпляр DB2 или разделы базы данных
FM_START	Запустить монитор отказов для экземпляра	Действие отсутствует
FM_STOP	Действие отсутствует	Остановить мониторинг отказов для экземпляра

Методы управления, выполняющие действия по запуску, применяются только к размещенным в данное время логическим хостам, а методы управления, которые выполняют действия по остановке, применяются только к неразмещенным в данное время логическим хостам.

Методы управления требуют также, чтобы каталог HA-NFS был смонтирован особым образом, если используется HA-NFS. Если локальная точка монтирования и каталог для HA-NFS не определены как - (тире), метод управления запускает `statvfs(2)` в локальной точке монтирования. Если файловая система для локальной точки монтирования - не `nfs`, агент делает попытку смонтировать файловую систему с помощью информации из строки `hadb2tab`. Если точка монтирования и каталог для HA-NFS определены как - (тире), требуется, чтобы файл `vfstab` соответствующего логического хоста монтировал файловую систему, содержащую домашний каталог экземпляра. Локальная точка монтирования и удаленный каталог для HA-NFS при конфигурациях горячего резервирования EE и EEE должны определяться только как - (тире).

Пользовательские сценарии

Эти сценарии запускаются из методов управления, чтобы добавить дополнительные функциональные возможности; им передаются те же аргументы командной строки, что и методам управления; пишет такие сценарии системный администратор или администратор базы данных.

Если программа должна запускаться из сценария, который запущен не в фоновом режиме, обдумайте запуск программы в фоновом режиме с nohup(1). Программа nohup защищает выполняемую программу от сигнала SIGHUP (или зависания). Без программы nohup программа, запускаемая в фоновом режиме из сценария, может прекратить работу, получив сигнал SIGHUP по завершении сценария.

Методы управления запускают следующие сценарии:

- /var/db2/v61/failover
- ~<экземпляр>/sqllib/ha/pre_db2start
- ~<экземпляр>/sqllib/ha/post_db2start
- ~<экземпляр>%s/sqllib/ha/post_failover
- ~<экземпляр>/sqllib/ha/pre_db2stop
- ~<экземпляр>/sqllib/ha/fm_warning

где ~*экземпляр* - домашний каталог экземпляра высокой доступности.

За исключением сценария fm_warning, каждый пользовательский сценарий запускается с теми же аргументами, что и вызвавший его метод управления. Если используются экземпляры EEE, пользовательскому сценарию передается также номер раздела базы данных (в качестве последнего аргумента).

Сценарий /var/db2/v71/failover вызывается в начале выполнения метода START_NET и работает в фоновом режиме. Такой сценарий можно использовать, например, для оповещения по электронной почте персонала поддержки в случае восстановления после отказа. Ниже приводится пример сценария восстановления после отказа:

```
#!/bin/ksh

# Уведомить персонал поддержки о произошедшем восстановлении после отказа
# по электронной почте или по пейджеру.

echo "Восстановление после отказа на компьютере 'hostname':Running $0!"
|/bin/mail admin@sphere.torolab.ibm.com
```

Чтобы успешно направлять сообщения по электронной почте из сценария, программа sendmail(1m) должна быть правильно сконфигурирована в данной системе.

Как можно заключить по его имени, сценарий pre_db2start запускается непосредственно перед вызовом команды **db2start**. С помощью этого сценария можно решать такие задачи, как изменение параметров конфигурации менеджера баз данных. На его завершение отводится не более 20 секунд. Для экземпляров EEE этот сценарий запускается до вызова команды **db2start** для

каждого раздела базы данных. Этот сценарий применим только к экземплярам данных, но не к экземплярам управления.

Сходным образом, сценарий `post_db2start` запускается сразу *после* вызова команды **db2start**. С помощью этого сценария можно решать такие задачи, как перезапуск баз данных. Он запускается в фоновом режиме, чтобы во время выполнения не мешать работе других экземпляров. Этот сценарий применим только к экземплярам данных, но не к экземплярам управления.

Сценарий `post_failover` в домашнем каталоге владельца экземпляра запускается после обработки этого экземпляра. С помощью этого сценария можно уведомлять клиентские прикладные программы о восстановлении работоспособности DB2, активировать базы данных или посылать администраторам файл состояния. Он запускается в фоновом режиме, чтобы во время выполнения не задерживать действия других экземпляров высокой доступности. Ниже приводится пример сценария, выполняемого после отказа и восстановления:

```
#!/bin/ksh
#

# Послать файл состояния администратору.
mail admin@sphere.torolab.ibm.com </tmp/HA.info.db2eee
```

Оба метода (START_NET и STOP_NET) агента высокой доступности DB2 создают файл состояния после обработки каждого экземпляра. Имя файла состояния:

```
/tmp/HA.info.<экземпляр>
```

где *экземпляр* - имя пользователя владельца экземпляра. Файл состояния содержит отчет о запуске и остановке экземпляра, а также время, затраченное на запуск метода управления. Ниже приводится пример файла состояния:

```
scadmin@crackle(173)# cat /tmp/HA.info.db2eee
-----  Elapsed Time: 00:00:18           -----
-----  Elapsed Time: 00:00:00  (HA-NFS) -----

NODE     ACTION     RESULT     TRIES     RC
----     ------     ------     -----     --
   4     stop       success        3    1064
   5     stop       success        1    1064
   6     stop       success        2    1064
   7     stop       success        2    1064
   8     stop       success        1    1064
-----------------------------------------------
```

Сценарий `pre_db2stop` запускается непосредственно перед вызовом команды **db2stop**. С помощью этого сценария можно уведомлять клиентские прикладные

программы о готовности DB2 к остановке. На его завершение отводится не более 20 секунд. Этот сценарий применим только к экземплярам данных, но не к экземплярам управления.

Если DB2 перезапускается из-за неожиданного прекращения работы, на мониторе отказов будет также запущен пользовательский сценарий. Этот сценарий называется:

```
~<экземпляр>/sqllib/ha/fm_warning
```

С помощью сценария `fm_warning` можно уведомлять системного администратора о перезапуске DB2 монитором отказов. Системный администратор должен выяснить причину неожиданного прекращения работы DB2 и предпринять соответствующие действия для предотвращения этого события впредь. Сценарий `fm_warning` запускается в фоновом режиме.

Другие особенности

При выключенной службе данных высокой доступности во время восстановления после отказа или повторного конфигурирования кластера работают только методы остановки; другие методы запускаются только, если служба данных высокой доступности правильно зарегистрирована и включена.

Убедитесь, что каждый компьютер в кластере располагает достаточными ресурсами для работы всех служб данных, за которые он может нести ответственность. Такие ресурсы, как нагрузка процессора, память, параметры подкачки и ядра, надо рассматривать до того, как кластер начнет работать в производственном режиме. Например, если на каком-либо компьютере в кластере может понадобиться выполнять два экземпляра DB2, для параметра ядра этого компьютера надо сложить величины, необходимые для каждого экземпляра.

Монитор отказов

При включенном мониторинге отказов монитор запускается во время переконфигурирования кластера или восстановления после отказа. Если DB2 не запущена сценарием START_NET, монитор отказов запустит ее сам. Монитор отказов способен обнаруживать, что DB2 не запущена или прекратила работу по неизвестной причине. Поэтому важно не закрывать DB2 вручную при включенном мониторе отказов. Монитор отказов воспримет это как неожиданное прекращение работы и перезапустит DB2. Если это будет происходить слишком часто, для соответствующего логического хоста будет выполнено восстановление после отказа.

Если для некоторого экземпляра включен мониторинг отказов, правильный способ запуска или остановки этого экземпляра вручную состоит в предварительном отключении мониторинга отказов или службы `hadb2`. Оба эти действия можно инициировать при помощи команды **hadb2_setup**, используя переключатели `-f` и `-s` (смотрите раздел “Команда hadb2_setup” на стр. 337).

Примечание: Не используйте более одного экземпляра для одного и того же логического хоста. Если с логическим хостом связано несколько экземпляров, нормально работающий экземпляр может быть подвергнут процедуре восстановления после отказа вместе с дефектным.

Особенности EEE

Принимая решение, какие разделы базы данных связать с логическим хостом, важно учитывать, как они восстанавливаются после отказов. Рассмотрим кластер из двух компьютеров, на которых размещены четыре раздела базы данных, как показано на рис. 62.

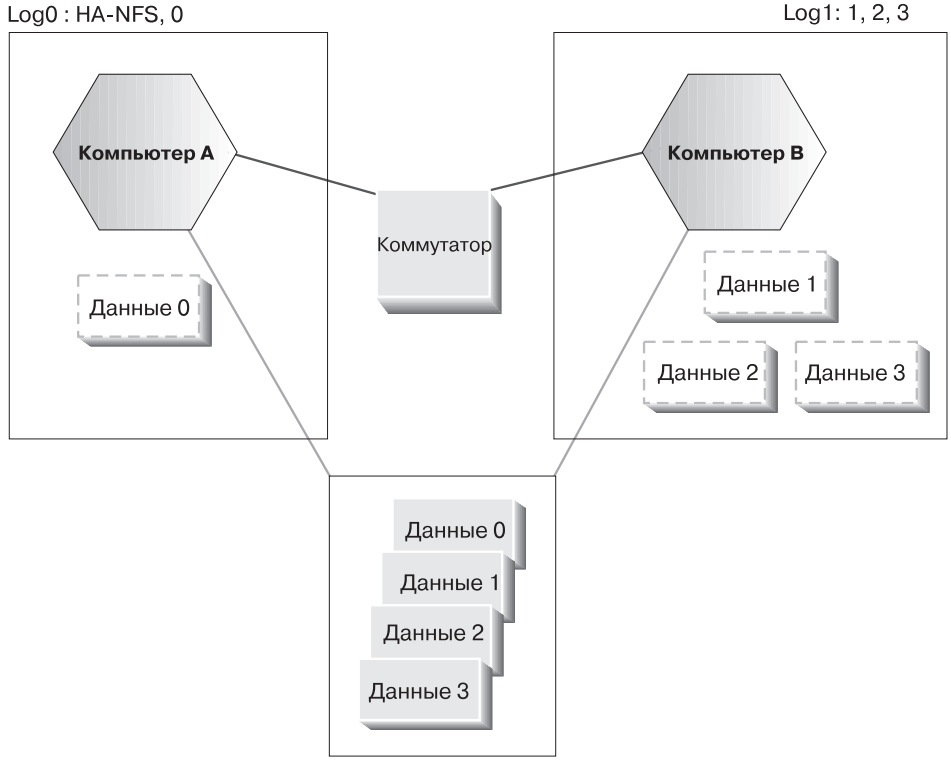


Рисунок 62. Кластер из двух компьютеров с четырьмя разделами базы данных

Один из логических хостов можно связать с каждым из разделов базы данных, а другой - с HA-NFS. В этом случае, если все логические хосты размещаются на одной системе, может возникнуть ошибка. При сбое в этой системе все логические хосты надо будет одновременно перенести с нее в другое место. К сожалению, программное обеспечение Sun Cluster не перемещает логические хосты в каком-либо фиксированном порядке, и возможна ситуация, когда логический хост, с которым связан раздел базы данных, будет перемещен

раньше, чем логический хост с HA-NFS. Обычно оказывается полезным группировать разделы базы данных в соответствии с тем, могут ли они быть размещены в единой системе. Это означает, что два раздела базы данных, обычно размещенные на одном компьютере, должны быть связаны с одним логическим хостом.

Файл db2nodes.cfg, используемый экземпляром EEE, изменяется; в нем указывается компьютер, на котором находятся разделы базы данных. Например, если все разделы базы данных находятся на компьютере "crackle", файл db2nodes.cfg будет выглядеть примерно так:

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 crackle 2 204.152.65.33
3 crackle 3 204.152.65.33
4 crackle 4 204.152.65.33
5 crackle 5 204.152.65.33
6 crackle 6 204.152.65.33
7 crackle 7 204.152.65.33
8 crackle 8 204.152.65.33
```

Если некоторые из этих разделов базы данных перемещаются на компьютер "thrash", файл db2nodes.cfg изменяется следующим образом:

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 crackle 2 204.152.65.33
3 crackle 3 204.152.65.33
4 thrash 0 204.152.65.34
5 thrash 1 204.152.65.34
6 thrash 2 204.152.65.34
7 thrash 3 204.152.65.34
8 thrash 4 204.152.65.34
```

Обратите внимание на то, что как имя хоста, так и название переключателя изменяются в соответствии с новым именем компьютера "thrash", и что номера портов также отличаются от прежних.

Файл HA.config

Если файл /etc/HA.config существует, он может содержать ряд опций конфигурации, в том числе:

```
scadmin@thrash(204)# cat /etc/HA.config
SYSLOG_FACILITY=LOG_LOCAL3
SYSLOG_LPRIORITY=LOG_INFO
SYSLOG_EPRIORITY=LOG_ERR
USE_INTERCONNECT=auto
SWITCH_NAME=204.152.65.18
DEBUG_LEVEL=2
FAILS_PER_HOUR=2
FAILS_PER_DAY=4
```

```
FAILS_PER_WEEK=10
FM_FAIL_SEV=soft
DB2START_TIMEOUT=60
DB2STOP_TIMEOUT=500
SCRIPT_USER=bin
```

Примечание: Если файл `HA.config` не существует, используются значения по умолчанию.

Переменная SYSLOG_FACILITY задает, что утилита SYSLOG должна регистрировать в журнале как сообщения, так и ошибки. Переменные SYSLOG_LPRIORITY и SYSLOG_EPRIORITY задают для SYSLOG приоритет регистрации, соответственно, информационных сообщений и сообщений об ошибках.

Чтобы разрешить демону SYSLOG регистрировать информацию от агента высокой доступности DB2, может потребоваться внести некоторые изменения. Например, одна из следующих двух строчек, добавляемых к файлу `/etc/syslog.conf`, указывает демону SYSLOG о необходимости записывать информацию в файл журнала.

```
*.notice                                        /var/adm/SC.x
local3.info                                     /var/adm/SC.LOG_LOCAL3
```

В Sun Cluster обычно имеется высокоскоростное межсоединение. Чтобы использовать высокоскоростное межсоединение вместе с DB2, задайте значение USE_INTERCONNECT `auto` или `override`. Значение `auto` (по умолчанию) использует внутренний логический сетевой интерфейс Sun. Этот интерфейс переносится на другой физический интерфейс в случае отказа исходного интерфейса. Если USE_INTERCONNECT имеет значение `override`, имя переключателя берется из переменной SWITCH_NAME. Другое возможное значение USE_INTERCONNECT - `no`, означающее, что высокоскоростное межсоединение не используется.

Переменная DEBUG_LEVEL задает количество информации, заносимой в системный журнал во время восстановления после отказов. Это целое число между 0 и 10; 10 соответствует высшему уровню отладки. Информация заносится в журнал в зависимости от заданного приоритета SYSLOG. При возникновении каких-либо неисправностей задайте максимальный уровень отладки, сконфигурируйте SYSLOG на запись вывода данных от агентов высокой доступности и отправьте вывод SYSLOG в центр обслуживания IBM.

Следующие три переменные помогают монитору неисправностей DB2 решить, когда следует восстанавливать логический хост: FAILS_PER_HOUR, FAILS_PER_DAY, и FAILS_PER_WEEK. У каждой среды высокой доступности есть свои особенности; вам надо решить, какое количество ошибок DB2 допустимо. После каждой ″допустимой″ ошибки DB2 перезапускается на том же

компьютере. При достижении одного из трех порогов ошибки для логического хоста, связанного с экземпляром или разделом базы данных, выполняется восстановление после отказа.

Переменная FM_FAIL_SEV задает ″мягкий″ или ″жесткий″ тип восстановления. Дополнительную информацию смотрите в описании `hactl(1m)` в документации по Sun Cluster.

Переменные DB2START_TIMEOUT и DB2STOP_TIMEOUT задают максимальный интервал времени в секундах, в течение которого разрешена работа команд **db2start** и **db2stop**. По истечении заданного срока агент высокой доступности считает, что операция завершилась неудачно.

Существуют пользовательские сценарии, которые не связаны ни с одним конкретным экземпляром. Обычно эти сценарии запускаются от имени пользователя root; это можно переопределить при помощи переменной SCRIPT_USER, в которой можно задавать ID пользователя, который будет запускать эти сценарии.

Как методы управления запускают команды DB2

Для запуска команд в качестве владельца экземпляра агент высокой доступности DB2 использует команду **su**. Эта команда будет выглядеть примерно так:

```
su - <экземпляр> -c "db2stop"
```

где *экземпляр* - имя пользователя экземпляра.

Важно, чтобы файл `.profile` владельца экземпляра был ″дружественным″ по отношению к **su**. Если это не так, команда **su** может работать неправильно. Вызовите команду **su** вручную или из сценария, чтобы проверить возможность ее успешной работы.

Установка

Прежде чем читать этот раздел, надо ознакомиться с программным обеспечением SC2.2. Данный раздел предполагает, что вы умеете устанавливать SC2.2 и HA-NFS, а также пользоваться вашим менеджером томов. Наряду с другими необходимыми исправлениями для DB2, для агента высокой доступности требуются следующие исправления:

```
Solaris 2.6:
   105210-17 (или более новое)
   105786-05 (или более новое)
```

Примечание: Для Solaris 7 (Solaris 2.7) исправления не требуются.

Общие шаги по установке

1. Установите SC2.2 на всех компьютерах в кластере. Во время установки SC2.2 запросит, какие агенты устанавливать. Поскольку DB2 не поставляется с SC2.2, ее нет в списке агентов. Агент для DB2 будет установлен вместе с ней и зарегистрирован при помощи команды **hadb2_reg**.
2. Сконфигурируйте логические хосты с группами дисков и логическими IP-адресами.

Установка на DB2 UDB Enterprise Edition

1. Создайте домашний каталог для экземпляра в файловой системе логического хоста.
2. Установите DB2 на всех компьютерах в кластере.
3. Создайте экземпляр на компьютере кластера, где в данный момент находится домашний каталог для экземпляра.
4. Добавьте пользователя для экземпляра на другие компьютеры кластера, проследив за тем, чтобы использовался один и тот же численный ID пользователя.
5. Зарегистрируйте службу `hadb2` при помощи команды **hadb2_reg**.
6. Запустите команду **hadb2_setup** для конфигурирования высокой доступности для экземпляра.

Установка на DB2 UDB Enterprise - Extended Edition

1. Создайте домашний каталог для владельца экземпляра высокой доступности.
   a. Для горячего резервирования создайте домашний каталог для экземпляра в файловой системе логического хоста.
   b. Для взаимной подмены сконфигурируйте HA-NFS и экспортируйте домашний каталог с одного из логических хостов. Смонтируйте каталог HA-NFS на одном из компьютеров под выбранной точкой монтирования.
2. Установите DB2 на всех компьютерах в кластере.
3. Создайте экземпляр на компьютере, где смонтирована файловая система HA-NFS.
4. Добавьте пользователя для экземпляра на другие компьютеры кластера, проследив за тем, чтобы использовался один и тот же численный ID пользователя.
5. Зарегистрируйте службу `hadb2` при помощи команды **hadb2_reg**.
6. Запустите команду **hadb2_setup** для конфигурирования высокой доступности для экземпляра.

Примечание: Использовать NIS для определения информации для экземпляра высокой доступности не рекомендуется, так как NIS может оказаться точкой возможной единственной ошибки, которая приведет к отказу системы.

Команда hadb2_setup

Команда **hadb2_setup** - центральная программа, поставляемая с агентом высокой доступности DB2. Ее можно использовать для установки, изменения и удаления экземпляра. Она также может включать и выключать службу hadb2_setup. Эта команда позволяет не редактировать файл hadb2tab вручную.

Примечание: Команда **hadb2_setup** выполняет действия только на том компьютере, на котором она запущена. Изменения, сделанные на одном из компьютеров кластера, должны быть также сделаны и на остальных компьютерах.

Поддерживаются следующие аргументы:

```
Для добавления экземпляра EE:
---------------------
   hadb2_setup -a -i <экземпляр> -f [on|off] -h <логический_хост>
   -p [DATA|ADMIN] -t EE

Например:
      hadb2_setup -a -i db2ee -f off -h log1 -p DATA -t EE



Для добавления экземпляра EEE:
----------------------
   hadb2_setup -a -i <экземпляр> -f [on|off] -h <nfs_host>
   -l <точка_монтирования> \
      -r <каталог_ha-nfs> -p [DATA|ADMIN] -t EEE -n "<информация_узла>"

Например:
      hadb2_setup -a -i db2eee -f off -h ha-sun1 -l /export/ha_home \
         -r /log0/home -p DATA -t EEE -n "log0[0,10,20],log1[30,40,50]"



Для удаления экземпляра:
---------------------
   hadb2_setup -d -i <экземпляр>



Для изменения экземпляра:
---------------------
   hadb2_setup -m -i <экземпляр> [-f [on|off] | -l <точка_монтирования> | \
      -h <хост> | -p [DATA|ADMIN] | -r <каталог_ha-nfs> | -t [EE|EEE] ]



Другие опции:
-------------
   -s   <on|off>          Включить или выключить hadb2 (для всех экземпляров
                                 высокой доступности)
   -y                     Предполагать "да" для проверок защиты
```

Чтобы включить или выключить службу hadb2, задайте переключатель -s. Это равносильно использованию команды **hareg** с переключателями -n и -y и заданию службы hadb2. Дополнительную информацию о команде **hareg(1m)** смотрите в документации по Sun Cluster.

Монитор отказов для экземпляра выключается с помощью переключателя -f. В результате этого монитор отказов для экземпляра на локальном компьютере останавливается, а в файл hadb2tab вносится изменение, отражающее это событие.

Для экземпляров EE выключение мониторинга отказов на всех компьютерах рекомендуется при восстановлении экземпляра после сбоя. Для экземпляров EEE мониторинг отказов на всех компьютерах с разделами базы данных для экземпляра должен быть выключен до его закрытия вручную.

Экземпляр удаляется при помощи переключателя -d. При этом экземпляр удаляется только из файла hadb2tab; никакие другие файлы или каталоги не удаляются и не изменяются. Поскольку файл hadb2tab - главный файл конфигурации для агента HA-DB2, удаление экземпляра из этого файла лишает метод управления информации о существовании экземпляра.

Экземпляр изменяется при помощи переключателя -m. При этом информация изменяется только в файле hadb2tab; никакие другие файлы или каталоги не удаляются и не изменяются. Переключатель -m можно использовать с любым переключателем, который имеет отношение к информации в файле hadb2tab. Файлы db2nodes.cfg и hadb2-eee.cfg после исходной установки надо изменять вручную, потому что команда **hadb2_setup** не поддерживает изменение этих файлов.

Добавление экземпляра представляет собой более сложную процедуру.

В случае экземпляров EE требуются следующие аргументы:

```
hadb2_setup -a -i <экземпляр> -f <fm> -h <логический_хост> -t <EEE_или_EE>
-p <назначение>
```

где *экземпляр* - имя добавляемого экземпляра, *fm* указывает, включен или выключен мониторинг отказов в исходном состоянии, *логический_хост* - связанный логический хост, *EEE_или_EE* задается равным EE, а *назначение* может быть как DATA, так и ADMIN.

Для экземпляров EEE требуются следующие аргументы:

```
hadb2_setup -a -i <экземпляр> -f <fm> -h <хост_nfs> -t <EEE_или_EE> -p
   <назначение> -l <точка_монтирования> -r <каталог_HA-NFS>
   -n <информация_узла>
```

где *экземпляр* - имя добавляемого экземпляра, *fm* указывает, включен или выключен мониторинг отказов в исходном состоянии, *хост_nfs* - имя логического хоста, экспортирующего файловую систему HA-NFS, *EEE_или_EE* задается равным EEE, *назначение* может быть как DATA, так и ADMIN, *точка_монтирования* - локальная точка монтирования для каталога HA-NFS, *каталог_HA-NFS* - каталог HA-NFS, а *информация_узла* - информация, связывающая разделы базы данных с логическим хостом. Например:

```
hadb2_setup -a -i db2eee -f on -h jolt -l /export/ha_home -p DATA -t EEE -r
   /log1/home -n "log0[0,1],log1[2,3]"
```

При добавлении экземпляра EEE информация узла должна быть заключена в кавычки. В этом примере экземпляр ″db2eee″ связан с двумя логическими хостами, ″log0″ и ″log1″. Разделы базы данных ″0″ и ″1″ экземпляра ″db2eee″ связаны с логическим хостом ″log0″, а разделы ″2″ и ″3″ - с логическим хостом ″log1″.

Чтобы добавить экземпляр на все компьютеры кластера, воспользуйтесь командой **hadb2_setup**. Экземпляр можно затем запустить принудительным повторным конфигурированием кластера или выключив, а затем вновь включив службу hadb2. Это можно сделать при помощи как команды **hareg**, так и переключателя -s команды **hadb2_setup**. Если экземпляр не запускается, смотрите раздел “Устранение неисправностей” на стр. 343.

Если команда **hadb2_setup** добавляет экземпляр EEE, следующие действия выполняются в прозрачном режиме:

- Проверка заданной информации. Проверяется, есть ли в системе пользователь и запущена ли HA-NFS.
- Создание файла db2nodes.cfg.
- Создание файла hadb2-eee.cfg.
- Создание файла .rhosts для экземпляра EEE.
- Создание символических ссылок от пути базы данных по умолчанию к связанным каталогам данных логических хостов.
- Добавление строки в файл hadb2tab.

Чтобы предотвратить ошибки в конфигурации и убедиться в способности экземпляра высокой доступности запускаться после запуска команды **hadb2_setup**, эта команда выполняет значительный объем проверок перед добавлением нового экземпляра.

Файл db2nodes.cfg создается и заполняется информацией в соответствии с текущим состоянием кластера. Например, если логический хост ″log0″ находится на компьютере ″crackle″, записи для разделов базы данных, связанных с ″log0″, будут содержать имя компьютера ″crackle″ и высокоскоростное межсоединение для ″crackle″:

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 thrash 0 204.152.65.34
3 thrash 1 204.152.65.34
```

Файл `hadb2-eee.cfg` создается только на основе информации узла, заданной в команде. На каждый раздел базы данных приходится одна строка:

```
sphere % cat hadb2-eee.cfg
NODE:log0 0
NODE:log0 1
NODE:log1 2
NODE:log1 3
```

Файл `.rhost` требуется для DB2 UDB EEE и должен содержать имена (или IP-адреса) всех хостов. Например:

```
crackle db2eee
204.152.65.1 db2eee
204.152.65.17 db2eee
thrash db2eee
204.152.65.2 db2eee
204.152.65.18 db2eee
crackle db2eee
jolt db2eee
bump db2eee
thrash.torolab.ibm.com db2eee
crackle.torolab.ibm.com db2eee
```

В соответствии со структурой файловой системы для табличных пространств SMS команда **hadb2_setup** задает несколько каталогов и символических связей. В их число входят:

- Каталог с именем ″data″ в файловой системе каждого логического хоста.
- Каталог узла (в этом каталоге data) для каждого раздела базы данных, связанного с логическим хостом.
- Символические связи на путь базы данных по умолчанию `~<экземпляр>`, где *~экземпляр* - домашний каталог экземпляра. Для каждого из разделов базы данных задается одна символическая связь, указывающая на соответствующий каталог узла. Дополнительную информацию смотрите в разделе “Структура диска для экземпляров EE и EEE” на стр. 320.

Срок восстановления после сбоя

Срок восстановления измеряется от момента, когда данные впервые становятся недоступными, до момента, когда они становятся снова доступными. Количество событий, происходящих во время восстановления после сбоя, может значительно повлиять на срок восстановления:

- Удаление и импортирование диска.

Удаление и импортирование диска обычно не отнимает слишком много времени по сравнению с другими событиями, хотя оно и увеличивает общие потери времени. Чем больше дисков требуется перенести с одного компьютера на другой во время восстановления после сбоя, тем больше времени отнимает процесс. При наличии дефектов на дисках процесс может идти еще дольше.

- Проверка файловых систем, монтируемых для логического хоста.

  Прежде чем файловые системы логического хоста можно будет смонтировать, им необходимо пройти проверку `fsck`, чтобы убедиться в нормальном состоянии файловой системы. Чем больше файловая система, тем больше времени отнимает процесс. Использование файловой системы с журналированием может резко сократить это время. Поскольку в среде высокой доступности как правило, используются файловые системы с журналированием, время, затрачиваемое на `fsck`, обычно не создает проблем.

- Пользовательские сценарии, вызываемые из агента высокой доступности.

  Агент высокой доступности вызывает пользовательские сценарии, если они существуют и выполняемы. Некоторые из этих сценариев запускаются синхронно и могут увеличить время запуска экземпляров высокой доступности. Надо сделать, чтобы они запускались как можно быстрее с учетом работы любых внешних программ, вызываемых этими сценариями в фоновом режиме.

- HA-NFS.

  В случае экземпляра EEE в конфигурации взаимной подмены для домашнего каталога владельца экземпляра надо использовать HA-NFS. HA-NFS увеличивает срок восстановления после сбоя за счет периода задержки для *lockd* (определяется в агенте высокой доступности для HA-NFS), который при работе HA-NFS составляет 90 секунд. Это влияет на сроки восстановления после отказа, потому что любой процесс, блокирующий какой-либо файл в файловой системе HA-NFS после восстановления, должен ждать окончания задержки. Агент составляет для DB2 - первый из процессов, блокирующий файл в домашнем каталоге владельца экземпляра после восстановления, и он записывает время, затраченное на получение первой блокировки. Это время выводится в отчете о состоянии после отказа и восстановления.

- Запуск DB2.

  Запуск DB2 занимает лишь малую часть времени восстановления после сбоя. Для экземпляра EE это в среднем около 5-15 секунд. Для экземпляра EEE это около 10 секунд плюс 5 на каждый восстанавливаемый раздел базы данных. Если, например, после сбоя восстанавливается три раздела базы данных, увеличение срока восстановления за счет запуска каждого из этих трех разделов составит примерно 25 секунд. Сюда *не* включено время на восстановление после аварии для баз данных экземпляра.

- Восстановление базы данных после аварии

Восстановление после аварии часто влияет на потери времени, связанные с восстановлением после отказа. Время, затрачиваемое на восстановление базы данных, зависит от ряда факторов, в том числе:

- Рабочая нагрузка клиента. В журналы транзакций заносятся только изменения в базе данных. Если рабочую нагрузку клиента составляют преимущественно операции ″только чтение″, к базе данных во время восстановления после аварии должно быть применено сравнительно немного транзакций.
- Скорость компьютера и диска. Скорость дисков и компьютера с экземпляром высокой доступности также влияет на время, требующееся на восстановление базы данных. Чем быстрее система, тем короче срок восстановления после аварии.
- Значение параметра конфигурации *softmax*. Значение *softmax* - это процент размера файла журнала, при котором должна быть выполнена мягкая контрольная точка и сделана запись в файл управления журналом. Файл управления журналом используется во время восстановления после аварии, чтобы определить, какие записи журнала действительно необходимы для восстановления базы данных до согласованного состояния. Снижение этого значения заставит менеджер баз данных чаще запускать чистильщики страниц и чаще выполнять мягкие контрольные точки; хотя производительность при этом падает, восстановление базы данных пойдет быстрее.
- Различие между экземплярами EE и EEE. Для экземпляра EEE операции перезапуска базы данных будут выполняться параллельно. Каждый раздел базы данных отвечает за перезапуск своей части баз данных. Если в базе данных 50 Гбайт данных, экземпляр EEE с четырьмя разделами способен восстановить базу данных примерно в четыре раза быстрее, чем экземпляр EE.

Устранение неисправностей

В следующей таблице перечислены возможные ошибки при работе, их вероятные причины и действия по их устранению.

Таблица 29. Устранение неисправностей высокой доступности в Sun Cluster 2.2

Симптом	Вероятная причина	Действие
Не удается смонтировать файловую систему логического хоста	Файловая система логического хоста обычно монтируется и демонтируется во время восстановления логического хоста после отказа. Во время восстановления в файловой системе логического хоста не должно быть активных процессов или открытых файлов. В отдельных случаях процессы, которые нельзя прервать, могут иметь в файловой системе логического хоста свой текущий рабочий каталог. Выяснить, находится ли процесс под точкой монтирования, можно при помощи `fuser(1m)` или утилиты GNU `lsof`. Если файловую систему логического хоста не удается смонтировать, выдаются сообщения об ошибках.[a]	Перезагрузите систему или присвойте другое имя файловой системе логического хоста и воссоздайте ее. В результате приостановленный процесс может остаться в каталоге (поскольку его нельзя прервать), и монтирование станет возможным.[b]
Не действует срок ожидания db2start или db2stop	Сигнал SIGALRM может не прервать блокирующий системный вызов. Вместо этого системный вызов будет перезапущен, как если бы флаг SA_RESTART был установлен с `sigaction()`. В результате этого сроки ожидания для агентов высокой доступности DB2 будут игнорироваться, а метод агента ″зависнет″ сам, вместо того, чтобы предохранить от ″зависания″ команду **db2start** или **db2stop**.	Примените необходимое исправление 105210-17 (или более новое) для Solaris 2.6.

Таблица 29. Устранение неисправностей высокой доступности в Sun Cluster 2.2 (продолжение)

Симптом	Вероятная причина	Действие
Регистрация на экземпляре ″зависает″	Это может произойти по разным причинам, но наиболее частые из них - ошибки NFS и программы `/usr/sbin/quota`.	Проверьте, нормально ли прошло монтирование NFS, и попробуйте найти процессы quota, принадлежащие владельцу экземпляра. С разрешения системного администратора ошибку можно устранить, поменяв программу quota на символическую ссылку на `/bin/true`. Это решение не рекомендуется, но его можно попытаться применить как обходной прием.
Только что установленный экземпляр EEE не запускается	Команда **hadb2_setup** не добавляет порты в файл `/etc/services`, ожидается, что администратор добавит их вручную. Возвращается сообщение об ошибке.[c]	Убедитесь, что у вас имеются подходящие порты, перечисленные в файле `/etc/services`.

Таблица 29. Устранение неисправностей высокой доступности в Sun Cluster 2.2 (продолжение)

Симптом	Вероятная причина	Действие
Метод START_NET не может запустить DB2		Выключите мониторинг неисправностей, чтобы убедиться, что экземпляр не начал восстанавливаться после отказа. Зарегистрируйтесь как владелец экземпляра и попробуйте запустить DB2 вручную. 1. Убедитесь, что в файле конфигурации `hadb2tab` указан правильный тип экземпляра. Например, при файле `db2nodes.cfg` для экземпляра управления EE будут возникать ошибки, а методы агента высокой доступности не смогут в этом случае выполнить восстановление. 2. Убедитесь, что файл `.rhosts` существует и содержит допустимые записи. 3. Убедитесь, что файловая система HA-NFS используется совместно с разрешениями root для всех компьютеров кластера. 4. Проверьте параметры ядра и убедитесь, что они заданы правильно. 5. Убедитесь, что файл `/etc/services` содержит записи для экземпляра.

Таблица 29. Устранение неисправностей высокой доступности в Sun Cluster 2.2 (продолжение)

Симптом	Вероятная причина	Действие
Экземпляр работает только на одном компьютере	• Числовой *ID пользователя* для экземпляра может не совпадать на разных компьютерах кластера. • Параметры ядра могут быть заданы верно не на всех компьютерах кластера. • Файл `hadb2tab` может быть разным на разных компьютерах. • Другие файлы конфигурации, например, файл `vfstab` логического хоста, могут не совпадать на разных компьютерах кластера.	Если ни одна из перечисленных причин не подходит, попробуйте зарегистрироваться как владелец экземпляра и запустить DB2 вручную. Для экземпляров EE это должно помочь, если логический хост с экземпляром находится на текущем компьютере. Для экземпляров EEE это должно быть сделано на каждом из компьютеров кластера, где могут быть разделы базы данных.
Не работает su - <экземпляр> -c ″db2start″	• Файл `.profile` для экземпляра может быть ″недружественным″ по отношению к **su**. • Это известная ошибка для оболочки Bourne (`/bin/sh`), где команда **su** запускается вручную, а не через агент высокой доступности.	• Попытайтесь запустить эту команду вручную как root и убедитесь, что она работает, прежде чем снова пробовать сделать это через агент высокой доступности. • Если это необходимо, переключитесь на оболочку Korn (`/bin/ksh`).
Экземпляр EEE не запускается, хотя домашний каталог смонтирован	Каталог HA-NFS, возможно, не был экспортирован на компьютеры кластера с разрешениями ″root″. Это требуется для нормальной работы как DB2, так и агентов высокой доступности.	Чтобы проверить это, попытайтесь создать файл (в качестве root) в домашнем каталоге владельца экземпляра.
При попытке обращения к каталогу экземпляра EEE возвращается сообщение об ошибке ″Stale NFS file handle″	В домашнем каталоге владельца экземпляра могут все еще оставаться процессы.	Демонтируйте домашний каталог владельца экземпляра и дайте возможность агенту высокой доступности смонтировать его заново. Агент высокой доступности смонтирует его, если службу `hadb2` выключить, а затем включить заново (смотрите описание переключателя `-s` команды **hadb2_setup** в разделе “Команда hadb2_setup” на стр. 337).

Таблица 29. Устранение неисправностей высокой доступности в Sun Cluster 2.2 (продолжение)

Симптом	Вероятная причина	Действие
Методы управления не запускаются успешно через SC2.2	Служба hadb2, возможно, не зарегистрирована с программным обеспечением Sun Cluster или не включена.	Если методы управления работают нормально из командной строки, проверьте, нет ли в файлах SYSLOG сообщений об ошибках, которые могли бы помочь понять суть данной ошибки. Убедитесь что служба hadb2 зарегистрирована с программным обеспечением Sun Cluster и что она включена. Запуск методов вручную полезен при устранении ошибок.[d] Методы надо запускать как root с соответствующими аргументами командной строки. Если список логических хостов пуст, аргумент должен задаваться как "". Двойные кавычки без пробела означают пустой аргумент. Например: `hadb2_startnet log0,log1 "" 600` Первый аргумент, log0,log1, сообщает методу hadb2_startnet, что логические хосты log0 и log1 находятся на текущем компьютере. Второй аргумент пуст, сообщая тем самым методу hadb2_startnet, что на других компьютерах кластера нет других логических хостов (все они находятся на текущем компьютере). Третий аргумент сообщает методу, что время ожидания для SC2.2 истечет через 600 секунд.

Таблица 29. Устранение неисправностей высокой доступности в Sun Cluster 2.2 (продолжение)

Симптом	Вероятная причина	Действие
Не запускаются пользовательские сценарии	Пользовательские сценарии можно запустить, только если они существуют в соответствующих каталогах и выполнимы.	Проверьте владельца файла и его атрибуты. Если сценарий по-прежнему не удается запустить, обратитесь в центр обслуживания IBM. Передайте список каталогов сценария, который не запускается, и запишите в SYSLOG вывод для восстановления после отказа или повторного конфигурирования кластера, после которых должен запуститься сценарий.
Информация не заносится в файл, указанный в /etc/syslog.conf		С помощью `touch(1)` создайте файл, указанный в файле `/etc/syslog.conf`, а затем перезапустите демон SYSLOG.

Таблица 29. Устранение неисправностей высокой доступности в Sun Cluster 2.2 (продолжение)

Симптом	Вероятная причина	Действие

[a] Сообщения об ошибках, которые выдаются, если файловую систему логического хоста не удается смонтировать, могут выглядеть примерно так:

```
Aug 17 11:14:01 rash ID[SUNWcluster.loghost.1170]: importing data1
Aug 17 11:14:06 rash ID[SUNWcluster.scnfs.3040]: mount -F ufs -o ""
     /dev/vx/dsk/data1/data1-stat /log1 failed.
Aug 17 11:14:07 rash ID[SUNWcluster.ccd.ccdd.5304]: error freeze cmd =
     /opt/SUNWcluster/bin/loghost_sync
CCDSYNC_POST_ADDU LOGHOST_CM:log1:rash /etc/opt/SUNWcluster/conf/ccd.database
     2 "0 1" 1 error code = 1
```

[b] Например:

```
scadmin@rash(218)# ps -fe | egrep db2
db2ee 1984 1 0 0:01 <defunct>

Решение:

   scadmin@rash(229)# cd /
   scadmin@rash(230)# mv /log1 /log1.bkp
   scadmin@rash(231)# mkdir /log1
```

[c] Сообщение об ошибке может выглядеть примерно так:

```
SQL6030N Команда START или STOP DATABASE MANAGER завершились неудачно.
Код причины "13".
```

[d] Например, если метод `hadb2_startnet` не может найти `libdb2.so.1`, но нормально запускается программными средствами Sun Cluster, сообщений об ошибках выдаваться не будет. Запуск метода вручную приводит к следующему результату:

```
scadmin@crackle(213)# hadb2_startnet '''log0,log1' 600
ld.so.1: hadb2_startnet: fatal: libdb2.so.1: open failed:
   No such file or directory
Killed
```

Часть 5. Приложения

Приложение A. 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. 368. 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 *Quick Beginnings* 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. 364. 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. 367
- “Поиск электронной информации” на стр. 372
- “Заказ печатных копий” на стр. 364
- “Печать книг PDF” на стр. 363

Таблица 30. Информация DB2

Имя	Описание	Номер формы Имя файла PDF	Каталог HTML
	Руководства и справочники по DB2		
Руководство администратора	*Руководство администратора: Планирование* содержит обзор понятий баз данных, информацию по вопросам разработки (в частности, по логическому и физическому проектированию баз данных) и обсуждение доступности баз данных.	SH43-0146 db2d1x70	db2d0
	Руководство администратора: Реализация содержит информацию о реализации ваших проектов, доступе к базам данных, аудите, резервном копировании и восстановлении.	SH43-0144 db2d2x70	
	Руководство администратора: Производительность содержит информацию о среде баз данных, оценке и настройке производительности программ.	SH43-0145 db2d3x70	
	Эти три тома книги *Руководство администратора* можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8934.		

Таблица 30. Информация DB2 (продолжение)

Имя	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 30. Информация DB2 (продолжение)

Имя	Описание	Номер формы Имя файла PDF	Каталог HTML
Дополнение по возможностям соединений	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.	SH43-0130 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
Glossary	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 30. Информация DB2 (продолжение)

Имя	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 настройке	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.	GP43-0126 db2iyx70	db2iy
Справочник по сообщениям	Содержит список сообщений и кодов, выдаваемых DB2, Information Catalog Manager, и Data Warehouse Center, и описывает для них рекомендуемые действия. Оба тома публикации Справочник по сообщениям можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8932.	Том 1 GH43-0128 db2m1x70 Том 2 GH43-0129 db2m2x70	db2m0
OLAP Integration Server Administration Guide	Объясняет, как использовать менеджер управления сервером OLAP Integration Server.	SC27-0787 db2dpx70	нет

Таблица 30. Информация DB2 (продолжение)

Имя	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 30. Информация DB2 (продолжение)

Имя	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.	SH43-0131 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 30. Информация DB2 (продолжение)

Имя	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. Быстрый старт	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.	GH43-0127 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 30. Информация DB2 (продолжение)

Имя	Описание	Номер формы Имя файла PDF	Каталог HTML
DB2 for OS/2 Quick Beginnings	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 Quick Beginnings	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 Quick Beginnings	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

Таблица 30. Информация DB2 (продолжение)

Имя	Описание	Номер формы Имя файла PDF	Каталог HTML
Кроссплатформенные программы примеров в формате HTML			
Программы примеров в виде HTML	Содержит 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 *Руководство администратора*, а имя файла `db2d0f70` соответствует французской версии этой же книги. Для обозначений языков на шестой позиции имени файла используются следующие буквы:

Язык	Обозначение
Бразильский португальский	b
Венгерский	h
Голландский	q
Греческий	a
Датский	d
Испанский	z
Итальянский	i

Корейский	k
Немецкий	g
Норвежский	n
Польский	p
Португальский	v
Русский	r
Словенский	l
Традиционный китайский	p
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. 30 на стр. 354.

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 *Quick Beginnings*.
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 31. 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 • Message Reference, Volumes 1 and 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 Центру 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 *Quick Beginnings*.
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	Создать, определить и заполнить план резервного копирования.	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**.
по созданию индексов	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 настройке*.
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 настройке*.

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. 368.

Приложение B. Правила именования

Задавая имена для перечисленных ниже баз данных и объектов баз данных, следуйте приведенным правилам именования:

- Имена баз данных
- Имена и алиасы баз данных
- ID пользователей и пароли
- Имена схем
- Имена групп и пользователей
- Имена объектов.

Не используйте зарезервированные слова IBM SQL и ISO/ANSI SQL92 для имен таблиц, производных таблиц, столбцов, индексов или ID авторизации. Список зарезервированных слов приведен в справочнике *SQL Reference*.

В руководствах *Quick Beginnings* описаны правила именования для ID авторизации (в том числе для имен пользователей и групп) и дополнительные ограничения для некоторых платформ.

Имена баз данных

Каждый раз при создании базы данных менеджер баз данных создает отдельный каталог для хранения управляющих файлов и файлов данных для этой базы.

Имена этих каталогов - от `SQL00001` до `SQL`*`nnnnn`*, где `SQL00001` содержит управляющие файлы для первой созданной базы данных, `SQL00002` - для второй и так далее.

Эти каталоги поддерживаются автоматически. Чтобы избежать потенциальных проблем с именами, не создавайте собственных каталогов с именами подобного вида и не меняйте каталоги, уже созданные менеджером баз данных.

Имена и алиасы баз данных

Имена баз данных - это имена, идентифицирующие базы; вы или ваши пользователи задаете их в команде CREATE DATABASE или в соответствующем API. Эти имена должны быть уникальными в том положении, где они занесены в каталог. Например, в реализациях DB2 для систем на основе UNIX положение - это путь каталога; в реализациях OS/2 это буква диска.

Алиас базы данных - это локальный синоним локальной или удаленной базы данных. Они должны быть уникальными в системном каталоге базы данных, где хранятся все алиасы отдельного экземпляра менеджера баз данных. При создании новой базы данных ее алиасом по умолчанию считается имя базы данных. Это означает, что нельзя создать базу данных, имя которой совпадало бы с существующим алиасом, даже если базы данных с этим именем не существует.

Задаваемые имена и алиасы баз данных:

- Могут содержать от 1 до 8 символов
- Должны начинаться с одного из следующих символов:
  - Буквы от A до Z (символы нижнего регистра преобразуются в верхний)
  - @, # или $
- Могут содержать символы:
  - Буквы от A до Z (символы нижнего регистра преобразуются в верхний)
  - Цифры от 0 до 9
  - @, #, $ и _ (подчеркивание)

Примечание: Во избежание возможных проблем не используйте символы @, # и $ в именах баз данных, к которым предполагается обращаться по связи. Кроме того, поскольку эти символы есть не на всех клавиатурах, не используйте их, если база данных должна использоваться в других странах.

ID пользователей и пароли

При задании ID пользователя или пароля задаваемое имя:

- не должно совпадать ни с одним из следующих имен:
  - USERS, ADMINS, GUESTS, PUBLIC, LOCAL а также зарезервированными словами SQL, перечисленными в книге *SQL Reference*.
- не должно начинаться с:
  - SQL, SYS или IBM
- может содержать символы:
  - Буквы от A до Z

    Примечание: В некоторых операционных системах ID пользователей и пароли регистрозависимы. Посмотрите в документации правила для вашей операционной системы.
  - Цифры от 0 до 9
  - @, # или $
- Длина ID пользователя не должна превышать 30 символов.

Примечание: Вам может потребоваться менять пароли. Обычно такие действия выполняются на сервере, но многие пользователи слабо знакомы с работой в среде сервера, и для них это может представлять значительные трудности. В DB2 UDB есть способ изменить и проверить пароль с клиента. Например, DB2 for OS/390 Версии 5 поддерживает следующий метод изменения пароля пользователя. Если вы получили сообщение SQL1404N "Срок действия пароля истек", измените пароль при помощи оператора CONNECT так:

```
CONNECT TO <база_данных> USER <id_пользователя> USING <пароль>
   NEW <новый_пароль> VERIFY <новый_пароль>
```

Пароль можно изменить также при помощи диалогового окна "Изменение пароля" Ассистента конфигурирования клиента DB2. Дополнительные сведения об этих методах изменения пароля смотрите в справочнике *SQL Reference* и в электронной справке Ассистента конфигурирования клиента.

Имена схем

Следующие имена схем зарезервированы и использовать их нельзя:

- SYSCAT
- SYSFUN
- SYSIBM
- SYSSTAT.

Вообще говоря, не следует использовать имена схем, начинающиеся с SYS во избежание будущих проблем с перенастройкой. Менеджер баз данных не позволит вам создавать триггеры, пользовательские типы и пользовательские функции, имена схем которых начинаются с SYS.

Рекомендуется также не использовать имя схемы SESSION. К этой схеме должны относиться объявленные временные таблицы. Поэтому возможна ситуация, когда программа объявит временную таблицу с именем, идентичным имени одной из постоянных таблиц, что может привести к сложностям в работе. Не используйте схему SESSION, если вы не работаете с объявленными временными таблицами.

Имена групп и пользователей

В системах на основе UNIX у группы и у пользователя могут быть одинаковые имена. В операторе GRANT надо указать, имеете ли вы в виду группу или же пользователя. В операторе REVOKE необходимость задания пользователя или группы зависит от того, есть ли в таблицах каталогов авторизации строки с одинаковыми GRANTEE, но разными значениями GRANTEETYPE.

В системах на основе OS/2 у группы и у пользователя не может быть одинаковых имен.

В Windows NT имена локальных групп, глобальных групп и ID пользователей не могут совпадать.

Имена групп не могут быть длиннее 8 символов.

Имена объектов

К объектам баз данных относятся:

- Схемы
- Таблицы
- Производные таблицы
- Столбцы
- Индексы
- Пользовательские функции
- Пользовательские типы
- Триггеры
- Алиасы
- Табличные пространства
- Хранимые процедуры
- Методы
- Группы узлов
- Пулы буферов
- Мониторы событий.

Задаваемое имя объекта базы данных:

- Может содержать от 1 до 18 символов (байт)

  Примечание: Исключения из этого правила:
  - Имена схем и столбцов могут содержать от 1 до 30 символов
  - Имена таблиц, производных таблиц, внутриоператорные имена и алиасы могут содержать от 1 до 128 символов.
- Должно начинаться с одного из следующих символов:
  - Буквы от A до Z (символы нижнего регистра преобразуются в верхний)
  - Допустимые буквы национальных алфавитов
  - В многобайтных средах - многобайтные символы, за исключением многобайтных пробелов.
- Может содержать символы:

– Буквы от A до Z (символы нижнего регистра преобразуются в верхний)
– Допустимые буквы национальных алфавитов
– Цифры от 0 до 9
– @, #, $ и _ (подчеркивание)
– В многобайтных средах - многобайтные символы, за исключением многобайтных пробелов.

Можно использовать ключевые слова. Если ключевое слово используется в контексте, где его можно принять за ключевое слово SQL, для него надо использовать ограничители. Информацию об идентификаторах с ограничителями смотрите в справочнике *SQL Reference*.

Для максимальной переносимости используйте зарезервированные слова IBM SQL и ISO/ANSI SQL92. Список этих слов приводится в справочнике *SQL Reference*.

Примечания:

1. Используя идентификаторы с ограничителями, можно создать объекты, имена которых нарушают приведенные правила; однако при последующем использовании такого объекта могут возникнуть ошибки. Чтобы избежать потенциальных проблем при использовании базы данных, **не нарушайте** приведенные выше правила.

   Например, если создать столбец, в имя которого входит символ + или −, и в дальнейшем использовать этот столбец в качестве индексного, вы столкнетесь с проблемами при попытке реорганизовать эту таблицу.

Имена объектов баз данных объединения

К объектам баз данных объединения относятся:

- Спецификации индексов
- Псевдонимы
- Серверы
- Оболочки
- Отображения функций
- Отображения типов
- Отображения пользователей.

При именовании объектов баз данных объединения есть определенные ограничения. Полный список ограничений и требований для имен объектов и идентификаторов приводится в справочнике *SQL Reference*. Кратко говоря, имена объектов:

- НЕ должны быть слишком длинными. Псевдонимы, имена отображений, спецификаций индексов, серверов и оболочек не должны быть длиннее 128 битов.
- Должны начинаться с одного из следующих символов:
  - От A до Z (имена, не заключенные в кавычки, преобразуются в верхний регистр)
  - Допустимые буквы национальных алфавитов
  - В многобайтных средах - многобайтные символы, за исключением многобайтных пробелов.
- Должны подчиняться внутренним правилам именования. В качестве остальных символов имени можно использовать:
  - Буквы от A до Z
  - Допустимые буквы национальных алфавитов
  - Цифры от 0 до 9
  - @, #, $ и _ (подчеркивание)
  - В многобайтных средах - многобайтные символы, за исключением многобайтных пробелов.

Можно использовать ключевые слова. Если ключевое слово используется в контексте, где его можно принять за ключевое слово SQL, для него надо использовать ограничители. Информацию об идентификаторах с ограничителями смотрите в справочнике *SQL Reference*.

Для максимальной переносимости используйте зарезервированные слова IBM SQL и ISO/ANSI SQL92. Список этих слов приводится в справочнике *SQL Reference*.

Опции (сервер, псевдоним) и значения опций не должны быть длиннее 255 байт.

Сохранение регистрозависимых значений в системе объединения

В распределенных требованиях бывает необходимым задание идентификаторов и паролей, которые на источнике данных считаются регистрозависимыми. Чтобы они правильно передавались на источник данных, выполняйте следующие правила:

- Задавайте их в нужном регистре, заключая в двойные кавычки.
- При задании ID пользователя установите для опции сервера fold_id значение ″n″ (“то есть не менять регистр”) для этого источника данных. При задании пароля установите для опции сервера fold_pw значение ″n″ для этого источника данных.

  Есть и другие варианты для ID пользователей и паролей. Если источник данных требует задавать ID пользователя в нижнем регистре, вы можете задать его в любом регистре и установить для опции сервера fold_id значение ″l″ (“Посылать ID на источник данных в нижнем регистре”). Если источник

данных требует задавать ID пользователя в верхнем регистре, вы можете задать его в любом регистре и установить для опции сервера fold_id значение ″u″ (“Посылать ID на источник данных в верхнем регистре”). Аналогичным образом, если требуется задавать пароль в нижнем или в верхнем регистре, это можно сделать, задав для опции сервера fold_pw server значение ″l″ или ″u″.

Дополнительную информацию об опциях сервера смотрите в разделе ″Использование опций сервера для определения характеристик источников данных и настройки процесса аутентификации″ книги *Руководство администратора: Реализация* .

- Если вы заключаете регистрозависимый идентификатор в двойные кавычки в командной строке операционной системы, надо обеспечить правильность анализа системой этих двойных кавычек. Для этого:
  - В системе на основе UNIX заключите весь оператор в одинарные кавычки.
  - В системе Windows NT перед каждой кавычкой ставьте обратную дробную черту.

  Например, на источниках данных семейства DB2 многие идентификаторы с ограничителями регистрозависимы. Предположим, вы хотите создать псевдоним `NICK1` для производной таблицы DB2 for CS `"my_schema"."wkly_sal"`, которая находится на источнике данных под названием `NORBASE`.

  В командной строке системы на основе UNIX введите:

  ```
  db2 'create nickname nick1 for norbase."my_schema"."wkly_sal"'
  ```

  В командной строке Windows NT введите:

  ```
  db2 create nickname nick1 for norbase.\"my_schema\".\"wkly_sal\"
  ```

  Если вы вводите оператор в командной строке DB2, одинарные кавычки или обратные косые черты не нужны. Например, в командной строке DB2, независимо от системы UNIX или Windows NT, надо ввести:

  ```
  create nickname nick1 for norbase."my_schema"."wkly_sal"
  ```

Приложение C. Планирование перенастройки баз данных

В этом разделе приводится обзор процесса перенастройки. Обратите внимание на то, что базы данных DB2 UDB Версии 6 не нужно перенастраивать в Версию 7. Подробную информацию о перенастройке баз данных DB2 UDB Версии 5.*x* можно найти в руководстве *Quick Beginnings* для вашей операционной системы.

Когда вы перенастраиваете базу данных:

- Перенастраиваются следующие объекты базы данных:
  - Файл конфигурации базы данных
  - Таблицы системного каталога базы данных
  - Каталоги базы данных
  - Заголовок файла журнала базы данных
- Таблицы системного каталога изменяются следующим образом:
  - Добавляются новые столбцы.
  - Создаются новые таблицы.
  - Набор производных таблиц каталога перенастраивается, и в схеме SYSCAT создается набор новых производных таблиц.
  - В схеме SYSSTAT создается набор изменяемых производных таблиц каталога.
  - Набор скалярных функций общего назначения сохраняется, и в схеме SYSFUN создается новый набор скалярных функций общего назначения. При перенастройке базы данных только скалярная функция SYSFUN.DIFFERENCE удаляется и пересоздается.
- В каталоге базы данных создается файл хронологии базы данных и его теневая копия. Этот файл содержит сводку информации резервного копирования, которую можно использовать при восстановлении базы данных; она обновляется при проведении определенных операций с базой данных. Сводка информации резервного копирования хранится также для операций резервного копирования и восстановления в табличном пространстве.

Особенности перенастройки

Чтобы успешно перенастроить базу данных, созданную предыдущей версией менеджера баз данных, необходимо учесть следующее:

- “Ограничения перенастройки” на стр. 382
- “Защита и авторизация” на стр. 382
- “Требования к памяти” на стр. 382

- “Несовместимость выпусков” на стр. 383

Ограничения перенастройки

Существуют определенные предварительные условия и ограничения, которые надо учитывать, прежде чем пытаться перенастроить базу данных в Версию 7:

- Перенастройка поддерживается только для Версий V5.x или V6. Перенастройка DB2 V1.2 Parallel Edition не поддерживается. Ранние версии DB2 должны быть перенастроены в Версии V5.x или V6, прежде чем перенастраивать их в V7.
- Клиент Версии V7 может передать команду перенастройки базы данных на сервер V7, но для клиентов более ранних версий DB2 эта возможность не поддерживается.
- Не поддерживается перенастройка с одной платформы на другую.
- Пользовательские объекты в базе данных не могут использовать зарезервированные имена схем V7 в качестве спецификаторов объектов. Зарезервированы следующие имена схем: SYSCAT, SYSSTAT и SYSFUN.
- Пользовательские особые типы с именами BIGINT, REAL, DATALINK или REFERENCE надо переименовать перед перенастройкой базы данных.
- Нельзя перенастраивать базу данных, которая находится в одном из следующих состояний:
  - Отложенное резервное копирование
  - Отложенный повтор
  - Одно или несколько табличных пространств в ненормальном состоянии
  - Несовместимая транзакция
- Восстановление ранних (V5.x или V6) резервных копий базы данных поддерживается, но повтор транзакций из журналов ранних версий не поддерживается.

Защита и авторизация

Для перенастройки базы данных необходимы полномочия SYSADM.

Требования к памяти

В процессе перенастройки требуется пространство как для старых, так и для новых каталогов. Размер требуемого пространства на диске зависит от сложности базы данных, количества и размера объектов базы данных. К объектам относятся все таблицы и производные таблицы. Необходимо освободить по крайней мере в два раза большее дисковое пространство, чем занимает каталог базы данных в текущий момент. Если дискового пространства не хватит, перенастройка завершится неудачно.

Если ваше табличное пространство SYSCAT - типа SMS, надо также учесть обновление параметров конфигурации базы данных, связанных с файлами журнала. Надо увеличить значения параметров *logfilsiz*, *logprimary* и *logsecond*, чтобы предотвратить ошибку из-за нехватки места для этих файлов (SQL1704N

с кодом причины 3). Если эта ошибка произошла, увеличьте параметры пространства для журналов и перезапустите команду MIGRATE DATABASE.

Несовместимость выпусков

Планируя перенастройку базы данных, учтите возможность несовместимости двух версий продукта.

Чтобы воспользоваться преимуществами Версии 7, после перенастройки баз данных надо выполнить настройку баз данных и конфигурацию менеджера баз данных. Советуем для этого записать и сравнить значения параметров конфигурации до и после перенастройки. (Описание команд GET DATABASE CONFIGURATION и GET DATABASE MANAGER CONFIGURATION смотрите в книге *Command Reference*.)

Перенастройка базы данных

Ниже приведены шаги, необходимые для перенастройки базы данных. Перед перенастройкой надо запустить менеджер баз данных.

ПЕРЕД ПЕРЕНАСТРОЙКОЙ:

Примечание: Подготовительные шаги необходимо выполнить в предыдущем выпуске (то есть в выпуске, установленном до перенастройки или установки нового выпуска).

1. Убедитесь, что все условия раздела “Ограничения перенастройки” на стр. 382 выполнены.
2. Отсоедините все прикладные программы и всех конечных пользователей от всех перенастраиваемых баз данных (используйте команду LIST APPLICATIONS и, если необходимо, FORCE APPLICATIONS).
3. Используйте утилиту подготовки к перенастройке DB2CKMIG, чтобы проверить, можно ли перенастроить базу данных (подробную информацию об использовании этой утилиты смотрите в книге *Quick Beginnings* для вашей платформы). Обратите внимание на то, что в среде Windows NT или OS/2 вам предлагается запустить эту утилиту в процессе установки, а в системах на основе UNIX она автоматически активируется при перенастройке экземпляра.
4. Сделайте резервную копию базы данных.

   Перенастройка - необратимый процесс. Если вы скопируете базу данных до изменения зарезервированных имен схем Версии 6, вы не сможете восстановить базу данных при помощи DB2 UDB Версии 7. Чтобы восстановить базу данных, вам придется воспользоваться предыдущей версией менеджера баз данных.

   Внимание! Если у вас нет резервной копии базы данных, а перенастройка завершилась неудачно, вы не сможете восстановить базу данных ни с помощью DB2 UDB Версии 7, ни с помощью предыдущей версии менеджера баз данных.

Надо понимать также, что любые транзакции базы данных, выполненные после резервного копирования, но до завершения перехода на Версию 7, будут потеряны. Это означает, что, если через некоторое время после завершения установки Версии 7 и перенастройки в нее базу данных понадобится восстановить (на уровне Версии 7), записи журнала, сделанные до установки Версии 7, нельзя будет использовать для повтора транзакций.

ПЕРЕНАСТРОЙКА:

5. Перенастройте базу данных, используя одну из следующих команд:
   - MIGRATE DATABASE
   - RESTORE DATABASE (при восстановлении полной резервной копии базы данных).
   - sqlemgdb - API перенастройки баз данных.

   В OS/2: Утилита перенастройки DB2CIDMG, работающая в среде Configuration/Installation/Distribution (CID), доступна только в DB2 for OS/2. Она позволяет проводить удаленную автоматическую установку и конфигурирование на сетевых рабочих станциях. Для перенастройки CID в сети должна работать NetView DM/2.

   В системах на основе UNIX: В книге *Quick Beginnings* для вашей платформы описано, что делать, если вы не хотите перенастраивать все базы данных для данного экземпляра.

ПОСЛЕ ПЕРЕНАСТРОЙКИ:

6. (Необязательно) Воспользуйтесь утилитой DB2UIDDL, чтобы облегчить управление поэтапной перенастройкой индексов уникальности в соответствии с вашим расписанием. (Для баз данных DB2 Версии 5, которые были созданы в Версии 5, эта утилита не требуется для отложенной проверки уникальности, потому что для всех индексов уникальности, созданных в Версии 5, это уже предполагается. Однако для баз данных, которые раньше были перенастроены в Версию 5, эта возможность не задана автоматически, если только вы не пользовались утилитой DB2UIDDL для изменения индексов уникальности.) Эта утилита порождает операторы CREATE UNIQUE INDEX для индексов уникальности пользовательских таблиц и записывает их в файл. Запуская этот файл как командный файл процессора командной строки DB2, вы преобразуете индекс уникальности в Версию 7. Подробную информацию об использовании этой утилиты смотрите в книгах *Quick Beginnings*.
7. (Необязательно) Запустите команду RUNSTATS для таблиц, которые особенно важны для выполнения запросов SQL. Прежняя статистика остается в перенастроенной базе данных и не обновляется, пока вы не выполните команду RUNSTATS.

8. (Необязательно) Воспользуйтесь утилитой DB2RBIND для перепроверки достоверности всех пакетов или же разрешите неявное выполнение этой операции при первом использовании пакета.
9. (Необязательно) Перенастройте таблицы объяснения, если вы собираетесь использовать их в Версии 7. Дополнительную информацию смотрите в разделе ″Утилита объяснения SQL″ руководства *Руководство администратора: Производительность*.
10. Настройте базу данных и параметры конфигурации менеджера баз данных, чтобы воспользоваться преимуществами Версии 7.

Приложение D. Несовместимость выпусков

В этом разделе перечислены несовместимости между DB2 Universal Database и предыдущими выпусками DB2.

Несовместимостью мы называем некоторую часть DB2 Universal Database, которая работает иначе, чем в предыдущем выпуске. При работе существующей прикладной программы несовместимость может привести к неожиданному результату, к необходимости изменить прикладную программу или к снижению производительности. В данном контексте под ″прикладной программой″ понимаются:

- Код прикладной программы
- Утилиты независимых производителей
- Интерактивные запросы SQL
- Команда или вызов API.

Ниже описаны несовместимости между Версиями 6 и 7 DB2 Universal Database. Они сгруппированы по следующим категориям:

- Производные таблицы системного каталога
- Прикладное программирование
- SQL
- Защита и настройка баз данных
- Утилиты и инструменты
- Возможности соединений и сосуществование
- Параметры конфигурации

В каждом разделе описаны несовместимости, их симптомы или влияние и возможные способы решения проблем. В начале описания приводится также индикатор операционной системы, к которой относится эта несовместимость:

WIN Платформы Microsoft Windows, поддерживаемые DB2

UNIX Платформы на основе UNIX, поддерживаемые DB2

OS/2 OS/2

Примечание: Как и в DB2 Universal Database Версии 6, клиенты Версии 1.x и Версии 2.x, в том числе клиенты, поставляемые с серверами DB2 Parallel Edition Версии 1.2, более не поддерживаются.

Планируемые несовместимости в DB2 Universal Database

В этом разделе описаны будущие несовместимости, которые пользователи DB2 Universal Database должны иметь в виду при написании новых или модификации существующих прикладных программ. Это облегчит перенастройку в будущие версии DB2 UDB.

Производные таблицы только для чтения в будущей версии DB2 Universal Database

WIN	UNIX	OS/2

Изменение
Производные таблицы системного каталога станут производными таблицами только для чтения. Производные таблицы SYSSTAT останутся изменяемыми.

Симптом
Операторы UPDATE, применяемые к столбцам производных таблиц SYSCAT, будут вызывать ошибки.

Объяснение
В коде инструмента или прикладной программы задано изменение значений в каталоге путем изменения столбца, определенного в производной таблице SYSCAT.

Решение
Измените инструмент или прикладную программу, чтобы менять каталог путем изменения столбца, определенного в производной таблице SYSSTAT.

PK_COLNAMES и FK_COLNAMES в будущей версии DB2 Universal Database

WIN	UNIX	OS/2

Изменение
SYSCAT.REFERENCES не будет включать столбцы PK_COLNAMES и FK_COLNAMES.

Симптом
Столбец не существует; выдается сообщение об ошибке.

Объяснение
В коде инструмента или прикладной программы содержится использование устаревших столбцов PK_COLNAMES и FK_COLNAMES.

Решение
Измените инструмент или прикладную программу, чтобы использовать вместо SYSCAT.REFERENCES производную таблицу SYSCAT.KEYCOLUSE.

Столбец COLNAMES в будущей версии DB2 Universal Database

WIN	UNIX	OS/2

Изменение
SYSCAT.INDEXES не будет содержать столбец COLNAMES.

Симптом
Столбец не существует; выдается сообщение об ошибке.

Объяснение
В коде инструмента или прикладной программы содержится использование устаревшего столбца COLNAMES.

Решение
Измените инструмент или прикладную программу, чтобы использовать вместо SYSCAT.INDEXES производную таблицу SYSCAT.INDEXCOLUSE.

Несовместимости DB2 Universal Database Версии 7

В этом разделе описаны несовместимости DB2 Universal Database Версии 7 с предыдущими версиями системы.

Прикладное программирование

Универсальный клиент Query Patroller

WIN	UNIX	OS/2

Изменение: Новая версия CAE (Client Application Enabler) будет работать только с сервером Query Patroller Версии 7, так как использует новые хранимые процедуры. CAE - это программный интерфейс к DB2, с помощью которого все прикладные программы получают в конечном счете доступ к базе данных.

Симптом: Если CAE работает с сервером устаревшей версии, выдается сообщение SQL29001.

Функции преобразования объектов и структурные типы

WIN	UNIX	OS/2

Изменение: Есть небольшая потенциально возможная несовместимость между клиентом версий до 7 и сервером Версии 7, связанная с изменениями в SQLDA.

Как описано в *Application Development Guide*, восьмой байт второй SQLVAR теперь может принимать значение X'12' (в дополнение к значениям X'00' и X'01'). Это значение может повлиять на поведение прикладных программ, где такая возможность не предусмотрена.

Решение: Поскольку в будущих версиях могут быть добавлены другие значения этого поля, мы советуем разработчикам при проверке задавать значения явно.

Версии файлов классов и jar-файлов, используемых виртуальной Java-машиной

WIN	UNIX	OS/2

Изменение: Раньше, когда запускалась хранимая Java-процедура или пользовательская функция, виртуальная Java-машина блокировала все файлы, указанные в CLASSPATH (включая файлы в `sqllib/function`). Виртуальная Java-машина использовала эти файлы вплоть до остановки менеджера баз данных. Теперь в зависимости от среды, в которой запускается хранимая процедура или пользовательская функция (то есть в зависимости от значения параметра конфигурации менеджера баз данных *keepdari* и от того, является ли хранимая процедура изолированной), обновляемые классы позволят заменять файлы классов и jar-файлы, не останавливая менеджер баз данных. Это отличается от поведения предыдущих версий.

Изменение функциональных возможностей команд Install, Replace и Remove для файлов jar

WIN	UNIX	OS/2

Изменение: Раньше установка файла jar вызывала очистку всех процессов DARI. Таким образом, при следующем вызове гарантированно выбирался новый класс хранимых процедур. Теперь команды jar не очищают процессы DARI. Чтобы обеспечить выбор классов из только что установленных или замененных файлов jar, выполните явно команду SQLEJ.REFRESH_CLASSES.

Другая несовместимость, связанная с отказом от очистки процессов DARI - это то, что для изолированных хранимых процедур при значении параметра конфигурации менеджера баз данных *keepdari* "YES" клиенты могут получить различные версии файлов jar. Рассмотрим следующий сценарий:

1. Пользователь A заменяет файл jar и не обновляет классы.
2. Далее он вызывает из jar хранимую процедуру. Если этот вызов использует тот же процесс DARI, пользователь A получит устаревшую версию файла jar.
3. Пользователь B вызывает ту же хранимую процедуру. Этот вызов использует новый процесс DARI, а это означает, что вновь созданный загрузчик классов выберет новую версию файла jar.

Другими словами, если после операций jar не обновлять классы, могут быть вызваны хранимые процедуры из разных версий файлов jar в зависимости от того, какие процессы DARI используются. Это отличается от предыдущего поведения, где (путем очистки процессов DARI) гарантировалось использование новых классов.

Несовместимость 32-битных прикладных программ

	UNIX	

Изменение: 32-битные выполняемые программы (прикладные программы DB2) не будут работать с новым 64-битным механизмом баз данных.

Симптом: Не удается связать прикладную программу. При попытке связать 32-битные объекты с 64-битной библиотекой прикладных программ DB2 выдается сообщение об ошибке компоновщика операционной системы.

Решение: Прикладную программу надо перекомпилировать как 64-битную и пересвязать с новыми 64-битными библиотеками DB2.

Изменение поля длины в Scratchpad

WIN	UNIX	OS/2

Изменение: Любая пользовательская функция, изменяющая поле длины в передаваемой ей области scratchpad, теперь получит SQLCODE -450.

Симптом: Пользовательская функция, изменяющая поле длины области scratchpad, завершается неудачно. Вызывающий оператор получит SQLCODE -450 с подставленной схемой и именем конкретной функции.

Решение: Перепишите тело пользовательской функции, чтобы она не меняла поле длины области scratchpad.

SQL

Прикладные программы, использующие обычные таблицы в схеме SESSION

WIN	UNIX	OS/2

Изменение: Временные таблицы могут размещаться только в схеме SESSION; теперь эта схема используется DB2, чтобы указать, что таблица может иметь отношение к временной таблице. Однако SESSION не является ключевым словом, зарезервированным для временных таблиц; эту схему можно использовать для обычных таблиц базы. Это значит, что в прикладной

программе может оказаться, что одновременно существуют настоящая таблица SESSION.T1 и объявленная временная таблица SESSION.T1. Если при связывании пакета встречается статический оператор со ссылкой на таблицу со спецификатором (явным или неявным) ″SESSION″, ни раздел, ни зависимости этого оператора не будут записаны в каталоги. Вместо этого данный раздел надо будет инкрементно связать во время выполнения. При этом копия раздела будет помещена в динамический кэш SQL, причем эта копия будет доступной только одному экземпляру прикладной программы. Если во время выполнения объявленная временная таблица с соответствующим именем существует, будет использована она, даже если существует постоянная таблица базы с тем же именем.

Симптом: В Версии 6 (и более ранних) любой пакет со статическими операторами, ссылающимися на таблицы из схемы SESSION, всегда относился к постоянной таблице базы. При связывании пакета раздел, как и соответствующие записи зависимостей этого оператора, сохранялся в каталогах. В Версии 7 эти операторы не связываются во время связывания, а во время выполнения ссылки могут быть разрешены как ссылки на объявленную временную таблицу. Таким образом, могут возникнуть следующие ситуации:

- Перенастройка из Версии 5. Если такой пакет существовал в Версии 5, он будет снова связан в Версии 6 и статические операторы теперь будут связаны инкрементно. Это может повлиять на производительность, потому что инкрементно связанные разделы ведут себя подобно кэшированному динамическому SQL, за исключением того, что кэшированный динамический раздел не может совместно использоваться другими прикладными программами (и даже разными экземплярами одной и той же выполняемой прикладной программы).
- Перенастройка из Версии 6 в Версию 7. Если такой пакет существовал в Версии 6, он необязательно будет пересвязан в Версии 7. Вместо этого операторы будут работать с ним как с обычным статическим SQL, используя раздел, сохраненный в каталоге во время исходного связывания. Однако, если данный пакет повторно связывается (явно или неявно), операторы в пакете со ссылками на таблицы схемы SESSION не будут записываться в каталог и потребуют инкрементного связывания. Это может ухудшить производительность.

В целом можно сказать, что любые связанные в Версии 7 пакеты со статическими операторами, в которых есть ссылки на таблицы схемы SESSION, не будут теперь выполняться как статический SQL, потому что для них требуется инкрементное связывание. Если в процессе прикладной программы будет выполнен оператор DECLARE GLOBAL TEMPORARY TABLE для таблицы с таким же именем, как у существующей таблицы, производной таблицы или алиаса схемы SESSION, ссылки на эти объекты всегда будут считаться ссылками на объявленную временную таблицу.

Решение: Если это возможно, измените имена схем постоянных таблиц с ″SESSION″ на другие. В противном случае надо иметь в виду последствия для производительности и возможные конфликты с объявленными временными таблицами.

Следующий запрос можно использовать для определения таблиц, производных таблиц и алиасов, на которые может повлиять использование прикладной программой временных таблиц:

```
select tabschema, tabname from SYSCAT.TABLES where tabschema = 'SESSION'
```

Следующий запрос можно использовать для определения связанных пакетов Версии 7, у которых есть статические разделы, хранящиеся в каталогах, и чье поведение может измениться, если пакет связывается повторно (это относится только к перенастройке с Версии 6 в Версию 7):

```
select pkgschema, pkgname, bschema, bname from syscat.packagedep
   where bschema = 'SESSION' and btype in ('T', 'V', 'I')
```

Утилиты и инструменты

Менеджер файлов связей данных и фильтр файловых систем в Solaris

	UNIX	

Изменение: Менеджер файлов связей данных и фильтр файловых систем не поддерживаются в операционной системе Solaris OS 2.5.1.

db2set в AIX и Solaris

	UNIX	

Изменение: Команда ″db2set -ul (user level - пользовательский уровень)″ и связанные с ней функции не переносятся на AIX или Solaris.

Возможности соединений и сосуществование

Несовместимость 32-битного клиента

WIN	UNIX	OS/2

Изменение: 32-битные клиенты не могут подключаться к экземплярам или соединяться с базами данных на 64-битных серверах.

Симптом: Если на клиенте и на сервере работает код Версии 7, возвращается сообщение SQL1434N; в противном случае попытка подключения или соединения завершается с SQLCODE -30081.

Решение: Используйте 64-битные клиенты.

Несовместимости DB2 Universal Database Версии 6

В этом разделе описаны несовместимости DB2 Universal Database Версии 6 с системами ранних версий.

Производные таблицы системных каталогов

Производные таблицы системных каталогов в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: В производных таблицах системных каталогов были введены новые коды: ″U″ для типизированных таблиц и ″W″ для типизированных производных таблиц.

Симптом: Запросы поиска таблиц и производных таблиц в системных каталогах, использующие код типа ″T″ для таблиц и ″V″ для производных таблиц, не находят более типизированные таблицы и типизированные производные таблицы.

Объяснение: В некоторых системных каталогах, в том числе в производных таблицах системных каталогов TABLES, PACKAGEDEP, TRIGDEP и VIEWDEP, есть столбец TYPE или BTYPE, содержащий однобуквенный код типа. В Версии 5.2 код типа ″T″ использовался для всех таблиц, а ″V″ - для всех производных таблиц. В Версии 6 у нетипизированных таблиц код типа по-прежнему будет ″T″, а для типизированных введен новый код типа ″U″. Подобно этому, у нетипизированных производных таблиц код типа по-прежнему будет ″V″, а у типизированных производных - ″W″. Кроме того, в таблицах системных каталогов кодом типа ″H″ обозначен новый тип таблиц - иерархические таблицы; они не создаются пользователями непосредственно, а используются системой для реализации иерархий таблиц.

Решение: Измените инструмент или прикладную программу, чтобы распознавать коды типизированных таблиц и производных таблиц. Если для инструмента или прикладной программы важно логическое представление таблиц, используйте коды типов ″T″, ″U″, ″V″ и ″W″. Если для инструмента или прикладной программы важно физическое представление таблиц, включая таблицы иерархии, используйте коды типов ″T″ и ″H″.

Имена столбцов первичных и внешних ключей в Версии 6 DB2 Universal Database

WIN	UNIX	OS/2

Изменение: Тип данных у двух столбцов таблицы SYSCAT.REFERENCES - PK_COLNAMES и FK_COLNAMES - изменен с VARCHAR(320) на VARCHAR(640).

Симптом: Имена столбцов первичных или внешних ключей усечены, неправильны или отсутствуют.

Объяснение: Когда в первичном или внешнем ключе используются имена столбцов длиной более 18 байт, невозможно соблюдать прежний формат списка имен столбцов. После столбца, длина имени которого (*n*) больше 18, 20-байтные имена столбцов, отделенные пробелами, будут сдвинуты на *n*-18 байт вправо. Если длина списка имен столбцов больше 640 байт, столбец будет содержать пустую строку.

Решение: Производная таблица SYSCAT.KEYCOLUSE содержит список столбцов, которые составляют первичный ключ, внешний ключ или ключ уникальности; эту таблицу надо использовать вместо столбцов из SYSCAT.REFERENCES. Другой вариант - ограничить длину имен столбцов 18 байтами или ограничить суммарную длину списка столбцов 640 байтами.

Столбец TEXT таблицы SYSCAT.VIEWS в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: При просмотре текста столбца TEXT таблицы SYSCAT.VIEWS он не будет разбиваться на строки. Тип данных изменен с VARCHAR(3600) на CLOB(64K).

Симптом: Прикладная программа или инструмент не выводят полный текст столбца.

Объяснение: Инструменты или прикладные программы, которые ожидают значение столбца TEXT не длиннее 3600 (или 3900) байт, не могут обработать более длинные значения из этого поля. Механизм получения нескольких строк и воссоздания текста, использующий поле SEQNO, более не нужен. Значение SEQNO всегда будет равно 1.

Решение: Измените инструмент или прикладную программу, чтобы они могли обрабатывать значения столбца TEXT длиннее 3600 байт. Другой вариант - переписать TEXT, сократив его до 3600 байт.

Столбец TEXT таблицы SYSCAT.STATEMENTS в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: При просмотре текста столбца TEXT в таблице SYSCAT.STATEMENTS он не будет разбиваться на строки. Тип данных изменен с VARCHAR(3600) на CLOB(64K).

Симптом: Прикладная программа или инструмент не выводят полный текст оператора.

Объяснение: Инструменты или прикладные программы, которые ожидают значение столбца TEXT не длиннее 3600 (или 3900) байт, не могут обработать более длинные значения из этого поля. Механизм получения нескольких строк и воссоздания текста, использующий поле SEQNO, более не нужен. Значение SEQNO всегда будет равно 1.

Решение: Измените инструмент или прикладную программу, чтобы они могли обрабатывать значения столбца TEXT длиннее 3600 байт. Другой вариант - переписать TEXT, сократив его до 3600 байт.

Столбец COLNAMES таблицы SYSCAT.INDEXES в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: Тип данных столбца COLNAMES таблицы SYSCAT.INDEXES изменен с VARCHAR(320) на VARCHAR(640).

Симптом: Имена столбцов отсутствуют в индексе.

Объяснение: В коде инструмента или прикладной программы предусматривается получение данных из столбца с типом данных VARCHAR(320), поэтому они не могут обработать более длинные значения из этого поля.

Решение: Производная таблица SYSCAT.INDEXCOLUSE содержит список столбцов, образующих индекс; эту таблицу надо использовать вместо столбца COLNAMES. Другой вариант - удалить столбец из индекса или сократить имя столбца, чтобы длина списка имен столбцов (с ведущими + или −) не превышала 320 байт.

Столбец TEXT таблицы SYSCAT.CHECKS в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: Тип данных столбца TEXT таблицы CHECKS изменен с CLOB(32K) на CLOB(64K).

Симптом: Условие проверочного ограничения неполно.

Объяснение: В коде инструментов или прикладных программ предусматривается получение данных из столбца с типом данных CLOB(32K), поэтому они не могут обработать более длинные значения из этого поля.

Решение: Измените инструмент или прикладную программу, чтобы они могли обрабатывать значения столбца TEXT длиннее 32 Кбайт. Другой вариант - перепишите условие проверочного ограничения, уложившись в 32 Кбайта.

Тип данных столбцов изменен на BIGINT в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: У нескольких столбцов производных таблиц системных каталогов тип данных был изменен с INTEGER на BIGINT.

Симптом: Значения (особенно статистических показателей) гораздо меньше (или больше), чем ожидалось.

Объяснение: В коде инструментов или прикладных программ предусматривается получение данных из столбца с типом данных INTEGER, поэтому они не могут обработать более длинные значения из этого поля.

Решение: Измените инструмент или прикладную программу, чтобы они могли обрабатывать значения больше максимального или меньше минимального значения поля INTEGER. Другой вариант - измените базовую структуру или код SQL, порождающий значение, которое нельзя представить как INTEGER.

Несоответствие столбцов в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: Новые столбцы не вставляются в конец производных таблиц в определении производных таблиц SYSCAT.

Симптом: Не удается выполнить повторную препроцессорную обработку прикладных программ из-за несоответствия столбцов или типов данных столбцов.

Объяснение: В производные таблицы системных каталогов введены новые столбцы; они размещены оптимальным образом для среды произвольных запросов, а именно, более короткие столбцы помещены перед длинными, а столбец REMARKS всегда идет последним.

Решение: Называйте столбцы в списке выбора явно, вместо того чтобы задавать их в коде ″SELECT *″.

SYSCAT.COLUMNS и SYSCAT.ATTRIBUTES в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: Таблицы SYSCAT.COLUMNS и SYSCAT.ATTRIBUTES теперь содержат записи для наследуемых столбцов и атрибутов.

Симптом: Запросы к таблице SYSCAT.COLUMNS для получения столбцов типизированной таблицы или производной таблицы и запросы, применяемые к таблице SYSCAT.ATTRIBUTES для получения атрибутов структурного типа, могут выдать больше строк в Версии 6, чем в Версии 5.2, если предметом запроса является подтаблица, производная подтаблица, или подтип.

Объяснение: В Версии 5.2 для данной таблицы, производной таблицы, или структурного типа в каталогах COLUMNS и ATTRIBUTES содержались записи только для столбцов и атрибутов, введенных самой этой таблицей, производной таблицей или типом. Столбцы и атрибуты, наследуемые от надтаблиц или надтипов, в каталог не заносились. В Версии 6 COLUMNS и SYSCAT.ATTRIBUTES будут содержать записи для наследуемых столбцов и атрибутов.

Решение: Измените инструмент или прикладную программу, чтобы распознавать новые записи в каталогах COLUMNS и ATTRIBUTES.

Производные таблицы OBJCAT более не поддерживаются в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: Рекурсивные производные таблицы каталога в схеме OBJCAT Версии 5.2 теперь не входят в поставляемый продукт DB2 Universal Database.

Симптом: Запросы к производным таблицам каталогов OBJCAT завершаются неудачно.

Решение: Большая часть информации, хранимой раньше в производных таблицах OBJCAT, теперь включена в обычные производные таблицы каталогов SYSCAT. В большинстве случаев можно получить информацию из производных таблиц системных каталогов. Если вы выполняете перенастройку из Версии 5.2 и существуют производные таблицы каталогов OBJCAT, их надо удалить. Это можно сделать при помощи сценария CLP под именем `objcatdp.db2`, находящегося в подкаталоге `misc` каталога `sqllib`.

Можно также создать собственный набор производных таблиц OBJCAT, эквивалентных производным таблицам каталогов из Версии 5.2.

В Версии 5.2 в разделе ″Приложение E″ справочника *SQL Reference* пользователей предупреждали о том, что производные таблицы каталогов OBJCAT - временная мера, которая не будет поддерживаться в следующих выпусках.

Коды зависимостей изменены в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: В производных таблицах системных каталогов иерархические зависимости, ранее обозначавшиеся кодом ″H″, теперь обозначаются кодом ″O″.

Симптом: Запросы, искавшие иерархические зависимости по коду ″H″ в производных таблицах каталогов, больше не будут правильно работать.

Объяснение: В нескольких системных каталогах, в том числе в производных таблицах системных каталогов PACKAGEDEP, TRIGDEP и VIEWDEP, есть столбец с именем BTYPE. В Версии 5.2 в производных таблицах OBJCAT иерархические зависимости обозначались кодом ″H″. В Версии 6 они обозначаются кодом ″O″.

Решение: Переделайте эти запросы так, чтобы они искали код ″O″.

Таблицы базовых каталогов SYSIBM в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: Ниже приводятся изменения в таблицах базовых каталогов SYSIBM, которые вы можете использовать вместо производных таблиц SYSCAT:

- Удалены поля (они остаются в производных таблицах SYSCAT):
  - SYSSTMT.SEQNO
  - SYSVIEWS.SEQNO
- Переименована таблица каталогов: SYSTRIGDEP в SYSDEPENDENCIES. Кроме того, столбцы BCREATOR и DCREATOR переименованы в BSCHEMA и DSCHEMA соответственно. Производная таблица SYSCAT.TRIGDEP не изменилась.
- Удалены поля (их не было в производных таблицах SYSCAT):
  - SYSATTRIBUTES.DEFAULT_VALUE
  - SYSATTRIBUTES.NULLS
  - SYSCOLUMNS.SERVERTYPE

  - SYSDATATYPES.REFREP_TYPENAME
  - SYSDATATYPES.REFREP_TYPESCHEMA
  - SYSDATATYPES.REFREP_LENGTH
  - SYSDATATYPES.REFREP_SCALE
  - SYSDATATYPES.REFREP_CODEPAGE
  - SYSINDEXES.TEXT

    (Было в производной таблице, но было зарезервировано только для будущего использования).
  - SYSPLANDEP.PUBLICPRIV
  - SYSSECTION.SEQNO
  - SYSTABAUTH.UPDATE_BY_COLS
  - SYSTABAUTH.REF_BY_COLS
  - SYSTABLES.MINPDLENGTH
  - SYSTABLESPACES.READONLY
  - SYSTABLESPACES.REMOVABLEMEDIA
- Изменен тип данных:
  - SYSSECTION.SECTION с VARCHAR(3600) на CLOB(10M)
  - SYSPLANDEP.COLUSAGE с VARCHAR(3000) FOR BIT DATA на BLOB(5K)

Прикладное программирование

Тип данных VARCHAR в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: Максимальный возможный размер типа данных VARCHAR (VARGRAPHIC) в Версии 6 увеличен с 4000 символов (2000 двухбайтных символов) до 32672 символов (16336 двухбайтных символов).

Симптом: Прикладная программа, использующая буферы фиксированной длины (4000 байт) для типа данных VARCHAR (VARGRAPHIC), потенциально может перезаписать или усечь значение данных, если она читает поле VARCHAR, занимающее более 4000 байт, в слишком маленький буфер. Функция CLI - `SQLGetTypeInfo()` - теперь возвращает размер VARCHAR 32672 байт. Прикладные программы CLI, использующие это значение в таблице DDL, могут приводить к ошибкам, так как табличные пространства с достаточным размером страницы не доступны. Подробную информацию о размере страницы табличных пространств смотрите в разделе “Данные пользовательских таблиц” на стр. 127.

Решение: При написании прикладной программы рекомендуем сначала описать столбцы набора результатов (используя оператор DESCRIBE), а потом использовать буферы, размер которых определяется длиной, возвращаемой оператором DESCRIBE.

Позиционированные UPDATE и DELETE в программировании на языке Java в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: При программировании на языке Java в Версии 6 надо учитывать, что позиционированные операторы UPDATE и DELETE по умолчанию используют идентификатор авторизации того, кто связывал пакет указателя. Это отличается от Версии 5.2, в которой использовался идентификатор авторизации того, кто запускал пакет.

Симптом: Пакет, содержащий позиционированные операторы UPDATE и DELETE, может не быть запущен, потому что идентификатор авторизации того, кто связывает пакет, не имеет достаточных полномочий.

Решение: Идентификатору авторизации того, кто связывает пакет, надо дать достаточные полномочия для запуска в пакете позиционированных операторов UPDATE и DELETE. Предоставьте нужные привилегии и пересвяжите пакет.

Изменение синтаксиса условия FOR UPDATE в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: В Версии 5.2 условие FOR UPDATE в операторе SELECT используется в программе SQLJ для идентификации столбцов, которые можно обновить в последующих операторах с указанием позиции UPDATE. В Версии 6 этот синтаксис изменен.

Симптом: Если оператор SELECT содержит условие FOR UPDATE, вы получите сообщение об ошибке SQJ0204E.

Решение: Удалите условие FOR UPDATE из оператора SELECT. Задайте изменяемый итератор с помощью условия объявления итераторов. Например:

```
#sql public iterator DelByName implements sqlj.runtime.ForUpdate(String EmpNo)
   with updateColumns = (salary);
```

Если вы хотите явным образом указать, какие столбцы являются изменяемыми, задайте их с помощью ключевого слова `updateColumns`, используемого совместно с условием WITH.

Дополнительную информацию об объявлении позиционированных итераторов смотрите в руководстве *Application Development Guide*.

Размеры символьных имен в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: DB2 Universal Database Версии 6 поддерживает 128-байтные имена таблиц, производных таблиц и алиасов и 30-байтные имена столбцов. Ранее максимальная длина этих имен составляла 18 байт.

Специальные регистры USER и CURRENT SCHEMA были типа CHAR(8), а теперь - VARCHAR(128). Специальный регистр CURRENT EXPLAIN MODE был типа CHAR(8), а теперь - VARCHAR(254). Выходные параметры встроенных функций TYPE_SCHEMA и TABLE_SCHEMA были типа CHAR(8), а теперь - VARCHAR(128).

Симптом: Если прикладные программы, разработанные до появления Версии 6, применяются к базе данных Версии 6, не использующей длинных имен, поведение прикладных программ не должно изменяться вообще. Однако применение этих прикладных программ к базе данных Версии 6, где *действительно* используются более длинные имена, может привести к некоторым побочным эффектам в зависимости от того, как были написаны прикладные программы.

Ниже приведены несколько примеров:

- Рассмотрим существующую прикладную программу, которая берет имя таблицы или столбца (обычно из производной таблицы каталога) и помещает его в переменную хоста, определенную с длиной 18 байт. Так как ранее (до Версии 6) предельная длина имени таблицы или столбца составляла 18 байт, эта программа могла не проверять бит `sqlwarn1` в SQLCA. Она предполагает (ошибочно), что усечение никогда не происходит.
- Рассмотрим прикладную программу, которая берет имя таблицы или столбца (обычно из производной таблицы каталога) и помещает его в SQLDA, где размер поля `sqldata` был задан на основании поля `sqllen` из условия DESCRIBE в операторе SELECT. При этом программе будет возвращен правильный (неусеченный) результат, даже если используются длинные имена таблиц или столбцов. Если логика другой прикладной программы основана на предположении, что длина имен столбцов не превышает 18 байт, получив более длинное имя, программа может обработать его неожиданным образом; например, вывод более длинных имен столбцов может быть усечен до 18 байт.
- Поле элементов SQLCA (`sqlerrmc`) ограничено 70 байтами, и это может повлиять на существующие программы, пытающиеся вставить строку в таблицу. Реагируя на ошибку SQL0204N, такие программы берут имя

таблицы из поля SQLCA `sqlerrmc`, а потом производят некоторые операции, основанные на этом имени объекта. В ранних версиях DB2 ограничение длины идентификатора таблицы или схемы гарантировало, что имя таблицы поместится в SQLCA полностью. В Версии 6 это не так.

- Прикладная программа, использующая устаревшие API, получит только первые 18 байт имени таблицы.
- Существующие прикладные программы CLI и ODBC, использующие функции схем (такие, как SQLTables(), SQLColumns() и другие), могут столкнуться с проблемами при соединении с сервером, который поддерживает длинные (больше 18 байт) имена. Хотя предупреждения об усечении выдаются, программа может не проверить их и продолжить работу с усеченным именем.

Решение: Лучший способ решения проблем этого типа - предусмотреть в коде прикладной программы обработку более длинных имен таблиц и столбцов. Иначе придется применять старые программы только к тем базам данных Версии 6, где длинные (более 18 байт) имена не используются.

Изменения формата PC/IXF в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: DB2 Universal Database Версии 6 поддерживает 128-байтные имена таблиц, производных таблиц и алиасы и 30-байтные имена столбцов. Ранее максимальная длина этих имен составляла 18 байт.

Симптом: Клиент DB2 Universal Database Версии 5 не может импортировать файл PC/IXF, экспортированный клиентом DB2 Universal Database Версии 6 (ошибка SQL3059N). Файл PC/IXF (экспортированный клиентом DB2 Universal Database Версии 6) не может быть загружен в базу данных DB2 Universal Database Версии 5 (ошибка SQL3059N).

Решение: Используйте для импорта и загрузки информации PC/IXF совместимые версии DB2 Universal Database.

SQLNAME в недублированной SQLVAR в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: DB2 Universal Database Версии 6 поддерживает 30-байтные имена столбцов. Предыдущая версия поддерживала 18-байтные имена. В Версии 5 в соответствии с документацией символ ″0xFF″ помещался в 30-й байт поля SQLNAME для недублированной SQLVAR; для имен, порожденных системой, и имен столбцов, заданных пользователем в условии ″AS″, в 30-й байт помещался символ ″0x00″.

В Версии 6 ″0xFF″ возвращается в 30-м байте только для порожденных системой имен.

Симптом: Любые прикладные программы, проверяющие 30-й байт поля SQLNAME для определения, является ли имя порожденным системой или заданным пользователем, могут получить неожиданные результаты, если длина имени столбца, заданного пользователем - 30 символов. Такие случаи должны быть довольно редки.

Решение: Эти прикладные программы надо модифицировать, чтобы они проверяли 30-й байт поля SQLNAME на равенство ″0xFF″, только если длина этого поля меньше 30 символов. Если длина поля - 30 символов, это значит, что имя задано пользователем.

Устаревшие ключевые слова конфигурации DB2 CLI/ODBC в DB2 Universal Database Версии 6

WIN		

Изменение: При перенастройке в новую версию DB2 UDB вы можете изменить поведение драйвера DB2 CLI/ODBC путем задания набора необязательных ключевых слов в файле `db2cli.ini`.

Для Версии 6 ключевые слова TRANSLATEDLL и TRANSLATEOPTION устарели.

Симптом: Если эти ключевые слова будут найдены, они будут проигнорированы. Вы можно заметить изменения поведения, вызванные игнорированием этих параметров.

Решение: Надо просмотреть новый список допустимых параметров и решить, какие ключевые слова и параметры подходят для вашей среды. Сведения об этих ключевых словах смотрите в книге *CLI Guide and Reference*.

Формат выходного потока монитора событий в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: В выходные потоки монитора событий не включается номер версии. В результате добавление поддержки длинных (более 18 байтов) имен таблиц требует изменения формата выходного потока.

Симптом: Прикладные программы, анализирующие выходные потоки монитора событий, не могут нормально работать.

Решение: Есть две возможности:

- Изменить прикладную программу, чтобы использовать новый поток данных.
- Задать переменную реестра

  ```
  DB2OLDEVMON=evmonname1,evmonname2,...
  ```

  где *evmonname* - имя монитора событий, для которого вы хотите задать запись в старом формате. Обратите внимание на то, что все новые поля в мониторе событий не будут доступны в старом формате.

SQL

Столбцы DATALINK в DB2 Universal Database Версии 6

	UNIX	

Изменение: Для значений DATALINK, вставленных DB2 Universal Database Версии 6, в дескрипторе значений столбцов требуется 4 дополнительных байта.

Симптом: Когда обновляются столбцы DATALINK, созданные в Версии 5.2, на странице набора данных требуются дополнительные 4 байта, чтобы сохранить новые значения столбцов. В результате на странице набора данных может не хватить места для завершения обновления, и ее, может быть, придется переместить на новую страницу. Это перемещение может вызвать нехватку пространства во время обновления.

Решение: Необходимо добавить пространства в вашу систему, чтобы изменения были возможны.

Сигнатуры строчных функций SYSFUN в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: Для многих строчных функций в схеме SYSFUN теперь есть усовершенствованные версии в схеме SYSIBM (встроенные функции). Это функции LCASE, LTRIM, RTRIM и UCASE.

Симптом: При подготовке операторов или создании производных таблиц возвращаемый тип данных любой из этих функций в Версии 6 может быть другим. Это происходит потому, что разрешение встроенных функций (из схемы SYSIBM) обычно происходит до разрешения функций из схемы SYSFUN.

Решение: Никаких действий не требуется. Встроенной функции обычно отдается предпочтение перед функцией из схемы SYSFUN. Поведение предыдущих версий можно воспроизвести, изменив путь SQL (чтобы SYSFUN

шла перед SYSIBM), но это ухудшит производительность. Функцию предыдущей версии можно также вызвать, задав имя этой функции с именем схемы SYSFUN.

Перенастроенные пакеты, производные таблицы, сводные таблицы, триггеры и ограничения, ссылающиеся на эти функции, продолжат использовать версию из схемы SYSFUN, пока это не будет изменено явно, например, при явном связывании пакета или пересоздании производной таблицы, сводной таблицы, триггера или ограничения.

Изменение столбцов SYSTABLE с новым состоянием целостности в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: Состояния ″U″ в столбце CONST_CHECKED таблицы SYSCAT.TABLES изменяются иначе при выполнении оператора SET INTEGRITY ... OFF.

Симптом: До Версии 6 любое состояние ″U″ в столбце CONST_CHECKED при выполнении оператора SET INTEGRITY ... OFF заменялось на состояние ″N″. Теперь состояние ″U″ заменяется на состояние ″W″.

Решение: Никаких действий не требуется. Новое состояние ″W″ в столбце CONST_CHECKED используется для того, чтобы указать, что тип ограничений ранее проверен пользователем и что некоторые данные таблицы, возможно, придется проверять на целостность.

Состояние ″N″ не уточняет, существуют ли старые данные, которые еще не проверены менеджером баз данных. Последующий оператор SET INTEGRITY ... IMMEDIATE CHECKED INCREMENTAL приведет к тому, что менеджер баз данных выдаст сообщение об ошибке, так как целостность данных не может быть гарантирована, если проверяются только новые изменения. С другой стороны, состояние ″W″ может быть обратно заменено на состояние ″U″ (если задана опция INCREMENTAL), чтобы указать, что пользователь по-прежнему сам отвечает за целостность данных в таблице. Если опция INCREMENTAL не задана, менеджер баз данных выберет полную обработку, заменит состояние ″W″ на состояние ″Y″, и возьмет на себя ответственность за целостность данных.

Защита и настройка баз данных

Создание клиентами баз данных в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: Метод, используемый клиентами для создания базы данных.

Симптом: Использование клиента устаревшей версии для создания базы данных приведет к ошибкам.

Решение: Если для создания базы данных используется клиент, удостоверьтесь, что на клиенте и на сервере работает код DB2 одной и той же версии.

Привилегия SELECT, требуемая для иерархии в DB2 Universal Database Версии 6

32-битные системы Windows	UNIX	OS/2

Изменение: Задание ключевого слова ONLY (для таблицы) теперь требует, чтобы у пользователя была привилегия SELECT для всех подтаблиц заданной типизированной таблицы. Подобным образом, задание ключевого слова ONLY (для производной таблицы) теперь требует, чтобы у пользователя была привилегия SELECT для всех производных подтаблиц заданной типизированной таблицы. В предыдущих версиях DB2 привилегия SELECT требовалась только для заданной таблицы или производной таблицы.

Симптом: Есть два возможных признака:

- Ошибка авторизации (SQLCODE -551, SQLSTATE 42501) при повторном связывании пакета, содержащего оператор SQL, в котором задано ключевое слово ONLY в условии FROM, если у ID авторизации, под которым связывался пакет, нет привилегии SELECT для подтаблиц заданной типизированной таблицы (или производной таблицы).
- Если определение производной таблицы или триггера содержит ключевое слово ONLY в условии FROM, производная таблица или триггер продолжают работать нормально. Однако определение производной таблицы или триггера нельзя использовать для создания новой производной таблицы или триггера, пока разработчик не получит привилегии SELECT для всех подтаблиц заданной таблицы (или производной таблицы).

Решение: Надо дать ID авторизации, под которым происходит повторное связывание пакета или создание новой производной таблицы или триггера, привилегию SELECT для всех подтаблиц (и производных подтаблиц) таблицы (или производной таблицы), заданной с ключевым словом ONLY.

Устаревшие переменные реестра профиля и среды в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: Следующие переменные реестра профиля или среды устарели:

- DB2_VECTOR

Решение: Эти переменные больше не нужны.

Утилиты и инструменты

CURRENT EXPLAIN MODE в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: Тип специального регистра ″CURRENT EXPLAIN MODE″ изменен с CHAR(8) на VARCHAR(254).

Симптом: Если прикладная программа полагает, что тип этого регистра - CHAR(8), значение может быть усечено с 254 до 8 байт.

Решение: Переопределите тип всех переменных хоста, куда читается этот специальный регистр, с CHAR(8) на VARCHAR(254).

Это изменение требуется для двух новых значений специального регистра ″CURRENT EXPLAIN MODE″. Это значения ″EVALUATE INDEXES″ и ″RECOMMEND INDEXES″.

Параметры COLUMNS и SORT BUFFER в DB2 Universal Database Версии 6

WIN	UNIX	OS/2

Изменение: В Версии 6 более не поддерживаются параметры USING и SORT BUFFER команды LOAD. Эти параметры игнорируются.

Симптом: Выдается предупреждение о том, что параметры USING и SORT BUFFER более не поддерживаются и будут игнорироваться утилитой загрузки.

Решение: Предупреждение можно игнорировать. Дополнительную информацию смотрите в руководстве *Data Movement Utilities Guide and Reference*.

Возможности соединений и сосуществование

Замена RUMBA на PCOMM в DB2 Universal Database Версии 6

WIN		

Изменение: В Версии 6 RUMBA заменена на PCOMM в Windows NT, Windows 98 и Windows 95 (но не в Windows 3.1).

Симптом: Отсутствует.

Решение: Отсутствует.

Параметры конфигурации

Устаревшие параметры конфигурации базы данных

WIN	UNIX	OS/2

Изменение: Следующие параметры конфигурации баз данных устарели:

- DL_NUM_BACKUP (заменен на параметр конфигурации базы данных NUM_DB_ BACKUP)

Решение: Удалите все ссылки на эти параметры из ваших программ.

Приложение E. Поддержка национальных языков (NLS)

В этом разделе содержится информация о поддержке в DB2 национальных языков (national language support - NLS), включая информацию о поддерживаемых странах, языках и кодовых страницах (кодовых наборах), а также о конфигурировании и использовании функций NLS DB2 в базах данных и прикладных программах.

Поддержка кода страны и кодовой страницы

В Табл. 32 на стр. 412 показаны языки и кодовые наборы, поддерживаемые серверами баз данных, а также отображение этих значений на значения кода страны и кодовой страницы, используемые менеджером баз данных.

Далее приводится описание всех столбцов в этой таблице:

- **Кодовая страница** содержит кодовую страницу стандарта IBM, как она отображается из кодового набора операционной системы.
- **Группа** показывает, является ли кодовая страница однобайтной (″S″) или многобайтной (″D″). К этой букве для создания буквенно-цифровой комбинации добавляется число ″-n″. Совпадение комбинаций означает, что DB2 разрешает соединение и поддерживает преобразование. Например, все группы ″S-1″ можно использовать вместе.
- **Кодовый набор** показывает кодовый набор, связанный с поддерживаемым языком. Этот кодовый набор отображается на кодовую страницу DB2.
- **Терр** содержит двухбуквенный идентификатор территории.
- **Код страны** содержит код страны, используемый внутри менеджера баз данных для поддержки особенностей этой страны.
- **Локаль** содержит значения локалей (национальных версий), поддерживаемые менеджером баз данных.
- **ОС** содержит операционную систему, поддерживающую эти языки и кодовые наборы.
- **Название страны** содержит названия стран.

Таблица 32. Поддерживаемые языки и кодовые наборы

Кодовая страница	Группа	Кодовый набор	Терр	Код страны	Локаль	ОС	Название страны
437	S-1	IBM-437	AL	355	-	OS2	Албания
850	S-1	IBM-850	AL	355	-	OS2	Албания
819	S-1	ISO8859-1	AL	355	sq_AL	AIX	Албания
850	S-1	IBM-850	AL	355	Sq_AL	AIX	Албания
819	S-1	iso88591	AL	355	-	HP	Албания
1051	S-1	roman8	AL	355	-	HP	Албания
819	S-1	ISO8859-1	AL	355	-	Sun	Албания
1252	S-1	1252	AL	355	-	WIN	Албания
1275	S-1	1275	AL	355	-	Mac	Албания
37	S-1	IBM-37	AL	355	-	HOST	Албания
1140	S-1	IBM-1140	AL	355	-	HOST	Албания
864	S-6	IBM-864	AA	785	-	OS2	арабские страны
1046	S-6	IBM-1046	AA	785	Ar_AA	AIX	арабские страны
1089	S-6	ISO8859-6	AA	785	ar_AA	AIX	арабские страны
1089	S-6	iso88596	AA	785	iso88596	HP	арабские страны
1256	S-6	1256	AA	785	-	WIN	арабские страны
420	S-6	IBM-420	AA	785	-	HOST	арабские страны
437	S-1	IBM-437	AU	61	-	OS2	Австралия
850	S-1	IBM-850	AU	61	-	OS2	Австралия
819	S-1	ISO8859-1	AU	61	en_AU	AIX	Австралия
850	S-1	IBM-850	AU	61	En_AU	AIX	Австралия
819	S-1	iso88591	AU	61	-	HP	Австралия
1051	S-1	roman8	AU	61	-	HP	Австралия
819	S-1	ISO8859-1	AU	61	en_AU	Sun	Австралия
819	S-1	ISO8859-1	AU	61	en_AU	SCO	Австралия
1252	S-1	1252	AU	61	-	WIN	Австралия
1275	S-1	1275	AU	61	-	Mac	Австралия
37	S-1	IBM-37	AU	61	-	HOST	Австралия
1140	S-1	IBM-1140	AU	61	-	HOST	Австралия
437	S-1	IBM-437	AT	43	-	OS2	Австрия
850	S-1	IBM-850	AT	43	-	OS2	Австрия
819	S-1	ISO8859-1	AT	43	ge_AT	AIX	Австрия
850	S-1	IBM-850	AT	43	Ge_AT	AIX	Австрия
819	S-1	iso88591	AT	43	-	HP	Австрия
1051	S-1	roman8	AT	43	-	HP	Австрия
819	S-1	ISO8859-1	AT	43	de_AT	SCO	Австрия
819	S-1	ISO-8859-1	AT	43	de_AT	Linux	Австрия
819	S-1	ISO8859-1	AT	43	de_AT	Sun	Австрия
1252	S-1	1252	AT	43	-	WIN	Австрия
1275	S-1	1275	AT	43	-	Mac	Австрия
37	S-1	IBM-37	AT	43	-	HOST	Австрия
1140	S-1	IBM-1140	AT	43	-	HOST	Австрия

Таблица 32. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	Терр	Код страны	Локаль	ОС	Название страны
915	S-5	ISO8859-5	BY	375	-	OS2	Белоруссия
915	S-5	ISO8859-5	BY	375	be_BY	AIX	Белоруссия
1131	S-5	IBM-1131	BY	375	-	OS2	Белоруссия
1251	S-5	1251	BY	375	-	WIN	Белоруссия
1283	S-5	1283	BY	375	-	Mac	Белоруссия
1025	S-5	IBM-1025	BY	375	-	HOST	Белоруссия
437	S-1	IBM-437	BE	32	-	OS2	Бельгия
850	S-1	IBM-850	BE	32	-	OS2	Бельгия
819	S-1	ISO8859-1	BE	32	nl_BE	AIX	Бельгия
850	S-1	IBM-850	BE	32	Nl_BE	AIX	Бельгия
819	S-1	iso88591	BE	32	-	HP	Бельгия
819	S-1	ISO8859-1	BE	32	fr_BE	SCO	Бельгия
819	S-1	ISO8859-1	BE	32	nl_BE	SCO	Бельгия
819	S-1	ISO-8859-1	BE	32	nl_BE	Linux	Бельгия
819	S-1	ISO8859-1	BE	32	nl_BE	Sun	Бельгия
1252	S-1	1252	BE	32	-	WIN	Бельгия
1275	S-1	1275	BE	32	-	Mac	Бельгия
500	S-1	IBM-500	BE	32	-	HOST	Бельгия
1148	S-1	IBM-1148	BE	32	-	HOST	Бельгия
855	S-5	IBM-855	BG	359	-	OS2	Болгария
915	S-5	ISO8859-5	BG	359	-	OS2	Болгария
915	S-5	ISO8859-5	BG	359	bg_BG	AIX	Болгария
915	S-5	iso88595	BG	359	iso88595	HP	Болгария
1251	S-5	1251	BG	359	-	WIN	Болгария
1283	S-5	1283	BG	359	-	Mac	Болгария
1025	S-5	IBM-1025	BG	359	-	HOST	Болгария
850	S-1	IBM-850	BR	55	-	OS2	Бразилия
850	S-1	IBM-850	BR	55	-	AIX	Бразилия
819	S-1	ISO8859-1	BR	55	pt_BR	AIX	Бразилия
819	S-1	ISO8859-1	BR	55	-	HP	Бразилия
819	S-1	ISO8859-1	BR	55	pt_BR	SCO	Бразилия
819	S-1	ISO8859-1	BR	55	pt_BR	Sun	Бразилия
819	S-1	ISO-8859-1	BR	55	pt_BR	Linux	Бразилия
1252	S-1	1252	BR	55	-	WIN	Бразилия
37	S-1	IBM-37	BR	55	-	HOST	Бразилия
1140	S-1	IBM-1140	BR	55	-	HOST	Бразилия

Таблица 32. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	Терр	Код страны	Локаль	ОС	Название страны
850	S-1	IBM-850	CA	1	-	OS2	Канада
850	S-1	IBM-850	CA	1	En_CA	AIX	Канада
819	S-1	ISO8859-1	CA	1	en_CA	AIX	Канада
819	S-1	iso88591	CA	1	iso88591	HP	Канада
1051	S-1	roman8	CA	1	roman8	HP	Канада
819	S-1	ISO8859-1	CA	1	en_CA	SCO	Канада
819	S-1	ISO8859-1	CA	1	fr_CA	SCO	Канада
819	S-1	ISO8859-1	CA	1	en_CA	Sun	Канада
819	S-1	ISO8859-1	CA	1	en_CA	Sun	Канада
819	S-1	ISO-8859-1	CA	1	en_CA	Linux	Канада
1252	S-1	1252	CA	1	-	WIN	Канада
1275	S-1	1275	CA	1	-	Mac	Канада
37	S-1	IBM-37	CA	1	-	HOST	Канада
1140	S-1	IBM-1140	CA	1	-	HOST	Канада
863	S-1	IBM-863	CA	2	-	OS2	Канада (французский)
1381	D-4	IBM-1381	CN	86	-	OS2	Китай (КНР)
1386	D-4	GBK	CN	86	-	OS2	Китай (КНР)
1383	D-4	IBM-eucCN	CN	86	zh_CN	AIX	Китай (КНР)
1386	D-4	GBK	CN	86	Zh_CN.GBK	AIX	Китай (КНР)
1383	D-4	hp15CN	CN	86	hp15CN	HP	Китай (КНР)
1383	D-4	eucCN	CN	86	zh_CN	SCO	Китай (КНР)
1383	D-4	eucCN	CN	86	eucCN	SCO	Китай (КНР)
1383	D-4	gb2312	CN	86	zh	Sun	Китай (КНР)
1381	D-4	IBM-1381	CN	86	-	WIN	Китай (КНР)
1386	D-4	GBK	CN	86	-	WIN	Китай (КНР)
935	D-4	IBM-935	CN	86	-	HOST	Китай (КНР)
1388	D-4	IBM-1388	CN	86	-	HOST	Китай (КНР)
852	S-2	IBM-852	HR	385	-	OS2	Хорватия
912	S-2	ISO8859-2	HR	385	hr_HR	AIX	Хорватия
912	S-2	iso88592	HR	385	iso88592	HP	Хорватия
912	S-2	ISO8859-2	HR	385	ISO8859-2	SCO	Хорватия
912	S-2	ISO-8859-2	HR	385	hr_HR	Linux	Хорватия
1250	S-2	1250	HR	385	-	WIN	Хорватия
1282	S-2	1282	HR	385	-	Mac	Хорватия
870	S-2	IBM-870	HR	385	-	HOST	Хорватия
852	S-2	IBM-852	CZ	421	-	OS2	Чехия
912	S-2	ISO8859-2	CZ	421	cs_CZ	AIX	Чехия
912	S-2	iso88592	CZ	421	iso88592	HP	Чехия
912	S-2	ISO8859-2	CZ	421	ISO8859-2	SCO	Чехия
912	S-2	ISO-8859-2	CZ	421	cs_CZ	Linux	Чехия
1250	S-2	1250	CZ	421	-	WIN	Чехия
1282	S-2	1282	CZ	421	-	Mac	Чехия
870	S-2	IBM-870	CZ	421	-	HOST	Чехия

Таблица 32. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	Терр	Код страны	Локаль	ОС	Название страны
850	S-1	IBM-850	DK	45	-	OS2	Дания
819	S-1	ISO8859-1	DK	45	da_DK	AIX	Дания
850	S-1	IBM-850	DK	45	Da_DK	AIX	Дания
819	S-1	iso88591	DK	45	iso88591	HP	Дания
1051	S-1	roman8	DK	45	roman8	HP	Дания
819	S-1	ISO8859-1	DK	45	da	SCO	Дания
819	S-1	ISO8859-1	DK	45	da_DA	SCO	Дания
819	S-1	ISO8859-1	DK	45	da_DK	SCO	Дания
819	S-1	ISO8859-1	DK	45	da	Sun	Дания
819	S-1	ISO8859-1	DK	45	da	Sun	Дания
819	S-1	ISO-8859-1	DK	45	da_DK	Linux	Дания
1252	S-1	1252	DK	45	-	WIN	Дания
1275	S-1	1275	DK	45	-	Mac	Дания
277	S-1	IBM-277	DK	45	-	HOST	Дания
1142	S-1	IBM-1142	DK	45	-	HOST	Дания
922	S-10	IBM-922	EE	372	-	OS2	Эстония
922	S-10	IBM-922	EE	372	Et_EE	AIX	Эстония
922	S-10	IBM-922	EE	372	-	WIN	Эстония
1122	S-10	IBM-1122	EE	372	-	HOST	Эстония
437	S-1	IBM-437	FI	358	-	OS2	Финляндия
850	S-1	IBM-850	FI	358	-	OS2	Финляндия
819	S-1	ISO8859-1	FI	358	fi_FI	AIX	Финляндия
850	S-1	IBM-850	FI	358	Fi_FI	AIX	Финляндия
819	S-1	iso88591	FI	358	iso88591	HP	Финляндия
819	S-1	ISO8859-1	FI	358	fi	SCO	Финляндия
819	S-1	ISO8859-1	FI	358	fi_FI	SCO	Финляндия
819	S-1	ISO8859-1	FI	358	sv_FI	SCO	Финляндия
819	S-1	ISO8859-1	FI	358	-	Sun	Финляндия
819	S-1	ISO-8859-1	FI	358	fi_FI	Linux	Финляндия
1051	S-1	roman8	FI	358	-	HP	Финляндия
1252	S-1	1252	FI	358	-	WIN	Финляндия
1275	S-1	1275	FI	358	-	Mac	Финляндия
278	S-1	IBM-278	FI	358	-	HOST	Финляндия
1143	S-1	IBM-1143	FI	358	-	HOST	Финляндия
855	S-5	IBM-855	MK	389	-	OS2	Македония
915	S-5	ISO8859-5	MK	389	-	OS2	Македония
915	S-5	ISO8859-5	MK	389	mk_MK	AIX	Македония
915	S-5	iso88595	MK	389	-	HP	Македония
1251	S-5	1251	MK	389	-	WIN	Македония
1283	S-5	1283	MK	389	-	Mac	Македония
1025	S-5	IBM-1025	MK	389	-	HOST	Македония

Таблица 32. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	Терр	Код страны	Локаль	ОС	Название страны
437	S-1	IBM-437	FR	33	-	OS2	Франция
850	S-1	IBM-850	FR	33	-	OS2	Франция
819	S-1	ISO8859-1	FR	33	fr_FR	AIX	Франция
850	S-1	IBM-850	FR	33	Fr_FR	AIX	Франция
819	S-1	iso88591	FR	33	iso88591	HP	Франция
1051	S-1	roman8	FR	33	roman8	HP	Франция
819	S-1	ISO8859-1	FR	33	fr	Sun	Франция
819	S-1	ISO8859-1	FR	33	fr	SCO	Франция
819	S-1	ISO8859-1	FR	33	fr_FR	SCO	Франция
819	S-1	ISO-8859-1	FR	33	fr_FR	Linux	Франция
1252	S-1	1252	FR	33	-	WIN	Франция
1275	S-1	1275	FR	33	-	Mac	Франция
297	S-1	IBM-297	FR	33	-	HOST	Франция
1147	S-1	IBM-1147	FR	33	-	HOST	Франция
437	S-1	IBM-437	DE	49	-	OS2	Германия
850	S-1	IBM-850	DE	49	-	OS2	Германия
819	S-1	ISO8859-1	DE	49	de_DE	AIX	Германия
850	S-1	IBM-850	DE	49	De_DE	AIX	Германия
819	S-1	iso88591	DE	49	iso88591	HP	Германия
1051	S-1	roman8	DE	49	roman8	HP	Германия
819	S-1	ISO8859-1	DE	49	de	SCO	Германия
819	S-1	ISO8859-1	DE	49	de_DE	SCO	Германия
819	S-1	ISO8859-1	DE	49	de	Sun	Германия
819	S-1	ISO-8859-1	DE	49	de_DE	Linux	Германия
1252	S-1	1252	DE	49	-	WIN	Германия
1275	S-1	1275	DE	49	-	Mac	Германия
273	S-1	IBM-273	DE	49	-	HOST	Германия
1141	S-1	IBM-1141	DE	49	-	HOST	Германия
819	S-1	ISO8859-1	DE	49	88591	SINIX	Германия
819	S-1	ISO8859-1	DE	49	6937	SINIX	Германия
813	S-7	ISO8859-7	GR	30	-	OS2	Греция
869	S-7	IBM-869	GR	30	-	OS2	Греция
813	S-7	ISO8859-7	GR	30	el_GR	AIX	Греция
813	S-7	iso88597	GR	30	el_GR.iso88597	HP	Греция
813	S-7	ISO8859-7	GR	30	el_GR.ISO8859-7	SCO	Греция
813	S-7	ISO-8859-7	GR	30	gr_GR	Linux	Греция
737	S-7	737	GR	30	-	WIN	Греция
1253	S-7	1253	GR	30	-	WIN	Греция
1280	S-7	1280	GR	30	-	Mac	Греция
423	S-7	IBM-423	GR	30	-	HOST	Греция
875	S-7	IBM-875	GR	30	-	HOST	Греция

Таблица 32. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	Терр	Код страны	Локаль	ОС	Название страны
852	S-2	IBM-852	HU	36	-	OS2	Венгрия
912	S-2	ISO8859-2	HU	36	hu_HU	AIX	Венгрия
912	S-2	iso88592	HU	36	hu_HU.iso88592	HP	Венгрия
912	S-2	ISO8859-2	HU	36	hu_HU.ISO8859-2	SCO	Венгрия
912	S-2	ISO-8859-2	HU	36	hu_HU	Linux	Венгрия
1250	S-2	1250	HU	36	-	WIN	Венгрия
1282	S-2	1282	HU	36	-	Mac	Венгрия
870	S-2	IBM-870	HU	36	-	HOST	Венгрия
850	S-1	IBM-850	IS	354	-	OS2	Исландия
819	S-1	ISO8859-1	IS	354	is_IS	AIX	Исландия
850	S-1	IBM-850	IS	354	Is_IS	AIX	Исландия
819	S-1	iso88591	IS	354	is_IS.iso88591	HP	Исландия
1051	S-1	roman8	IS	354	is_IS.roman8	HP	Исландия
819	S-1	ISO8859-1	IS	354	is	SCO	Исландия
819	S-1	ISO8859-1	IS	354	is_IS	SCO	Исландия
819	S-1	ISO8859-1	IS	354	-	Sun	Исландия
819	S-1	ISO-8859-1	IS	354	is_IS	Linux	Исландия
1252	S-1	1252	IS	354	-	WIN	Исландия
1275	S-1	1275	IS	354	-	Mac	Исландия
871	S-1	IBM-871	IS	354	-	HOST	Исландия
1149	S-1	IBM-1149	IS	354	-	HOST	Исландия
437	S-1	IBM-437	IE	353	-	OS2	Ирландия
850	S-1	IBM-850	IE	353	-	OS2	Ирландия
819	S-1	ISO8859-1	IE	353	en_IE	AIX	Ирландия
850	S-1	IBM-850	IE	353	En_IE	AIX	Ирландия
819	S-1	iso88591	IE	353	-	HP	Ирландия
1051	S-1	roman8	IE	353	-	HP	Ирландия
819	S-1	ISO8859-1	IE	353	en_IE	Sun	Ирландия
819	S-1	ISO8859-1	IE	353	en_IE.ISO8859-1	SCO	Ирландия
819	S-1	ISO-8859-1	IE	353	en_IE	Linux	Ирландия
1252	S-1	1252	IE	353	-	WIN	Ирландия
1275	S-1	1275	IE	353	-	Mac	Ирландия
285	S-1	IBM-285	IE	353	-	HOST	Ирландия
1146	S-1	IBM-1146	IE	353	-	HOST	Ирландия
806	S-12	IBM-806	IN	91	hi_IN	-	Индия
1137	S-12	IBM-1137	IN	91	-	HOST	Индия
862	S-8	IBM-862	IL	972	-	OS2	Израиль
916	S-8	ISO8859-8	IL	972	iw_IL	AIX	Израиль
916	S-8	ISO-8859-8	IL	972	iw_IL	Linux	Израиль
1255	S-8	1255	IL	972	-	WIN	Израиль
424	S-8	IBM-424	IL	972	-	HOST	Израиль

Таблица 32. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	Терр	Код страны	Локаль	ОС	Название страны
437	S-1	IBM-437	IT	39	-	OS2	Италия
850	S-1	IBM-850	IT	39	-	OS2	Италия
819	S-1	ISO8859-1	IT	39	it_IT	AIX	Италия
850	S-1	IBM-850	IT	39	It_IT	AIX	Италия
819	S-1	iso88591	IT	39	it_IT.iso88591	HP	Италия
1051	S-1	roman8	IT	39	it_IT.roman8	HP	Италия
819	S-1	ISO8859-1	IT	39	it	SCO	Италия
819	S-1	ISO8859-1	IT	39	it_IT	SCO	Италия
819	S-1	ISO8859-1	IT	39	it	Sun	Италия
819	S-1	ISO-8859-1	IT	39	it_IT	Linux	Италия
1252	S-1	1252	IT	39	-	WIN	Италия
1275	S-1	1275	IT	39	-	Mac	Италия
280	S-1	IBM-280	IT	39	-	HOST	Италия
1144	S-1	IBM-1144	IT	39	-	HOST	Италия
932	D-1	IBM-932	JP	81	-	OS2	Япония
942	D-1	IBM-942	JP	81	-	OS2	Япония
943	D-1	IBM-943	JP	81	-	OS2	Япония
954	D-1	IBM-eucJP	JP	81	ja_JP	AIX	Япония
932	D-1	IBM-932	JP	81	Ja_JP	AIX	Япония
954	D-1	eucJP	JP	81	ja_JP.eucJP	HP	Япония
5039	D-1	SJIS	JP	81	ja_JP.SJIS	HP	Япония
954	D-1	eucJP	JP	81	ja	SCO	Япония
954	D-1	eucJP	JP	81	ja_JP	SCO	Япония
954	D-1	eucJP	JP	81	ja_JP.EUC	SCO	Япония
954	D-1	eucJP	JP	81	ja_JP.eucJP	SCO	Япония
954	D-1	eucJP	JP	81	ja	Sun	Япония
954	D-1	EUC-JP	JP	81	ja_JP	Linux	Япония
943	D-1	IBM-943	JP	81	-	WIN	Япония
930	D-1	IBM-930	JP	81	-	HOST	Япония
939	D-1	IBM-939	JP	81	-	HOST	Япония
5026	D-1	IBM-5026	JP	81	-	HOST	Япония
5035	D-1	IBM-5035	JP	81	-	HOST	Япония
1390	D-1		JP	81	-	HOST	Япония
1399	D-1		JP	81	-	HOST	Япония
949	D-3	IBM-949	KR	82	-	OS2	Южная Корея
970	D-3	IBM-eucKR	KR	82	ko_KR	AIX	Южная Корея
970	D-3	eucKR	KR	82	ko_KR.eucKR	HP	Южная Корея
970	D-3	eucKR	KR	82	ko_KR.eucKR	SGI	Южная Корея
970	D-3	5601	KR	82	ko	Sun	Южная Корея
1363	D-3	1363	KR	82	-	WIN	Южная Корея
933	D-3	IBM-933	KR	82	-	HOST	Южная Корея
1364	D-3	IBM-1364	KR	82	-	HOST	Южная Корея

Таблица 32. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	Код Терр	Код страны	Локаль	ОС	Название страны
437	S-1	IBM-437	Lat	3	-	OS2	Латинская Америка
850	S-1	IBM-850	Lat	3	-	OS2	Латинская Америка
819	S-1	ISO8859-1	Lat	3	-	AIX	Латинская Америка
850	S-1	IBM-850	Lat	3	-	AIX	Латинская Америка
819	S-1	iso88591	Lat	3	-	HP	Латинская Америка
819	S-1	ISO8859-1	Lat	3	-	Sun	Латинская Америка
819	S-1	ISO-8859-1	Lat	3	-	Linux	Латинская Америка
1051	S-1	roman8	Lat	3	-	HP	Латинская Америка
1252	S-1	1252	Lat	3	-	WIN	Латинская Америка
1275	S-1	1275	Lat	3	-	Mac	Латинская Америка
284	S-1	IBM-284	Lat	3	-	HOST	Латинская Америка
1145	S-1	IBM-1145	Lat	3	-	HOST	Латинская Америка
921	S-10	IBM-921	LV	371	-	OS2	Латвия
921	S-10	IBM-921	LV	371	Lv_LV	AIX	Латвия
921	S-10	IBM-921	LV	371	-	WIN	Латвия
1112	S-10	IBM-1112	LV	371	-	HOST	Латвия
921	S-10	IBM-921	LT	370	-	OS2	Литва
921	S-10	IBM-921	LT	370	Lt_LT	AIX	Литва
921	S-10	IBM-921	LT	370	-	WIN	Литва
1112	S-10	IBM-1112	LT	370	-	HOST	Литва
437	S-1	IBM-437	NL	31	-	OS2	Нидерланды
850	S-1	IBM-850	NL	31	-	OS2	Нидерланды
819	S-1	ISO8859-1	NL	31	nl_NL	AIX	Нидерланды
850	S-1	IBM-850	NL	31	Nl_NL	AIX	Нидерланды
819	S-1	iso88591	NL	31	nl_NL.iso88591	HP	Нидерланды
1051	S-1	roman8	NL	31	nl_NL.roman8	HP	Нидерланды
819	S-1	ISO8859-1	NL	31	nl	SCO	Нидерланды
819	S-1	ISO8859-1	NL	31	nl_NL	SCO	Нидерланды
819	S-1	ISO8859-1	NL	31	nl	Sun	Нидерланды
819	S-1	ISO-8859-1	NL	31	nl_NL	Linux	Нидерланды
1252	S-1	1252	NL	31	-	WIN	Нидерланды
1275	S-1	1275	NL	31	-	Mac	Нидерланды
37	S-1	IBM-37	NL	31	-	HOST	Нидерланды
1140	S-1	IBM-1140	NL	31	-	HOST	Нидерланды
850	S-1	IBM-850	NZ	64	-	OS2	Новая Зеландия
850	S-1	IBM-850	NZ	64	En_NZ	AIX	Новая Зеландия
819	S-1	ISO8859-1	NZ	64	en_NZ	AIX	Новая Зеландия
819	S-1	ISO8859-1	NZ	64	-	HP	Новая Зеландия
819	S-1	ISO8859-1	NZ	64	en_NZ	SCO	Новая Зеландия
819	S-1	ISO8859-1	NZ	64	en_NZ	Sun	Новая Зеландия
1252	S-1	1252	NZ	64	-	WIN	Новая Зеландия
37	S-1	IBM-37	NZ	64	-	HOST	Новая Зеландия
1140	S-1	IBM-1140	NZ	64	-	HOST	Новая Зеландия

Таблица 32. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	Терр	Код страны	Локаль	ОС	Название страны
850	S-1	IBM-850	NO	47	-	OS2	Норвегия
819	S-1	ISO8859-1	NO	47	no_NO	AIX	Норвегия
850	S-1	IBM-850	NO	47	No_NO	AIX	Норвегия
819	S-1	iso88591	NO	47	no_NO.iso88591	HP	Норвегия
1051	S-1	roman8	NO	47	no_NO.roman8	HP	Норвегия
819	S-1	ISO8859-1	NO	47	no	SCO	Норвегия
819	S-1	ISO8859-1	NO	47	no_NO	SCO	Норвегия
819	S-1	ISO8859-1	NO	47	no	Sun	Норвегия
819	S-1	ISO-8859-1	NO	47	no_NO	Linux	Норвегия
1252	S-1	1252	NO	47	-	WIN	Норвегия
1275	S-1	1275	NO	47	-	Mac	Норвегия
277	S-1	IBM-277	NO	47	-	HOST	Норвегия
1142	S-1	IBM-1142	NO	47	-	HOST	Норвегия
852	S-2	IBM-852	PL	48	-	OS2	Польша
912	S-2	ISO8859-2	PL	48	pl_PL	AIX	Польша
912	S-2	iso88592	PL	48	pl_PL.iso88592	HP	Польша
912	S-2	ISO8859-2	PL	48	pl_PL.ISO8859-2	SCO	Польша
912	S-2	ISO-8859-2	PL	48	pl_PL	Linux	Польша
1250	S-2	1250	PL	48	-	WIN	Польша
1282	S-2	1282	PL	48	-	Mac	Польша
870	S-2	IBM-870	PL	48	-	HOST	Польша
860	S-1	IBM-860	PT	351	-	OS2	Португалия
850	S-1	IBM-850	PT	351	-	OS2	Португалия
819	S-1	ISO8859-1	PT	351	pt_PT	AIX	Португалия
850	S-1	IBM-850	PT	351	Pt_PT	AIX	Португалия
819	S-1	iso88591	PT	351	pt_PT.iso88591	HP	Португалия
1051	S-1	roman8	PT	351	pt_PT.roman8	HP	Португалия
819	S-1	ISO8859-1	PT	351	pt	SCO	Португалия
819	S-1	ISO8859-1	PT	351	pt_PT	SCO	Португалия
819	S-1	ISO8859-1	PT	351	pt	Sun	Португалия
819	S-1	ISO-8859-1	PT	351	pt_PT	Linux	Португалия
1252	S-1	1252	PT	351	-	WIN	Португалия
1275	S-1	1275	PT	351	-	Mac	Португалия
37	S-1	IBM-37	PT	351	-	HOST	Португалия
1140	S-1	IBM-1140	PT	351	-	HOST	Португалия
852	S-2	IBM-852	RO	40	-	OS2	Румыния
912	S-2	ISO8859-2	RO	40	ro_RO	AIX	Румыния
912	S-2	iso88592	RO	40	ro_RO.iso88592	HP	Румыния
912	S-2	ISO8859-2	RO	40	ro_RO.ISO8859-2	SCO	Румыния
912	S-2	ISO-8859-2	RO	40	ro_RO	Linux	Румыния
1250	S-2	1250	RO	40	-	WIN	Румыния
1282	S-2	1282	RO	40	-	Mac	Румыния
870	S-2	IBM-870	RO	40	-	HOST	Румыния

Таблица 32. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	Терр	Код страны	Локаль	ОС	Название страны
866	S-5	IBM-866	RU	7	-	OS2	Россия
915	S-5	ISO8859-5	RU	7	-	OS2	Россия
915	S-5	ISO8859-5	RU	7	ru_RU	AIX	Россия
915	S-5	iso88595	RU	7	ru_RU.iso88595	HP	Россия
915	S-5	ISO8859-5	RU	7	ru_RU.ISO8859-5	SCO	Россия
915	S-5	ISO-8859-5	RU	7	ru_RU	Linux	Россия
1251	S-5	1251	RU	7	-	WIN	Россия
1283	S-5	1283	RU	7	-	Mac	Россия
1025	S-5	IBM-1025	RU	7	-	HOST	Россия
855	S-5	IBM-855	SP	381	-	OS2	Югославия
915	S-5	ISO8859-5	SP	381	-	OS2	Югославия
915	S-5	ISO8859-5	SP	381	sr_SP	AIX	Югославия
915	S-5	iso88595	SP	381	-	HP	Югославия
1251	S-5	1251	SP	381	-	WIN	Югославия
1283	S-5	1283	SP	381	-	Mac	Югославия
1025	S-5	IBM-1025	SP	381	-	HOST	Югославия
852	S-2	IBM-852	SK	422	-	OS2	Словакия
912	S-2	ISO8859-2	SK	422	sk_SK	AIX	Словакия
912	S-2	iso88592	SK	422	sk_SK.iso88592	HP	Словакия
912	S-2	ISO8859-2	SK	422	sk_SK.ISO8859-2	SCO	Словакия
1250	S-2	1250	SK	422	-	WIN	Словакия
1282	S-2	1282	SK	422	-	Mac	Словакия
870	S-2	IBM-870	SK	422	-	HOST	Словакия
852	S-2	IBM-852	SI	386	-	OS2	Словения
912	S-2	ISO8859-2	SI	386	sl_SI	AIX	Словения
912	S-2	iso88592	SI	386	sl_SI.iso88592	HP	Словения
912	S-2	ISO8859-2	SI	386	sl_SI.ISO8859-2	SCO	Словения
912	S-2	ISO-8859-2	SI	386	sl_SI	Linux	Словения
1250	S-2	1250	SI	386	-	WIN	Словения
1282	S-2	1282	SI	386	-	Mac	Словения
870	S-2	IBM-870	SI	386	-	HOST	Словения
437	S-1	IBM-437	ZA	27	-	OS2	ЮАР
850	S-1	IBM-850	ZA	27	-	OS2	ЮАР
819	S-1	ISO8859-1	ZA	27	en_ZA	AIX	ЮАР
850	S-1	IBM-850	ZA	27	En_ZA	AIX	ЮАР
819	S-1	iso88591	ZA	27	-	HP	ЮАР
1051	S-1	roman8	ZA	27	-	HP	ЮАР
819	S-1	ISO8859-1	ZA	27	-	Sun	ЮАР
819	S-1	ISO8859-1	ZA	27	en_ZA.ISO8859-1	SCO	ЮАР
1252	S-1	1252	ZA	27	-	WIN	ЮАР
1275	S-1	1275	ZA	27	-	Mac	ЮАР
285	S-1	IBM-285	ZA	27	-	HOST	ЮАР
1146	S-1	IBM-1146	ZA	27	-	HOST	ЮАР

Таблица 32. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	Терр	Код страны	Локаль	ОС	Название страны
437	S-1	IBM-437	ES	34	-	OS2	Испания
850	S-1	IBM-850	ES	34	-	OS2	Испания
819	S-1	ISO8859-1	ES	34	es_ES	AIX	Испания
850	S-1	IBM-850	ES	34	Es_ES	AIX	Испания
819	S-1	iso88591	ES	34	es_ES.iso88591	HP	Испания
1051	S-1	roman8	ES	34	es_ES.roman8	HP	Испания
819	S-1	ISO8859-1	ES	34	es	Sun	Испания
819	S-1	ISO8859-1	ES	34	es	SCO	Испания
819	S-1	ISO8859-1	ES	34	es_ES	SCO	Испания
819	S-1	ISO-8859-1	ES	34	es_ES	Linux	Испания
1252	S-1	1252	ES	34	-	WIN	Испания
1275	S-1	1275	ES	34	-	Mac	Испания
284	S-1	IBM-284	ES	34	-	HOST	Испания
1145	S-1	IBM-1145	ES	34	-	HOST	Испания
437	S-1	IBM-437	SE	46	-	OS2	Швеция
850	S-1	IBM-850	SE	46	-	OS2	Швеция
819	S-1	ISO8859-1	SE	46	sv_SE	AIX	Швеция
850	S-1	IBM-850	SE	46	Sv_SE	AIX	Швеция
819	S-1	iso88591	SE	46	sv_SE.iso88591	HP	Швеция
1051	S-1	roman8	SE	46	sv_SE.roman8	HP	Швеция
819	S-1	ISO8859-1	SE	46	sv	SCO	Швеция
819	S-1	ISO8859-1	SE	46	sv_SE	SCO	Швеция
819	S-1	ISO8859-1	SE	46	sv	Sun	Швеция
819	S-1	ISO-8859-1	SE	46	sv_SE	Linux	Швеция
1252	S-1	1252	SE	46	-	WIN	Швеция
1275	S-1	1275	SE	46	-	Mac	Швеция
278	S-1	IBM-278	SE	46	-	HOST	Швеция
1143	S-1	IBM-1143	SE	46	-	HOST	Швеция
437	S-1	IBM-437	CH	41	-	OS2	Швейцария
850	S-1	IBM-850	CH	41	-	OS2	Швейцария
819	S-1	ISO8859-1	CH	41	de_CH	AIX	Швейцария
850	S-1	IBM-850	CH	41	De_CH	AIX	Швейцария
819	S-1	iso88591	CH	41	-	HP	Швейцария
1051	S-1	roman8	CH	41	-	HP	Швейцария
819	S-1	ISO8859-1	CH	41	de_CH	SCO	Швейцария
819	S-1	ISO8859-1	CH	41	fr_CH	SCO	Швейцария
819	S-1	ISO8859-1	CH	41	it_CH	SCO	Швейцария
819	S-1	ISO8859-1	CH	41	de_CH	Sun	Швейцария
819	S-1	ISO-8859-1	CH	41	de_CH	Linux	Швейцария
1252	S-1	1252	CH	41	-	WIN	Швейцария
1275	S-1	1275	CH	41	-	Mac	Швейцария
500	S-1	IBM-500	CH	41	-	HOST	Швейцария
1148	S-1	IBM-1148	CH	41	-	HOST	Швейцария

Таблица 32. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	Терр	Код страны	Локаль	ОС	Название страны
938	D-2	IBM-938	TW	88	-	OS2	Тайвань
948	D-2	IBM-948	TW	88	-	OS2	Тайвань
950	D-2	big5	TW	88	-	OS2	Тайвань
950	D-2	big5	TW	88	Zh_TW	AIX	Тайвань
964	D-2	IBM-eucTW	TW	88	zh_TW	AIX	Тайвань
950	D-2	big5	TW	88	zh_TW.big5	HP	Тайвань
964	D-2	eucTW	TW	88	zh_TW.eucTW	HP	Тайвань
950	D-2	big5	TW	88	big5	Sun	Тайвань
964	D-2	cns11643	TW	88	zh_TW	Sun	Тайвань
950	D-2	big5	TW	88	-	WIN	Тайвань
937	D-2	IBM-937	TW	88	-	HOST	Тайвань
874	S-20	TIS620-1	TH	66	-	OS2	Таиланд
874	S-20	TIS620-1	TH	66	Th_TH	AIX	Таиланд
874	S-20	tis620	TH	66	th_TH.tis620	HP	Таиланд
874	S-20	TIS620-1	TH	66	-	WIN	Таиланд
838	S-20	IBM-838	TH	66	-	HOST	Таиланд
857	S-9	IBM-857	TR	90	-	OS2	Турция
920	S-9	ISO8859-9	TR	90	tr_TR	AIX	Турция
920	S-9	iso88599	TR	90	tr_TR.iso88599	HP	Турция
920	S-9	ISO8859-9	TR	90	tr_TR.ISO8859-9	SCO	Турция
920	S-9	ISO-8859-9	TR	90	tr_TR	Linux	Турция
1254	S-9	1254	TR	90	-	WIN	Турция
1281	S-9	1281	TR	90	-	Mac	Турция
1026	S-9	IBM-1026	TR	90	-	HOST	Турция
437	S-1	IBM-437	GB	44	-	OS2	Великобритания
850	S-1	IBM-850	GB	44	-	OS2	Великобритания
819	S-1	ISO8859-1	GB	44	en_GB	AIX	Великобритания
850	S-1	IBM-850	GB	44	En_GB	AIX	Великобритания
819	S-1	iso88591	GB	44	en_GB.iso88591	HP	Великобритания
1051	S-1	roman8	GB	44	en_GB.roman8	HP	Великобритания
819	S-1	ISO8859-1	GB	44	en_UK	Sun	Великобритания
819	S-1	ISO8859-1	GB	44	en_GB	SCO	Великобритания
819	S-1	ISO8859-1	GB	44	en	SCO	Великобритания
819	S-1	ISO-8859-1	GB	44	en_GB	Linux	Великобритания
1252	S-1	1252	GB	44	-	WIN	Великобритания
1275	S-1	1275	GB	44	-	Mac	Великобритания
285	S-1	IBM-285	GB	44	-	HOST	Великобритания
1146	S-1	IBM-1146	GB	44	-	HOST	Великобритания
819	S-1	88591	GB	44	En_GB.88591	SINIX	Великобритания
819	S-1	ISO8859-1	GB	44	En_GB.6937	SINIX	Великобритания
1125	S-5	IBM-1125	UA	380	-	OS2	Украина
1124	S-5	IBM-1124	UA	380	uk_UA	AIX	Украина
1251	S-5	1251	UA	380	-	WIN	Украина
1123	S-5	IBM-1123	UA	380	-	HOST	Украина

Таблица 32. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	Терр	Код страны	Локаль	ОС	Название страны
437	S-1	IBM-437	US	1	-	OS2	США
850	S-1	IBM-850	US	1	-	OS2	США
819	S-1	ISO8859-1	US	1	en_US	AIX	США
850	S-1	IBM-850	US	1	En_US	AIX	США
819	S-1	iso88591	US	1	en_US.iso88591	HP	США
1051	S-1	roman8	US	1	en_US.roman8	HP	США
819	S-1	ISO8859-1	US	1	en_US	Sun	США
819	S-1	ISO8859-1	US	1	en_US	SGI	США
819	S-1	ISO8859-1	US	1	en_US	SCO	США
819	S-1	ISO-8859-1	US	1	en_US	Linux	США
1252	S-1	1252	US	1	-	WIN	США
1275	S-1	1275	US	1	-	Mac	США
37	S-1	IBM-37	US	1	-	HOST	США
1140	S-1	IBM-1140	US	1	-	HOST	США
1163	S-11	IBM-1163	VN	84	-	OS2	Вьетнам
1163	S-11	IBM-1163	VN	84	vi_VN	AIX	Вьетнам
1258	S-11	1258	VN	84	-	WIN	Вьетнам
1164	S-11	IBM-1164	VN	84	-	HOST	Вьетнам

```
Арабские страны (AA):
-----------------------------------------
    /* Арабский язык (Саудовская Аравия) */
    /* Арабский язык (Ирак) */
    /* Арабский язык (Египет) */
    /* Арабский язык (Ливия) */
    /* Арабский язык (Алжир) */
    /* Арабский язык (Марокко) */
    /* Арабский язык (Тунис) */
    /* Арабский язык (Оман) */
    /* Арабский язык (Йемен) */
    /* Арабский язык (Сирия) */
    /* Арабский язык (Иордания) */
    /* Арабский язык (Ливан) */
    /* Арабский язык (Кувейт) */
    /* Арабский язык (Объединенные Арабские Эмираты) */
    /* Арабский язык (Бахрейн) */
    /* Арабский язык (Катар) */
```

```
Английский язык (US):
---------------------------------
    /* Английский язык (Ямайка) */
    /* Английский язык (Страны Карибского бассейна) */
```

Таблица 32. Поддерживаемые языки и кодовые наборы (продолжение)

```
Кодовая                                  Код
страница  Группа  Кодовый набор Терр страны  Локаль  ОС    Название страны
--------  ------  ------------ ---- ------  ------ ----   ---------------
```

```
 Латинская Америка (Lat):
 ----------------------------------------
     /* Испанский язык (Мексика) */
     /* Испанский язык (Гватемала) */
     /* Испанский язык (Коста-Рика) */
     /* Испанский язык (Панама) */
     /* Испанский язык (Доминиканская Республика) */
     /* Испанский язык (Венесуэла) */
     /* Испанский язык (Колумбия) */
     /* Испанский язык (Перу) */
     /* Испанский язык (Аргентина) */
     /* Испанский язык (Эквадор) */
     /* Испанский язык (Чили) */
     /* Испанский язык (Уругвай) */
     /* Испанский язык (Парагвай) */
     /* Испанский язык (Боливия) */
```

Примечание: Кодовая страница 950 Solaris не поддерживает следующие символы из IBM 950:

Диапазон кодов	**Описание**	**Sun Big-5**	**IBM Big-5**
C6A1-C8FE	Символы	Зарезервированная область	Символы
F9D6-F9FE	Расширение ETen	Зарезервированная область	Расширение ETen
F286-F9A0	Выбранные IBM символы	Зарезервированная область	Выбранные IBM

Примечание: В этой версии DB2 UDB поддерживает символ евро. Кодовые страницы Microsoft Windows ANSI изменены в соответствии с последним определением Microsoft и включают символ евро в позиции 0x80. Раньше этот код был не определен. Кроме того, определение кодовой страницы 850 было изменено так, что символ ″i без точки″ (код 0xD5) был заменен на символ евро. DB2 UDB по умолчанию использует новые определения этих кодовых страниц для поддержки символа евро. Такая реализация является подходящей установкой по умолчанию как для тех пользователей DB2 UDB, которым требуется поддержка символа евро, так и для

тех, кому она не нужна. Если вы, тем не менее, хотите продолжить использовать старые определения этих кодовых страниц, скопируйте следующие файлы:

- 12520850.cnv
- 08501252.cnv
- IBM00850.ucs
- IBM01252.ucs

из каталога

```
sqllib/conv/alt/
```

в каталог

```
sqllib/conv/
```

после завершения установки. Перед заменой файлов `IBM01252.usc` и `IBM00850.ucs` на версии без символа евро можно сделать их резервные копии. После копирования этих файлов поддержки символа евро в DB2 UDB не будет.

Определение значений кодовых страниц

Кодовая страница прикладной программы определяется из активной среды при выполнении соединения с базой данных. Если задана переменная реестра DB2CODEPAGE, ее значение берется в качестве кодовой страницы прикладной программы. Тем не менее устанавливать переменную реестра DB2CODEPAGE нет необходимости, поскольку DB2 определит необходимое значение кодовой страницы из операционной системы. Задание для переменной реестра DB2CODEPAGE неправильного значения может привести к непредсказуемым результатам.

Кодовая страница базы данных определяется значением, указанным при создании этой базы данных (явно или по умолчанию). Ниже приводятся примеры того, как *активная среда* определяется в разных операционных системах:

UNIX — В операционных системах на основе UNIX активная среда определяется из значения локали (национальной версии), включающего в себя информацию о языке, территории и кодовом наборе.

OS/2 — В OS/2 первичная и вторичная кодовые страницы указываются в файле `CONFIG.SYS`. Во время текущего сеанса для вывода и динамической смены кодовых страниц можно использовать команду **chcp**.

Macintosh Для операционной системы Macintosh, если не задана переменная реестра DB2CODEPAGE, кодовая страница Macintosh определяется кодом региональной версии из установленного сценария.

32-битные операционные системы Windows
Во всех 32-битных операционных системах Windows, если не установлена переменная реестра DB2CODEPAGE, кодовая страница определяется по значению кодовой страницы ANSI в реестре.

Полный список кодовых страниц для разных сред смотрите в Табл. 32 на стр. 412.

Наборы символов

Обычно менеджер баз данных не ограничивает набор символов, доступный для прикладной программы. Подробное объяснение многобайтных наборов символов (multi-byte character sets - MBCS), поддерживаемых DB2, смотрите в книге *Application Development Guide*.

Набор символов для идентификаторов

Основной набор символов, который можно использовать в именах баз данных, состоит из однобайтных прописных и строчных латинских букв (A...Z, a...z), арабских цифр (0...9) и символа подчеркивания (_). К этому списку для обеспечения совместимости с программными продуктами баз данных хоста добавляется три специальных символа (#, @ и $) . Однако в среде NLS эти символы следует использовать с осторожностью, поскольку они не включены в инвариантный набор символов хоста NLS (EBCDIC).

При именовании объектов баз данных (таких, как таблицы и производные таблицы), меток программ, переменных хоста и указателей могут быть также использованы элементы из расширенного набора символов (например, русские буквы). Какие конкретно символы доступны, зависит от используемой кодовой страницы. Если вы используете базу данных в среде с несколькими кодовыми страницами, необходимо убедиться, что все кодовые страницы поддерживают все элементы из того расширенного набора символов, который вы планируете использовать. Информацию об идентификаторах с ограничителями, содержащих символы за пределами расширенного набора символов, которые можно использовать в операторах SQL, смотрите в справочнике *SQL Reference*.

Определение расширенного набора символов для идентификаторов DBCS

В среде DBCS расширенный набор символов состоит из всех символов основного набора плюс:

- Все двухбайтные символы всех кодовых страниц DBCS, кроме двухбайтного пробела, являются допустимыми буквами.
- Двухбайтный пробел является специальным символом.
- Однобайтные символы, доступные на каждой из страниц со смешанной кодировкой, распадаются на несколько категорий:

Категория
Действительные значения кодов на каждой из страниц со смешанной кодировкой

Цифры
x30-39

Буквы x23-24, x40-5A, x61-7A, xA6-DF (A6-DF только для кодовых страниц 932 и 942)

Специальные символы
Все другие действительные значения кодов однобайтных символов

Кодирование операторов SQL

Кодирование операторов SQL не зависит от языка. Ключевые слова SQL можно вводить как в верхнем регистре, так и в нижнем регистре, а также в смешанном регистре. Имена объектов баз данных и переменных хоста, а также ярлыки программ в операторе SQL не могут содержать символы, выходящие за пределы расширенного набора символов, как описано выше.

Поддержка CCSID с двумя направлениями письма

Для правильной обработки данных с двумя направлениями письма на разных платформах требуются следующие атрибуты направления письма:

```
- Тип текста (LOGICAL или VISUAL)
- Форма (SHAPED или UNSHAPED)
- Ориентация (RIGHT-TO-LEFT или LEFT-TO-RIGHT)
- Форма цифр (ARABIC или HINDI)
- Симметричное обращение (YES или NO)
```

Поскольку на разных платформах по умолчанию устанавливаются разные значения, при переносе данных DB2 с одной платформы на другую могут происходить ошибки. Например, операционная система Windows использует данные LOGICAL UNSHAPED, а OS/390 обычно использует данные SHAPED VISUAL. Поэтому без поддержки для двунаправленных атрибутов данные, посланные из DB2 Universal Database for OS/390 в DB2 UDB 32-битной системы Windows, могут отображаться неправильно.

CCSID с двумя направлениями письма

DB2 поддерживает атрибуты данных с двумя направлениями письма через специальные идентификаторы наборов кодовых символов с двумя направлениями письма (Coded Character Set Identifiers - CCSID). Для DB2 UDB определены и используются следующие CCSID с двумя направлениями письма.

CCSID (дес)	CCSID (шестн)	Кодовая страница	Тип строки
00420	x'01A4'	420	4
00424	x'01A8'	424	4
08612	x'21A4'	420	5
08616	x'21A8'	424	10
00856	x'0358'	856	5
00862	x'035E'	862	4
00864	x'0360'	864	5
00916	x'0394'	916	5
01046	x'0416'	1046	5
01089	x'0441'	1089	5
01255	x'04E7'	1255	5
01256	x'04E8'	1256	5
62208	x'F300'	856	4
62209	x'F301'	862	10
62210	x'F302'	916	4
62211	x'F303'	424	5
62213	x'F305'	862	5
62215	x'F307'	1255	4
62218	x'F30A'	864	4
62220	x'F30C'	856	6
62221	x'F30D'	862	6
62222	x'F30E'	916	6
62223	x'F30F'	1255	6
62224	x'F310'	420	6
62225	x'F311'	864	6
62226	x'F312'	1046	6
62227	x'F313'	1089	6
62228	x'F314'	1256	6
62229	x'F315'	424	8
62230	x'F316'	856	8
62231	x'F317'	862	8
62232	x'F318'	916	8
62233	x'F319'	420	8
62234	x'F31A'	420	9
62235	x'F31B'	424	6
62236	x'F31C'	856	10
62237	x'F31D'	1255	8
62238	x'F31E'	916	10
62239	x'F31F'	1255	10
62240	x'F320'	424	11
62241	x'F321'	856	11
62242	x'F322'	862	11
62243	x'F323'	916	11
62244	x'F324'	1255	11
62245	x'F325'	424	10
62246	x'F326'	1046	8

```
62247   x'F327'  1046      9
62248   x'F328'  1046      4
62249   x'F329'  1046     12
62250   x'F32A'   420     12
```

где типы строк CDRA определены так:

Тип строки	Тип текста	Форма цифр	Ориентация	Форма	Симметричное обращение
4	Visual	Passthru	LTR	Shaped	OFF
5	Implicit	Arabic	LTR	Unshaped	ON
6	Implicit	Arabic	RTL	Unshaped	ON
7(*)	Visual	Arabic	Contextual(*)	Unshaped-Lig	OFF
8	Visual	Arabic	RTL	Shaped	OFF
9	Visual	Passthru	RTL	Shaped	ON
10	Implicit	Passthru	Contextual-L	Unshaped	ON
11	Implicit	Passthru	Contextual-R	Unshaped	ON
12	Implicit	Arabic	RTL	Shaped	ON

Примечание: (*) Ориентация поля - слева направо (LTR), если первый алфавитный символ - латинский, и справа налево (RTL), если первый алфавитный символ - не латинский. Форма символов не меняется, но лигатуры типа лам-алеф сохраняются, а не разбиваются на составляющие.

Поддержка двух направлений письма в DB2 Universal Database

Размещение текста с двумя направлениями письма реализуется в DB2 Universal Database при помощи новых определений CCSID. Для этих новых CCSID с двумя направлениями письма вместо преобразований кодовых страниц или дополнительно к ним производится преобразование размещения. Чтобы использовать эту поддержку, для переменной регистра DB2BIDI должно быть задано значение YES. По умолчанию эта переменная не задается. Она используется сервером для всех преобразований и может быть установлена только при запуске сервера. Задание значения YES для DB2BIDI может оказать влияние на производительность из-за дополнительной проверки и преобразований размещения.

Чтобы указать конкретный CCSID с двумя направлениями письма в среде (кроме DRDA), выберите в приведенной выше таблице CCSID, соответствующий характеристикам вашего клиента, и задайте это значение для DB2CODEPAGE. Если у вас уже установлено соединение с базой данных, необходимо ввести команду TERMINATE, а затем снова установить соединение, чтобы новое значение DB2CODEPAGE вступило в силу. Если выбрать CCSID, который не подходит для кодовой страницы или типа строки вашей платформы клиента, можно получить непредсказуемые результаты. Если выбран несовместимый CCSID (например, CCSID иврита для связи с базой данных на арабском языке) или если DB2BIDI не был установлен для этого сервера, при попытке соединения вы получите сообщение об ошибке.

В средах DRDA, если платформа хоста с кодом EBCDIC также поддерживает двунаправленные CCSID, необходимо только задать значение DB2CODEPAGE. Однако, если платформа хоста не поддерживает эти CCSID, необходимо также указать новый CCSID для сервера баз данных хоста, с которым вы связываетесь. Это необходимо потому, что в среде DRDA преобразования кодовых страниц и размещения выполняются получателем данных. Однако, если сервер хоста не поддерживает эти CCSID с двумя направлениями письма, он не выполняет преобразование размещения для данных, которые получает от DB2 UDB. Если вы используете переопределение CCSID, клиент DB2 UDB выполняет также преобразование размещения для выходных данных. Информацию о переопределении CCSID смотрите в книге *DB2 Connect. Руководство пользователя*.

Переопределение CCSID не поддерживается для тех случаев, когда платформа хоста с кодом EBCDIC является клиентом, а DB2 UDB - сервером.

Поддержка двух направлений письма в DB2 Connect

При обмене данными между DB2 Connect и базой данных на сервере преобразование входящих данных обычно выполняет получатель. Такое же соглашение будет обычно действовать и для преобразования размещения в дополнение к обычному преобразованию кодовых страниц. У DB2 Connect есть дополнительная возможность выполнять преобразования размещения и для данных, подготовленных к отправке в базу данных сервера, в дополнение к данным, получаемым от базы данных сервера.

Чтобы DB2 Connect выполняла преобразования размещения для исходящих данных, отправляемых на базу данных сервера, CCSID базы сервера надо переопределить. Это достигается с помощью параметра BIDI в поле PARMS записи каталога базы данных DCS для базы данных сервера.

Примечание: Если вы хотите, чтобы DB2 Connect выполняла преобразования размещения для данных, подготовленных к отправке на базу данных хоста DB2, даже в том случае, если вы не переопределили ее CCSID, вам все равно необходимо добавить параметр BIDI в поле PARMS каталога базы данных DCS. В этом случае надо задать CCSID, который используется по умолчанию базой данных хоста DB2.

Параметр BIDI должен быть указан девятым параметром в поле PARMS вместе с CCSID, на который вы хотите заменить устанавливаемый по умолчанию CCSID базы данных сервера:

```
",,,,,,,,BIDI=xyz"
```

где *xyz* - новый CCSID.

Примечание: Чтобы параметр BIDI вступил в действие, для переменной регистра DB2BIDI должно быть задано значение YES.

Список поддерживаемых CCSID с двумя направлениями письма, а также их типы строк можно найти в разделе “CCSID с двумя направлениями письма” на стр. 428.

Использование этой функции лучше всего описать на примере.

Предположим, что у вас есть клиент DB2 с ивритом, на котором работает CCSID 62213 (тип строки 5), и вы хотите обратиться к базе данных хоста DB2, на которой запущен CCSID 00424 (тип строки 4). Однако вы знаете, что для данных, содержащихся в базе данных хоста DB2, используется CCSID 08616 (тип строки 6).

Здесь мы сталкиваемся с двумя проблемами: Во-первых, база данных хоста DB2 не знает, чем различаются типы двунаправленных строк с CCSID 00424 и 08616. Во-вторых, база данных хоста DB2 не распознает CCSID клиента DB2 (62213). Она поддерживает только CCSID 00862, основанный на той же кодовой странице, что и CCSID 62213.

Вам надо добиться, чтобы данные, посланные на базу данных хоста DB2, начинались с формата строки типа 6, а также сообщить DB2 Connect, что она должна выполнять преобразование размещения для данных, которые она получает от базы данных хоста DB2. Надо использовать следующую команду catalog для базы данных хоста DB2:

```
db2 catalog dcs database nydb1 as telaviv parms ",,,,,,,,BIDI=08616"
```

Эта команда заставляет DB2 Connect использовать CCSID базы данных хоста DB2 08616 вместо 00424. Такая замена включает в себя следующие действия:

1. DB2 Connect связывается с базой данных хоста DB2 с использованием CCSID 00862.
2. DB2 Connect выполняет преобразование размещения для данных, которые она собирается *послать* базе данных хоста DB2. Производится преобразование из CCSID 62213 (тип строки 5) в CCSID 62221 (тип строки 6).
3. DB2 Connect выполняет преобразование размещения для данных, которые она *получает* от базы данных хоста DB2. Эта трансформация производится из CCSID 08616 (тип строки 6) в CCSID 62213 (тип строки 5).

Примечание: В некоторых случаях использование CCSID с двумя направлениями письма может привести к тому, что сам запрос SQL будет модифицирован и сервер DB2 его не распознает. Чтобы этого не случилось, следует исключить использование CCSID с IMPLICIT CONTEXTUAL и IMPLICIT RIGHT-TO-LEFT, когда могут быть использованы разные типы строк. CCSID с

CONTEXTUAL может давать непредсказуемые результаты, если в запросе SQL содержатся строки в двойных кавычках. Избегайте использования строк в двойных кавычках в операторах SQL; где это возможно, используйте переменные хоста.

Если использование конкретного CCSID с двумя направлениями письма вызывает ошибки, которые не удается устранить посредством приведенных рекомендаций, установите NO для DB2BIDI.

Последовательность сортировки

Менеджер баз данных сравнивает символьные данные при помощи *последовательности сортировки*. Это упорядочение для набора символов, которое определяет, будет ли при сортировке конкретный символ расположен до другого, после другого или там же, где другой. Например, последовательность сортировки может быть использована для того, чтобы указать, что прописные и строчные версии конкретного символа должны сортироваться одинаково.

Последовательность сортировки указывается во время создания базы данных и не может быть изменена после этого.

Менеджер баз данных позволяет создавать базы данных с произвольной последовательностью сортировки, используя интерфейс прикладного программирования (API). Информацию о создании пользовательской таблицы последовательности сортировки смотрите в книге *Application Development Guide*.

Примечание: Данные строк символов, определенные с атрибутом FOR BIT DATA, и данные двоичных больших объектов сортируются при помощи двоичной последовательности сортировки.

Общие соображения

После определения последовательности сортировки все последующие сравнения символов для этой базы данных будут выполняться с ее помощью. За исключением символьных данных, определенных как FOR BIT DATA или данные типа двоичный большой объект, последовательность сортировки будет использоваться для всех сравнений SQL и условий ORDER BY, а также для построения индексов и статистики. Дополнительную информацию об использовании последовательности сортировки базы данных смотрите в разделе ″String Comparisons″ в справочнике *SQL Reference*.

Ошибки могут происходить в следующих случаях:

- Прикладная программа производит слияние сортированных данных из базы данных с данными прикладной программы, сортированными с использованием другой последовательности сортировки.

- Прикладная программа производит слияние сортированных данных из одной базы данных с сортированными данными другой базы данных, но эти базы данных имеют разные последовательности сортировки.
- Прикладная программа делает предположения относительно сортированных данных, которые не верны для определенной последовательности сортировки. Например, предположение о расположении цифр после букв может быть верным для конкретной последовательности сортировки, а может и не быть.

Наконец, следует помнить, что результаты любой сортировки, основанной на прямом сравнении значений кода символов, будут совпадать только с результатами запроса, упорядоченными с использованием идентичной последовательности сортировки.

Особенности баз данных объединения

Ваш выбор последовательности сортировки базы данных может влиять на производительность системы объединения. Если источник данных использует ту же самую последовательность сортировки, что и база данных объединения DB2, то DB2 может передать на источник данных обработку, включающую символьные данные. Если последовательность сортировки источника данных не совпадает с последовательностью сортировки DB2, данные считываются DB2 и вся обработка с изменением уровня в отношении символьных данных выполняется локально (это может снизить производительность).

При определении, совпадают ли последовательности сортировки источника данных и DB2, учитывайте следующее:

- Поддержку национальных языков.

  Последовательность сортировки зависит от языка, поддерживаемого на сервере. Сравните информацию о NLS DB2 и NLS источника данных.
- Характеристики источника данных.

  Некоторые источники данных созданы с использованием регистронезависимых последовательностей сортировки, что в операциях, зависящих от порядка, может приводить к результатам, отличным от результатов DB2.
- Пользовательские настройки.

  Некоторые источники данных предоставляют для последовательностей сортировки различные опции или допускают пользовательскую настройку последовательности сортировки.

Выбирайте последовательность сортировки для базы данных объединения DB2 на основе совокупности источников данных, к которым с этой базы данных будет производиться доступ. Например:

- Если база данных DB2 будет обращаться в основном к базам данных Oracle с той же самой кодовой страницей (NLS), что и DB2, при создании базы данных

укажите идентичную последовательность (базы данных Oracle используют эквивалентную последовательность сортировки).

- Если база данных DB2 будет обращаться только к базам данных DB2 UDB, задайте совпадающие значения последовательностей сортировки.

Информацию о задании последовательности сортировки MVS смотрите в руководстве *Administrative API Reference* (примеры после описания **sqlecrea** - Create Database API). В этих примерах содержатся таблицы последовательностей сортировки для кодовых страниц EBCIDIC 500, 37 и 5026/5035.

После установки последовательности сортировки для базы данных DB2 убедитесь, что опция сервера *collating_sequence* установлена для всех серверов источников данных. Эта опция указывает, совпадает ли последовательность сортировки данного сервера источника данных с последовательностью сортировки базы данных DB2.

Задайте ″Y″ для опции collating_sequence, если последовательности сортировки совпадают. Такая установка позволит оптимизатору DB2 рассматривать вариант обработки с зависимостью от порядка на источнике данных, что может повысить производительность. Однако, если последовательность сортировки источника данных не совпадает с последовательностью сортировки базы данных DB2, вы можете получить неверные результаты. Например, если ваш план использует объединение слияния, оптимизатор DB2 передаст операции упорядочения в максимально возможном объеме на источники данных. Если у источника данных другая последовательность сортировки, результаты объединения могут быть неверными.

Если последовательности сортировки не совпадают, задайте ″N″ для опции collating_sequence. Используйте это значение, когда последовательности сортировки источников данных и DB2 разные или когда операции сортировки источников данных могут быть регистронезависимыми. Например, на регистронезависимом источнике данных с кодовой страницей английского языка TOLLESON, ToLLeSoN и tolleson будут рассматриваться как равные. Если вы не уверены, что последовательность сортировки источника данных идентична последовательности сортировки DB2, задайте ″N″ для опции collating_sequence.

Значения даты и времени

Ниже описываются типы данных даты и времени. Несмотря на то, что значения даты и времени могут быть использованы в определенных арифметических и строчных операциях и совместимы с определенными строками, они не являются ни строками, ни числами.

Дата

Дата представляет собой значение из трех частей (год, месяц и день). Диапазон части года от 0001 до 9999. Диапазон части месяца от 1 до 12. Диапазон части дня от 1 до *x*, где *x* зависит от месяца.

Внутреннее представление даты - символьная строка длиной 4 байта. Каждый байт состоит из 2 упакованных десятичных цифр. Первые 2 байта представляют год, третий байт - месяц, а последний - день.

Длина столбца DATE, как описано в SQLDA - 10 байт, что позволяет представить это значение символьной строкой.

Время

Время - значение из трех частей (часы, минуты и секунды), обозначающее время суток в 24-часовом формате. Диапазон части часов от 0 до 24. Диапазон других частей - от 0 до 59. Если значение часов 24, для минут и секунд устанавливается 0.

Внутреннее представление времени - символьная строка длиной 3 байта. Каждый байт состоит из 2 упакованных десятичных цифр. Первый байт представляет часы, второй байт - минуты, а последний - секунды.

Длина столбца TIME, как описано в SQLDA - 8 байт, что позволяет представить это значение символьной строкой.

Отметка времени

Отметка времени - значение из семи частей (год, месяц, день, часы, минуты, секунды и микросекунды), обозначающее дату и время суток, как описано выше, за тем исключением, что во время включаются микросекунды.

Внутреннее представление отметки времени - символьная строка длиной 10 байт. Каждый байт состоит из 2 упакованных десятичных цифр. Первые 4 байта представляют дату, следующие 3 байта - время и последние 3 байта - микросекунды.

Длина столбца TIMESTAMP, как описано в SQLDA - 26 байт, что позволяет представить это значение символьной строкой.

Строчные представления значений даты и времени

Значения с типом данных DATE, TIME или TIMESTAMP представляются во внутренней форме, прозрачной для пользователя SQL. Даты, время и отметки времени, тем не менее, можно также представить символьными строками, и такие представления непосредственно затрагивают пользователя SQL, поскольку они не являются ни константами, ни переменными с типами данных DATE, TIME или TIMESTAMP. Поэтому для использования значение даты и времени должно быть назначено переменной символьной строки. Представление символьной строки обычно является используемым по умолчанию форматом значений даты и времени, ассоциированным с кодом страны клиента, если только переопределяется заданием опции формата ″F″ при прекомпиляции программы или ее связывании с базой данных. Список форматов символьных строк для кодов различных стран смотрите в Табл. 35 на стр. 439.

Когда действительное строчное представление значения даты и времени используется в операции с внутренним значением даты и времени, это строчное представление перед выполнением такой операции преобразуется во внутреннюю форму даты, времени или отметки времени. Действительные строчные представления значений даты и времени определены в следующих разделах.

Строки даты

Строчное представление даты представляет собой символьную строку, начинающуюся с цифры, длиной не менее 8 символов. Могут быть включены хвостовые пробелы; в частях месяца и дня даты могут отсутствовать начальные нули.

Действительные строчные форматы для дат перечислены в Табл. 33. Каждый формат идентифицируется названием и включает связанное с ним сокращение, а также пример использования.

Таблица 33. Форматы для строчных представлений даты

Название формата	Сокращение	Формат даты	Пример
ISO	ISO	гггг-мм-дд	1991-10-27
Стандарт IBM для США	USA	мм/дд/гггг	10/27/1991
Стандарт IBM для Европы	EUR	дд.мм.гггг	27.10.1991
Японский промышленный стандарт, христианское летоисчисление	JIS	гггг-мм-дд	1991-10-27
Локальный	LOC	Зависит от кода страны базы данных	–

Строки времени

Строчное представление времени представляет собой символьную строку, начинающуюся с цифры, длиной не менее 4 символов. Могут быть включены хвостовые пробелы; в части часов может отсутствовать начальный нуль, а секунды могут отсутствовать полностью. Если секунды не указаны, неявно подразумевается 0 секунд. Это значит, что `13.30` эквивалентно `13.30.00`.

Действительные строчные форматы для времени перечислены в Табл. 34. Каждый формат идентифицируется названием и включает связанное с ним сокращение, а также пример использования.

Таблица 34. Форматы для строчных представлений времени

Название формата	Сокращение	Формат времени	Пример
ISO	ISO	чч.мм.сс	13.30.05

Таблица 34. Форматы для строчных представлений времени (продолжение)

Название формата	Сокращение	Формат времени	Пример
Стандарт IBM для США	USA	чч:мм AM или PM	1:30 PM
Стандарт IBM для Европы	EUR	чч.мм.сс	13.30.05
Японский промышленный стандарт, христианское летоисчисление	JIS	чч:мм:сс	13:30:05
Локальный	LOC	Зависит от кода страны прикладной программы	–

Примечания:

1. В форматах строки времени ISO, EUR и JIS `.cc` (или `:cc`) - необязательно.
2. В формате строки времени USA минуты могут отсутствовать, что неявно подразумевает 00 минут. Это значит, что 1 PM эквивалентно 1:00 PM.
3. В формате строки времени USA спецификация часов не может быть больше 12 и не может равняться 0, за исключением специального случая 00:00 AM. При использовании 24-часового формата ISO соответствие с форматом USA следующее:
   - 12:01 AM - 12:59 AM соответствует 00.01.00 - 00.59.00.
   - 01:00 AM - 11:59 AM соответствует 01.00.00 - 11.59.00.
   - 12:00 PM (полдень) - 11:59 PM соответствует 12.00.00 - 23.59.00.
   - 12:00 AM (полночь) соответствует 24.00.00, и 00:00 AM (полночь) соответствует 00.00.00.

Строки отметки времени

Строчное представление отметки времени представляет собой символьную строку, начинающуюся с цифры, длиной не менее 16 символов. Полное представление строки отметки времени имеет вид *гггг-мм-дд-чч.мм.сс.нннннн*. Могут быть включены хвостовые пробелы; в частях месяца, дня и часов отметки времени могут отсутствовать начальные нули, а микросекунды могут быть сокращены или отсутствовать полностью. Если вы выбираете отсутствие любой цифры в части микросекунд, неявно подразумевается 0. Это значит, что `1991-3-2-8.30.00` эквивалентно `1991-03-02-08.30.00.000000`.

MBCS в отметках даты и времени

Строки отметки даты и времени должны содержать только однобайтные символы и цифры.

Форматы даты и времени: Формат символьной строки, представляющей дату и время - это формат по умолчанию для значений даты и времени, связанный с

кодом страны для программы. Этот формат по умолчанию может быть переопределен, если при прекомпиляции программы или ее связывании с базой данных задана опция формата ″F″ .

Ниже приводится описание входных и выходных форматов для даты и времени:

- Входной формат времени
  - Для времени не существует входного формата по умолчанию
  - Любой формат допустим на вводе для любого кода страны.
- Выходной формат времени
  - Выходной формат времени по умолчанию совпадает с местным форматом времени.
- Входной формат даты
  - Для даты не существует входного формата по умолчанию
  - Там, где локальный формат даты несовместим с форматами дат ISO, JIS, EUR или USA, при вводе распознается местный формат даты. (Например, посмотрите запись в Табл. 35 для Великобритании.)
- Выходной формат даты
  - Выходные форматы дат по умолчанию показаны в Табл. 35.

Примечание: В Табл. 35 приводится также список форматов строк для различных кодов стран.

Таблица 35. Форматы даты и времени для кодов стран

Код языка	Местный формат даты	Местный формат времени	Выходной формат даты по умолчанию	Входные форматы даты
355 Албания	гггг-мм-дд	JIS	LOC	LOC, USA, EUR, ISO
785 Арабский	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
001 Австралия (1)	мм-дд-гггг	JIS	LOC	LOC, USA, EUR, ISO
061 Австралия	дд-мм-гггг	JIS	LOC	LOC, USA, EUR, ISO
032 Бельгия	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
055 Бразилия	дд.мм.гггг	JIS	LOC	LOC, EUR, ISO
359 Болгария	дд.мм.гггг	JIS	EUR	LOC, USA, EUR, ISO
001 Канада	мм-дд-гггг	JIS	USA	LOC, USA, EUR, ISO
002 Канада (французский)	дд-мм-гггг	ISO	ISO	LOC, USA, EUR, ISO

Таблица 35. Форматы даты и времени для кодов стран (продолжение)

Код языка	Местный формат даты	Местный формат времени	Выходной формат даты по умолчанию	Входные форматы даты
385 Хорватия	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
042 Чехия	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
045 Дания	дд-мм-гггг	ISO	ISO	LOC, USA, EUR, ISO
358 Финляндия	дд/мм/гггг	ISO	EUR	LOC, EUR, ISO
389 Македония	дд.мм.гггг	JIS	EUR	LOC, USA, EUR, ISO
033 Франция	дд/мм/гггг	JIS	EUR	LOC, EUR, ISO
049 Германия	дд/мм/гггг	ISO	ISO	LOC, EUR, ISO
030 Греция	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
036 Венгрия	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
354 Исландия	дд-мм-гггг	JIS	LOC	LOC, USA, EUR, ISO
091 Индия	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
972 Израиль	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
039 Италия	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
081 Япония	мм/дд/гггг	JIS	ISO	LOC, USA, EUR, ISO
082 Корея	мм/дд/гггг	JIS	ISO	LOC, USA, EUR, ISO
001 Латинская Америка (1)	мм-дд-гггг	JIS	LOC	LOC, USA, EUR, ISO
003 Латинская Америка	дд-мм-гггг	JIS	LOC	LOC, EUR, ISO
031 Голландия	дд-мм-гггг	JIS	LOC	LOC, USA, EUR, ISO
047 Норвегия	дд/мм/гггг	ISO	EUR	LOC, EUR, ISO
048 Польша	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
351 Португалия	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
086 КНР	мм/дд/гггг	JIS	ISO	LOC, USA, EUR, ISO

Таблица 35. Форматы даты и времени для кодов стран (продолжение)

Код языка	Местный формат даты	Местный формат времени	Выходной формат даты по умолчанию	Входные форматы даты
040 Румыния	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
007 Россия	дд/мм/гггг	ISO	LOC	LOC, EUR, ISO
381 Югославия	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
042 Словакия	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
386 Словения	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
034 Испания	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
046 Швеция	дд/мм/гггг	ISO	ISO	LOC, EUR, ISO
041 Швейцария	дд/мм/гггг	ISO	EUR	LOC, EUR, ISO
088 Тайвань	мм-дд-гггг	JIS	ISO	LOC, USA, EUR, ISO
066 Таиланд (2)	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
090 Турция	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
044 Великобритания	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
001 США	мм-дд-гггг	JIS	USA	LOC, USA, EUR, ISO
084 Вьетнам	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO

Примечания:

1. Для стран, использующих национальную версию C по умолчанию, используется код страны 001.
2. гггг в буддийском летоисчислении эквивалентно григорианскому + 543 года (только Таиланд).

Поддержка Unicode/UCS-2 и UTF-8 в DB2 UDB

Ниже описана поддержка этих двух стандартов.

Введение

Стандарт кодировки символов Unicode - схема кодировки символов с фиксированной длиной, включающая символы практически изо всех живых языков мира. Символы Unicode обычно записываются в виде ″U+*xxxx*″, где *xxxx* - шестнадцатеричный код символа.

Каждый символ, независимо от языка, имеет длину 16 бит (2 байта). Несмотря на то, что получающихся 65000 элементов кода достаточно для кодирования большинства символов основных мировых языков, стандарт Unicode также предоставляет механизм расширения, позволяющий кодирование еще около одного миллиона символов. Это расширение резервирует диапазон значений кода (U+D800 - U+D8FF, называемый ″суррогаты″) для кодирования 32-битных символов как двух последовательных элементов кода.

Стандарт Международной организации по стандартизации (ISO) и Международной электротехнической комиссии (IEC) 10646 (ISO/IEC 10646) описывает Универсальный многооктетный набор кодовых символов (Universal Multiple-Octet Coded Character Set, UCS) с 2-байтной версией (UCS-2) и 4-байтной версией (UCS-4). 2-байтная версия этого стандарта ISO идентична Unicode без суррогатов. ISO 10646 определяет также технику расширения для кодирования некоторых кодов UCS-4 в кодовой строке UCS-2. Это расширение, называющееся UTF-16, идентично Unicode с суррогатами.

DB2 UDB поддерживает UCS-2, то есть Unicode без суррогатов.

Соединение клиента UTF-8 (кодовая страница 1208) с базой данных не-Unicode не поддерживается.

UTF-8

При кодировке UCS-2 или Unicode ASCII и управляющие символы также имеют длину два байта, причем первый байт является нулем. Например, пустое значение (NULL) представляется как U+0000, а прописная латинская ″A″ - как U+0041. Это может создавать большие проблемы в основанных на ASCII прикладных программах и файловых системах ASCII, поскольку в строке UCS-2 дополнительные NULL могут появляться в любом месте. Эту проблему для программ, предполагающих использование кода ASCII, можно обойти при помощи алгоритма преобразования UTF-8.

UTF-8 (формат преобразования UCS 8, UCS Transformation Format 8) - алгоритмическое преобразование, переводящее символы UCS-4 фиксированной длины в байтовые строки переменной длины. В UTF-8 символы ASCII представлены своими обычными однобайтными кодами, а остальные символы UCS-2 получают длину два или три байта. Другими словами, UTF-8 трансформирует символы UCS-2 в многобайтный кодовый набор, не меняя символы ASCII. Число байтов для каждого символа UCS-2 в формате UTF-8 можно определить из следующей таблицы:

UCS-2 (шестнадцатеричный)	UTF-8 (двоичный)	Описание
От 0000 до 007F	0xxxxxxx	ASCII
От 0080 до 07FF	110xxxxx 10xxxxxx	до U+07FF
От 0800 до FFFF	1110xxxx 10xxxxxx 10xxxxxx	другие UCS-2

```
ПРИМЕЧАНИЕ: диапазон от D800 до DFFF будет исключен из обработки третьей строкой
этой таблицы, управляющей диапазоном UCS-4 от 0000 0800 до 0000 FFFF.
```

Последовательность *x* в таблице - битовое представление UCS символа. Например, U0080 трансформируется в 11000010 10000000.

Реализация UCS-2/UTF-8 в DB2 UDB

Номера кодовых страниц/CCSID

В программных продуктах фирмы IBM кодовая страница UCS-2 зарегистрирована как кодовая страница 1200. Все кодовые страницы определены с расширяемыми наборами символов, то есть когда к кодовой странице добавляются новые символы, номер этой кодовой страницы не изменяется. Кодовая страница 1200 всегда относится к текущей версии Unicode/UCS-2 и была использована для поддержки UCS-2 в DB2 UDB.

Конкретная версия стандарта UCS, как она описана Unicode 2.0 и ISO/IEC 10646-1, также зарегистрирована в IBM как CCSID 13488. Этот CCSID используется внутри DB2 UDB для хранения данных графических строк в базах данных euc-Japan и euc-Taiwan. И CCSID 13488, и кодовая страница 1200 относятся к UCS-2 и обрабатываются одинаково, за исключением значения двухбайтного (DBCS) пробела:

CP/CCSID	Однобайтный (SBCS) пробел	Двухбайтный (DBCS) пробел
1200	Отсутствует	U+0020
13488	Отсутствует	U+3000

```
ПРИМЕЧАНИЕ: в базе данных UCS-2 у U+3000 нет специального значения.
```

Что касается таблиц преобразования, поскольку кодовая страница 1200 является надмножеством CCSID 13488, для обеих используются одни и те же таблицы (надмножества).

В продуктах фирмы IBM UTF-8 зарегистрирован как CCSID 1208 с расширяемым набором символов (иногда также называемым кодовой страницей 1208). При добавлении к стандарту новых символов этот номер (1208) не изменяется. Этот номер 1208 используется в качестве номера многобайтной кодовой страницы для поддержки UCS-2/UTF-8 в DB2.

DB2 UDB поддерживает UCS-2 как новую многобайтную кодовую страницу. Номер кодовой страницы MBCS равен 1208, что является номером кодовой страницы базы данных и кодовой страницей данных символьных строк в базе данных. Номер двухбайтной кодовой страницы для UCS-2 - 1200, что является кодовой страницей данных графических строк внутри базы данных. Когда база данных создается в UCS-2/UTF-8, данные CHAR, VARCHAR, LONG VARCHAR и символьных больших объектов сохраняются в UTF-8, а данные GRAPHIC,

VARGRAPHIC, LONG VARGRAPHIC и двухбайтных символьных больших объектов - в UCS-2. Для простоты будем называть это базой данных UCS-2.

Создание базы данных UCS-2

По умолчанию базы данных создаются в кодовой странице создавшей их прикладной программы. Поэтому, если вы создаете базу данных из клиента UTF-8 (например, из положения UNIVERSAL в AIX) или если на клиенте для переменной реестра DB2CODEPAGE задано значение 1208, база данных будет создана как база данных UCS-2. В качестве альтернативы можно явным образом указать ″UTF-8″ для CODESET и использовать любые две действительные буквы кода TERRITORY, поддерживаемого DB2 UDB.

Например, чтобы создать базу данных UCS-2 с кодом территории для США, введите:

```
DB2 CREATE DATABASE dbname USING CODESET UTF-8 TERRITORY US
```

Чтобы создать базу данных UCS-2 с использованием API **sqlecrea**, необходимо соответствующим образом установить значения в *sqledbcountryinfo*. Например, установите для SQLDBCODESET UTF-8, а для SQLDBLOCALE - любой действительный код территории (например, US).

По умолчанию для базы данных UCS-2 используется последовательность сортировки IDENTITY, использующая порядок кодов UCS-2. Поэтому по умолчанию все символы UCS-2/UTF-8 упорядочиваются и сравниваются в соответствии с последовательностью их кодов в UCS-2.

Все зависящие от территории параметры, такие как формат даты и времени, разделитель десятичных разрядов и прочие, определяются по текущему значению территории клиента.

База данных UCS-2 позволяет выполнять соединения из любых однобайтных и многобайтных кодовых страниц, поддерживаемых DB2 UDB. Преобразования символов кодовой страницы между кодовой страницей клиента и UTF-8 автоматически выполняются менеджером баз данных. Данные с типами графических строк всегда находятся в UCS-2 и не подвергаются преобразованиям кодовых страниц. Исключением является среда процессора командной строки (CLP). Если в CLP выбрать данные графической строки (UCS-2), возвращенные данные графической строки преобразуются (посредством CLP) из UCS-2 в кодовую страницу среды вашего клиента.

Все клиенты ограничены наборами символов, методами ввода и шрифтами, поддерживаемыми в их средах, однако сама база данных UCS-2 принимает и сохраняет все символы UCS-2. Следовательно, каждый клиент обычно работает с подмножеством символов UCS-2, но менеджер баз данных позволяет использовать все символы UCS-2.

При преобразовании символов из локальной кодовой страницы в UTF-8 может произойти увеличение числа байтов. Для символов ASCII такого увеличения не происходит, но другие символы UCS-2 увеличиваются в два или три раза. Число байтов для каждого символа UCS-2 в формате UTF-8 можно определить из таблицы в разделе “UTF-8” на стр. 442.

Типы данных

Все типы данных, поддерживаемые DB2 UDB, поддерживаются и в базе данных UCS-2. В частности, данные графической строки поддерживаются для базы данных UCS-2 и хранятся в UCS-2/Unicode. Все клиенты, включая клиенты SBCS, могут работать с типами данных графических строк в UCS-2/Unicode, когда они связаны с базой данных UCS-2.

База данных UCS-2 подобна любой базе данных MBCS, в которой строчные данные измеряются в числе байтов. При работе с данными символьных строк в UTF-8 не следует полагать, что каждый символ равен одному байту. При многобайтном кодировании UTF-8 каждый символ ASCII занимает один байт, а остальные символы - два или три байта. Это следует учитывать при определении полей CHAR. В зависимости от соотношения символов ASCII и не-ASCII поле CHAR размером *n* байтов может содержать от *n*/3 до *n* символов.

Использование кодирования UTF-8 символьной строки по сравнению с типом данных UCS-2 графической строки также оказывает влияние на общие требования к хранению. Когда большая часть символов является символами ASCII, а между ними расположено несколько прочих символов, лучшей альтернативой может быть сохранение данных в UTF-8, поскольку требования к хранению близки к одному байту на символ. С другой стороны, в ситуации, когда большинство символов не являются символами ASCII и расширяются до трехбайтных последовательностей UTF-8, лучшей альтернативой может быть формат графической строки UCS-2, поскольку каждому символу UCS-2 требуется точно два байта, а не три, как для соответствующего символа в формате UTF-8.

В средах MBCS скалярные функции SQL, работающие с символьными строками, такие как LENGTH, SUBSTR, POSSTR, MAX, MIN и подобные, оперируют с числом ″байтов″, а не с числом ″символов″. В базе данных UCS-2 поведение такое же, но необходимо соблюдать дополнительные предосторожности при указании сдвигов и длин для базы данных USC-2, поскольку эти значения всегда определяются в контексте кодовой страницы базы данных. То есть в случае базы данных UCS-2 эти сдвиги должны быть определены в UTF-8. Поскольку для некоторых однобайтных символов в UTF-8 требуется больше одного байта, индексы SUBSTR, действительные для однобайтной базы данных, могут не быть действительными для базы данных UCS-2. Если указаны неправильные индексы, будет возвращено SQLCODE -191 (SQLSTATE 22504). Описание поведения этих функций смотрите в справочнике *SQL Reference*.

Типы данных SQL CHAR поддерживаются (в языке C) в пользовательских программах, как тип данных `char`. Типы данных SQL GRAPHIC в пользовательских программах поддерживаются, как тип `sqldbchar`. Обратите внимание на то, что для базы данных UCS-2 данные `sqldbchar` всегда находятся в формате с прямым порядком байтов (первым идет старший байт). Когда прикладная программа связана с базой данных UCS-2, данные символьной строки преобразуются DB2 UDB между кодовой страницей этой программы и UTF-8, но данные графической строки всегда находятся в UCS-2.

Идентификаторы

В базе данных UCS-2 все идентификаторы находятся в многобайтном UTF-8. Поэтому можно использовать любые символы UCS-2 в идентификаторах, в которых использование символа из расширенного набора символов (например, русской буквы или многобайтного символа) разрешено DB2 UDB. Подробнее о том, какие идентификаторы позволяют использовать расширенные символы, смотрите в разделе “Приложение B. Правила именования” на стр. 373.

Клиенты могут вводить любые символы, поддерживаемые их средой SBCS или MBCS; все эти символы в идентификаторах будут преобразованы менеджером баз данных в UTF-8. При задании символов национальных языков в идентификаторах для базы данных UCS-2 следует учитывать два обстоятельства:

- Для каждого символа, кроме символов ASCII, требуется два или три байта. Поэтому *n*-байтный идентификатор может содержать только от *n*/3 до *n* символов в зависимости от доли символов ASCII в нем. Если у вас только один-два символа не относятся к ASCII (например, в слове есть символ с диакритическим значком), значение ближе к *n* символам, а идентификатор, ни один символ которого не входит в ASCII (например, на японском языке), может содержать всего *n*/3 символов.
- Если идентификаторы будут вводиться из разных сред клиентов, они должны быть определены с использованием общего подмножества символов, доступных этим клиентам. Например, если обращение к базе данных UCS-2 производится из сред с латиницей-1, арабским и японским языками, все идентификаторы реально должны ограничиваться ASCII.

Литералы UCS-2

Литералы UCS-2 могут быть заданы двумя способами:

- Как константа графической строки с использованием формата G’...’ или N’....’, описанного в разделе ″Graphic String Constants″ главы ″Language Elements″ справочника *SQL Reference*. Любой заданный таким образом литерал будет преобразован менеджером баз данных из кодовой страницы прикладной программы в UCS-2.
- Как шестнадцатеричная строка UCS-2 с использованием формата UX’....’ или GX’....’. Длина константы, указанной в кавычках после UX или GX, должна

быть кратной четырем шестнадцатеричным цифрам. Каждая группа из четырех цифр представляет один кодовый элемент UCS-2.

При использовании процессора командной строки (CLP) первый метод проще, если символ UCS-2 существует в локальной кодовой странице прикладной программы (например, при вводе любого символа кодовой страницы 850 с терминала, использующего кодовую страницу 850). Второй метод должен использоваться для символов, выходящих за пределы совокупности кодовой страницы прикладной программы (например, при вводе символов японского языка с терминала, использующего кодовую страницу 850).

Поиск по шаблону в базе данных UCS-2

Поиск по шаблону - одна из областей, в которой поведение существующих баз данных MBCS несколько отличается от поведения базы данных UCS-2.

Для баз данных MBCS в DB2 UDB текущее поведение следующее: Если искомая строка содержит данные MBCS, шаблон может включать в себя как символы SBCS, так и MBCS. Специальные символы в шаблоне интерпретируются так:

- Однобайтный символ подчеркивания соответствует одному символу SBCS.
- Двухбайтный символ подчеркивания соответствует одному символу MBCS.
- Символ процента (как SBCS, так и DBCS) соответствует строке из нулевого или большего числа символов SBCS или MBCS.

Если искомые строки содержат двухбайтные строчные графические данные, эти выражения могут содержать только двухбайтные символы. Специальные символы в шаблоне интерпретируются так:

- Двухбайтный символ подчеркивания соответствует одному символу DBCS.
- Двухбайтный символ процента соответствует строке из нулевого или большего числа символов DBCS.

В базе данных UCS-2 нет реальных различий между ″однобайтными″ и ″двухбайтными″ символами; каждый символ UCS-2 занимает два байта. Хотя формат UTF-8 представляет собой кодирование символов UCS-2 со смешанным числом байтов, в нем нет реальных различий между символами SBCS и MBCS. Каждый символ рассматривается как символ UCS-2, независимо от числа его байтов, используемых в формате UTF-8. При указании выражения символьной строки или графической строки символ подчеркивания соответствует одному символу UCS-2, а символ процента соответствует строке из нулевого или большего числа символов UCS-2.

Со стороны клиента выражения символьных строк из кодовой страницы клиента будут преобразованы менеджером баз данных в UTF-8. В кодовых страницах клиента SBCS нет двухбайтных символов процента и подчеркивания, но каждая поддерживаемая кодовая страница содержит однобайтный символ процента

(соответствующий U+0025) и однобайтный символ подчеркивания (соответствующий U+005F). Интерпретация специальных символов для базы данных UCS-2:

- Однобайтный символ подчеркивания (U+0025) соответствует одному символу UCS-2 в выражении графической строки или одному символу UTF-8 в выражении символьной строки.
- Однобайтный символ процента (U+005F) соответствует строке из нулевого или большего числа символов UCS-2 в выражении графической строки или строке из нулевого или большего числа символов UTF-8 в выражении символьной строки.

Кодовые страницы DBCS также поддерживают двухбайтный символ процента (соответствующий U+FF05) и двухбайтный символ подчеркивания (соответствующий U+FF3F). У этих символов нет специальных значений для базы данных UCS-2.

Для необязательного ″эскейп-выражения″, которое указывает, что символ используется для изменения специального значения символов подчеркивания и процента, поддерживаются только символы ASCII или символы, которые расширяются до двухбайтной последовательности UTF-8. Если указать управляющий символ, расширяющийся в трехбайтное значение UTF-8, будет возвращено сообщение об ошибке (SQL0130N, SQLSTATE 22019).

Особенности импорта, экспорта и загрузки

Для базы данных UCS-2 поддерживаются форматы файлов DEL, ASC и PC/IXF, как описано в этом разделе. Формат WSF не поддерживается.

При экспорте из базы данных UCS-2 в файл ASCII с ограничителями (DEL) все символьные данные преобразуются в кодовую страницу прикладной программы. И данные символьных строк, и данные графических строк преобразуются в одну и ту же кодовую страницу SBCS или MBCS клиента. Такое поведение ожидается для экспорта любой базы данных и не может быть изменено, поскольку у всего файла ASCII с ограничителями может быть только одна кодовая страница. Поэтому при экспорте файла ASCII с ограничителями будут сохранены только те символы UCS-2, которые существуют в кодовой странице вашей прикладной программы. Другие символы заменяются на используемый по умолчанию символ замены для кодовой страницы прикладной программы. Для клиентов UTF-8 (кодовая страница 1208) потери данных не происходит, поскольку все символы UCS-2 поддерживаются клиентами UTF-8.

При импорте из файла ASCII (DEL или ASC) в базу данных UCS-2 данные символьных строк преобразуются из кодовой страницы прикладной программы в UTF-8, а данные графических строк преобразуются из кодовой страницы прикладной программы в UCS-2. Потери данных не происходит. Если вы хотите импортировать данные ASCII, которые были сохранены в другой кодовой странице, перед использованием команды IMPORT необходимо изменить

кодовую страницу этого файла данных. Одним из способов сделать это - задать DB2CODEPAGE для кодовой страницы файла данных ASCII.

Диапазон действительных ограничений ASCII для клиентов SBCS и MBCS идентичен тому, что в данное время поддерживается для этих клиентов DB2 UDB. Диапазон действительных ограничителей для клиентов UTF-8 - от 0x01 до 0x7F с обычными ограничениями. Полный список этих ограничений смотрите в приложении ″Export/Import/Load Utility File Formats″ к книге *Data Movement Utilities Guide and Reference*.

При экспорте из базы данных UCS-2 в файл PC/IXF данные символьных строк преобразуются в кодовую страницу SBCS/MBCS клиента. Данные графических строк не преобразуются и сохраняются в UCS-2 (кодовая страница 1200). Потери данных не происходит.

При импорте из файла PC/IXF в базу данных UCS-2 данные символьных строк считаются данными в кодовой странице SBCS/MBCS, хранящейся в заголовке PC/IXF, а данные графических строк считаются данными в кодовой странице DBCS, хранящейся в заголовке PC/IXF. Данные символьных строк преобразуются утилитой импорта из кодовой страницы, указанной в заголовке PC/IXF, в кодовую страницу клиента, а затем из кодовой страницы клиента в UTF-8 (при помощи оператора INSERT). Данные графических строк преобразуются утилитой импорта из кодовой страницы DBCS, указанной в заголовке PC/IXF, непосредственно в UCS-2 (кодовая страница 1200).

Утилита загрузки помещает данные непосредственно в базу данных и по умолчанию считает, что данные в файлах ASC или DEL будут находиться в кодовой странице базы данных. Поэтому по умолчанию для файлов ASCII преобразования кодовых страниц не происходит. Когда для файла данных явно указана кодовая страница (с использованием модификатора `codepage`), утилита загрузки перед загрузкой данных использует эту информацию для преобразования из указанной кодовой страницы в кодовую страницу базы данных. Для файлов PC/IXF утилита загрузки всегда выполняет преобразование из кодовых страниц, указанных в заголовке IXF, в кодовую страницу базы данных (1208 для CHAR и 1200 для GRAPHIC).

Кодовая страница файлов DBCLOB для UCS-2 - всегда 1200. Кодовая страница для файлов CLOB - та же самая, что и кодовая страница для импортируемых, загружаемых или экспортируемых файлов данных. Например, при загрузке или импорте данных с использованием формата PC/IXF считается, что файл большого символьного объекта использует кодовую страницу, указанную в заголовке PC/IXF. Если файл двухбайтного символьного большого объекта находится в формате ASC или DEL, утилита загрузки считает, что для данных символьного большого объекта используется кодовая страница базы данных

(если явным образом при помощи модификатора `codepage` не указано иное), а утилита импорта считает, что для них используется кодовая страница прикладной программы клиента.

Для базы данных UCS-2 модификатор `nochecklengths` указывается всегда, поскольку:

- Любой SBCS может быть связан с базой данных, для которой нет кодовой страницы DBCS
- У символьных строк в формате UTF-8 длина обычно отличается от их длины в кодовых страницах клиентов.

Дополнительную информацию об утилитах загрузки, импорта и экспорта смотрите в руководстве *Data Movement Utilities Guide and Reference*.

Несовместимости

В прикладных программах, связанных с базой данных UCS-2, для данных графических строк всегда используется UCS-2 (кодовая страница 1200). В прикладных программах, связанных с другими базами данных, для данных графических строк используется кодовая страница DBCS прикладной программы, если же кодовая страница прикладной программы - страница SBCS, такие данные на разрешены. Например, если клиент 932 связан с базой данных (не UCS-2) с японским языком, для данных графических строк используется кодовая страница 301. В прикладных программах клиента 932, связанных с базой данных UCS-2, для данных графических строк используется UCS-2.

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 СОБЛЮДЕНИЯ ЧЬИХ-ЛИБО АВТОРСКИХ ПРАВ, 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 получены в системах разработки и нет никаких гарантий, что в общедоступных системах эти показатели будут теми же. Более того, некоторые результаты, возможно, были 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 примеры 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
Сервер OLAP DB2
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
SystemView
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

Б

В

Г

Д

Е

Ж

З

И

К

Л

М

Н

О

П

P

C

T

У

Ф

Х

Ц

Ч

Ш

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.

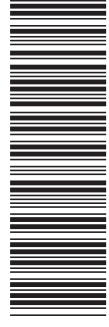
Информацию о том, как связаться с IBM из других стран, смотрите в Приложении A книги *IBM Software Support Handbook*. Этот документ можно найти в Web, обратившись по адресу: http://www.ibm.com/support/ и выбрав ссылку на IBM Software Support Handbook у нижнего края страницы.

Примечание: В некоторых странах авторизованные дилеры IBM должны обращаться не в центр поддержки IBM, а в структуры поддержки дилеров.

Код изделия: CT7XVRU

Напечатано в Дании

(1P) P/N: CT7XVRU



SH43-0146-00