IBM® DB2® Universal Database

Справочное руководство по восстановлению данных и высокой доступности

Версия 7

IBM® DB2® Universal Database

Справочное руководство по восстановлению данных и высокой доступности

Версия 7

Перед тем как использовать данный документ и продукт, описанный в нем, прочтите общие сведения под заголовком “Приложение K. Замечания” на стр. 539.

Этот документ содержит информацию, которая является собственностью IBM. Она предоставляется в соответствии с лицензионным соглашением и защищена законами об авторском праве. Информация в данной публикации не включает никаких гарантий на продукт и никакое из утверждений в данном руководстве не следует понимать подобным образом.

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

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

© Copyright International Business Machines Corporation 2001. Все права защищены.

Содержание

© Copyright IBM Corp. 2001

Об этой книге

В этой книге подробно рассказано об утилитах резервного копирования и восстановления IBM DB2 Universal Database (UDB) и показано их использование. Кроме того, в книге объясняется значение высокой доступности и описывается поддержка восстановления при отказах DB2 на различных платформах.

Для кого предназначена эта книга

Это руководство предназначено для администраторов база данных, прикладных программистов и других пользователей DB2 UDB, которые отвечают за операции резервного копирования и восстановления в системах баз данных DB2, или для тех, кто хочет эти операции освоить.

Предполагается, что вы хорошо знаете DB2 Universal Database, SQL (Structured Query Language - язык структурированных запросов) и среду операционной системы, в которой запускается DB2 UDB. Общую информацию о DB2 UDB смотрите в книге *Administration Guide*. Информацию о языке SQL смотрите в справочнике *SQL Reference*. Информацию о конфигурировании, вызовах и использовании процессора командной строки DB2 UDB смотрите в справочнике *Command Reference*. Информацию об API (application programming interface - интерфейс прикладного программирования) для DB2 UDB смотрите в справочнике *Administrative API Reference*. Общую информацию о создании программ, содержащих API управления DB2, смотрите в руководстве *Application Building Guide*. В этом руководстве не даются инструкции по установке DB2, так как они зависят от вашей операционной системой. Информацию об установке можно найти в соответствующей книге *Быстрый старт* для конкретной операционной системы.

Структура книги

Рассматриваются следующие темы:

Восстановление данных

Глава 1. Разработка правильной стратегии резервного копирования и восстановления

Обсуждаются различные показатели при выборе методов восстановления базы данных и табличных пространств, включая резервное копирование и восстановление базы данных и табличных пространств, а также применение восстановления с повтором транзакций.

© Copyright IBM Corp. 2001

Глава 2. Резервное копирование базы данных

Рассматривается утилита резервного копирования DB2, используемая для создания резервных копий базы данных или табличных пространств.

Глава 3. Восстановление базы данных

Описывается утилита восстановления DB2, используемая для воссоздания поврежденных или неисправных базы данных или табличных пространств, для которых предварительно были изготовлены резервные копии.

Глава 4. Восстановление с повтором транзакций

Описывается утилита повтора транзакций DB2, используемая для восстановления базы данных с использованием транзакций, записанных в файлах журнала восстановлений базы данных.

Высокая доступность

Глава 5. Высокая доступность и восстановление при отказах. Введение

Обзор поддержки восстановления при отказах для обеспечения высокой доступности в DB2.

Глава 6. Высокая доступность в AIX

Обсуждается поддержка DB2 высокодоступного восстановления при отказах в AIX, которая сейчас реализована при помощи возможности ES (Enhanced Scalability - расширенная масштабируемость) для HACMP for AIX (High Availability Cluster Multi-processing - высокодоступная кластерная параллельная обработка).

Глава 7. Высокая доступность в операционной системе Windows

Обсуждается поддержка DB2 высокодоступного восстановления при отказах в Windows NT, которая сейчас осуществляется при помощи сервера Microsoft Cluster Server (MSCS).

Глава 8. Высокая доступность в операционной среде Solaris

Обсуждается поддержка DB2 высокодоступного восстановления при отказах в операционной среде Solaris, которая сейчас осуществляется при помощи серверов Sun Cluster 2.x (SC2.x), Sun Cluster 3.0 (SC3.0) или Veritas Cluster Server (VCS).

Приложения

Приложение A. Как читать синтаксические диаграммы

Объясняются соглашения, используемые в синтаксических диаграммах.

Приложение B. Предупреждения, сообщения об ошибках и завершении

Даются объяснения сообщений, генерируемых менеджером баз данных при обнаружении ошибочных ситуаций или ситуаций предупреждений.

Приложение C. Дополнительные команды DB2

Описаны команды DB2, связанные с восстановлением.

Приложение D. Дополнительные API и связанные с ними структуры данных
Описаны API, относящиеся к восстановлению, и их структуры данных.

Приложение E. Программы примеров восстановления
Приводятся исходные тексты программ примеров, в которых содержатся относящиеся к восстановлению вызовы API и встроенного SQL DB2, и объясняется их использование.

Приложение F. Сценарий восстановления CLP
Приводится исходный текст командного сценария DB2, в котором содержатся относящиеся к восстановлению команды процессора командной строки, и объясняется их использование.

Приложение G. Tivoli Storage Manager
Содержит информацию о продукте Tivoli Storage Manager (TSM, прежнее название - ADSM), который может использоваться для управления операциями резервного копирования базы данных или табличных пространств.

Приложение H. Обработчик пользователя для восстановления баз данных
Обсуждаются возможности использования программ обработчиков пользователей с файлами журналов баз данных и описывается несколько программ примеров.

Приложение I. API резервного копирования и восстановления для продуктов других поставщиков
Описывается функции и использование API, которые позволяют DB2 взаимодействовать с программным обеспечением других поставщиков (не IBM).

Часть 1. Восстановление данных

© Copyright IBM Corp. 2001

Глава 1. Разработка правильной стратегии резервного копирования и восстановления

База данных может оказаться непригодной для использования из-за ошибки аппаратных средств, программного обеспечения или общей ошибки. Вы можете время от времени сталкиваться с проблемами ошибок памяти, отключениями питания, ошибками программ; различные сценарии отказов требуют различных процедур восстановления. Чтобы защитить ваши данные от возможных потерь, необходима тщательно продуманная стратегия. Вот некоторые из вопросов, на которые надо ответить при разработке стратегии восстановления: Будет ли возможность восстановить базу данных? Сколько времени можно потратить на восстановление базы данных? Как часто будет выполняться резервное копирование? Сколько места можно выделить для резервных копий и архивных журналов? Достаточно ли выполнять копирование на уровне табличных пространств,или же необходимо полное копирование базы данных?

Стратегия восстановления базы данных должна гарантировать, что вся информация будет доступной, когда она потребуется для восстановления базы данных. Она должна задавать график регулярного снятия резервных копий базы данных, в том числе для систем многораздельных баз данных - резервных копий для масштабирования системы (на случай добавления или отбрасывания узлов). Общая стратегия должна включать также процедуры восстановления командных сценариев, программ, пользовательских функций, кодов хранимых процедур в библиотеках операционной системы и загрузочных копий.

В этом разделе обсуждаются различные методы восстановления; вы должны определить, какой метод восстановления лучше подходит для вашей конкретной среды.

Идея *резервного копирования* базы данных - та же, что и для любых других данных: копия данных снимается и затем 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. Если затем база данных будет каким-либо образом повреждена или испорчена, вы сможете ее воссоздать.

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

© Copyright IBM Corp. 2001

повторное применение транзакций, записанных в файлах журналов базы данных, после того, как база данных или табличное пространство восстановлены из образа резервной копии.

Восстановление после отказа - это автоматическое восстановление базы данных в случае отказа до момента завершения и принятия всех изменений, составляющих часть одной или нескольких единиц работы (транзакций). Это достигается откатом незавершенных транзакций и завершением принятых транзакций, которые еще оставались в памяти к моменту отказа.

Подробную информацию о различных методах восстановления смотрите в разделах “Восстановление версии” на стр. 24, “Восстановление с повтором транзакций” на стр. 25 или “Восстановление после отказа” на стр. 11.

Файлы журналов восстановления и файл хронологии восстановления создаются автоматически при создании базы данных (рис. 1 на стр. 5). Файл журнала восстановления или файл хронологии восстановления нельзя изменять непосредственно; однако они незаменимы при использовании образа резервной копии базы данных для восстановления потерянных или поврежденных данных.

Объект/концепт базы данных	Эквивалентный физический объект

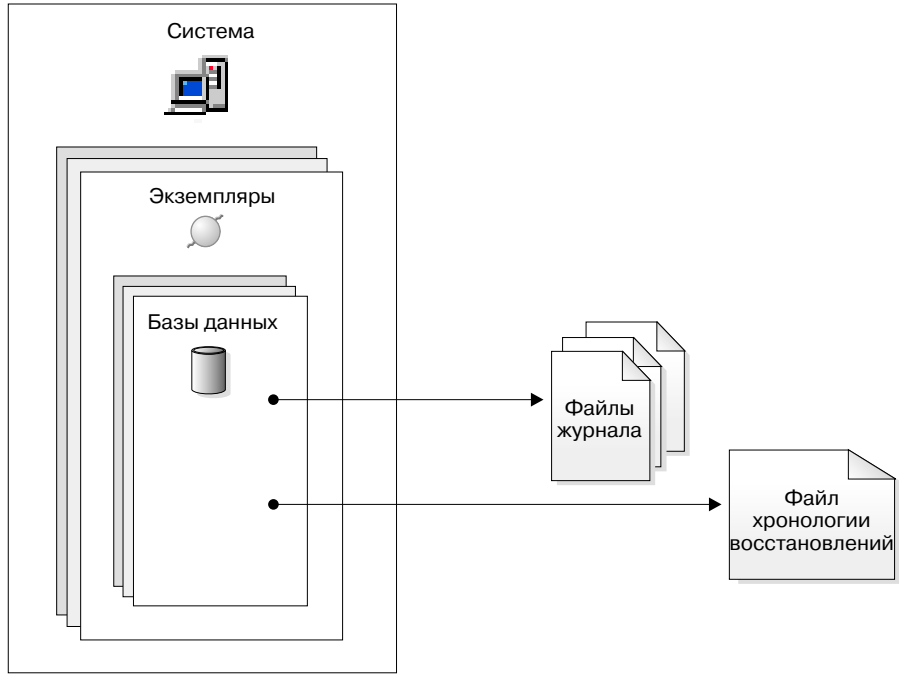


Рисунок 1. Файлы журнала восстановления и файл хронологии восстановления

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.

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, в частности, операций резервного копирования и восстановления.

Те данные, которые легко воссоздать, можно хранить в невосстановимых базах данных. К таким данным относятся данные из внешних источников, используемые только для чтения,а также данные, малый объем операций с которыми делает неоправданным дополнительную сложности управления файлами журналов и повтора транзакций при восстановлении. У *невосстановимых баз данных* оба параметра конфигурации, *logretain* и *userexit*,

отключены. Это значит, что хранятся только журналы, требующиеся для восстановления после отказа. 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. Она восстанавливается до состояния, предшествующего созданию образа ее резервной копии.

Данные, которые *нельзя* легко воссоздать, надо хранить в восстановимой базе данных. К ним относятся данные, источник которых удаляется после их загрузки, и данные, которые модифицируются прикладными программами или пользователями после загрузки. У *восстановимых баз 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. Такую базу данных можно восстановить только в автономном режиме. Она восстанавливается до состояния, предшествующего 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.

Как часто выполнять резервное копирование

План восстановления должен предполагать регулярное создание резервных копий, поскольку резервное копирование базы данных требует времени и системных ресурсов. Ваш план может предполагать сочетание полных и инкрементных образов резервных копий (смотрите раздел “Инкрементное резервное копирование и восстановление” на стр. 28).

Следует регулярно создавать полные резервные копии базы данных, даже если вы архивируете журналы (нужные для восстановления с повтором транзакций). Воссоздать базу данных из набора образов копий табличных пространств сложнее, чем из образа полной резервной копии. Образы копий табличных пространств полезны для восстановления в случае ошибки отдельного диска или программной ошибки.

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

Если главный фактор - время, необходимое на применение архивных журналов при восстановлении или повторе транзакций, рассмотрите возможность выполнять резервное копирование чаще. При этом сократится число архивных журналов, необходимых для повтора транзакций.

Резервное копирование можно запускать как при *подключенной*, так и при *отключенной* базе данных. При резервном копировании без отключения базы данных другие программы и процессы могут соединяться с ней, а также читать и изменять данные во время копирования. При автономном резервном копировании другие программы *не могут* соединяться с базой данных.

Оперативное резервное копирование позволяет сократить время, в течении которого база данных остается недоступной. Оперативное резервное копирование поддерживается, только если включена поддержка восстановления с повтором транзакций. При поддержке восстановления с повтором транзакций и наличии полного набора журналов восстановления базу данных можно восстановить, когда бы в этом ни возникла потребность. Оперативную резервную копию можно использовать только при наличии журналов базы данных, покрывающих время создания данной резервной копии.

Автономное резервное копирование выполняется быстрее, чем оперативное резервное копирование.

Если база данных содержит много данных длинных полей и больших объектов, резервное копирование базы данных может отнять очень много времени. Утилита резервного копирования позволяет копировать выбранные табличные пространства. 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 воспроизводятся из отдельного источника, при создании или изменении таблицы, содержащей столбцы больших объектов, выберите опцию NOT LOGGED.

Примечание: Если вы размещаете данные длинных полей, данные больших объектов и индексы в отдельных табличных пространствах, но не копируете их вместе, необходимо учитывать следующее: 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.

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.

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 формулировании 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.

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 выбрать 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 “Параметры конфигурации для записи в журнал базы данных” на стр. 40.)

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. Например, 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 “Приложение H. Обработчик пользователя для восстановления баз данных” на стр. 471.)

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 нужна возможность выполнять восстановление с повтором транзакций до указанного момента времени для табличных пространств.

Использование различных операционных систем

При работе в среде, где используется несколько операционных систем, следует учитывать, что в большинстве случаев планы резервного копирования и восстановления могут не интегрироваться. Это значит, что, получив копию базы данных в одной операционной системе, нельзя восстановить ее в другой операционной системе. В таких случаях для каждой операционной системы планы восстановления должны быть отдельными и независимыми.

Однако между системами Sun Solaris и HP возможны операции межплатформенного копирования и восстановления. Передача резервной копии между системами должна происходить в двоичном режиме. В системе назначения база данных должна быть создана с той же кодовой страницей и территорией, что и система, в которой была создана исходная база данных.

Если вам надо переместить таблицы из одной операционной системы в другую, а межплатформенные операции копирования и восстановления для вашей среды

не поддерживаются, можно воспользоваться командой **db2move** или утилитой export, а затем утилитой import или load. Дополнительную информацию об этих утилитах смотрите в руководстве *Data Movement Utilities Guide and Reference*.

Восстановление после отказа

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. *Восстановление после отказа* - это процесс, при котором база данных возвращается в согласованное и пригодное к использованию состояние. Это достигается откатом незавершенных транзакций и завершением принятых транзакций, которые еще оставались в памяти к моменту сбоя (смотрите рис. 2). Когда база данных находится в согласованном и пригодном к использованию состоянии, говорят, что она достигла так называемой ″точки согласованности″. *Ошибка транзакции* происходит в результате серьезной ошибки или условия,

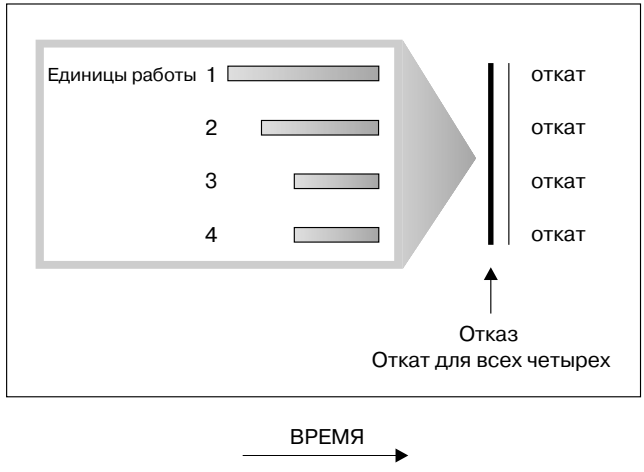


Рисунок 2. Откат единиц работы (восстановление после отказа)

которые приводят к ненормальному завершению работы базы данных или менеджера баз данных. Частично завершенные единицы работы, не записавшие результаты на диск к моменту ошибки, оставляют базу данных в несогласованном и непригодном к использованию состоянии. После ошибки транзакции база данных должна быть восстановлена. Возможные причины, которые могут привести к ошибке транзакции:

- Отказ питания компьютера, который приводит к прекращению работы менеджера базы данных и разделов базы данных.
- Серьезная ошибка операционной системы, которая приводит к прекращению работы DB2.

Если вы хотите, чтобы менеджер баз данных автоматически выполнял откат незавершенных единиц работы, включите параметр конфигурации автоматического перезапуска (*autorestart*), задав для него значение ON. (Это значение по умолчанию. Дополнительную информацию об этом параметре смотрите в руководстве *Administration Guide: Performance*.) Если вы не хотите использовать автоматический перезапуск, при ошибке базы данных надо выполнить команду RESTART DATABASE. При начале восстановления базы данных производится запись файла db2diag.log.

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 обычно будут пригодными для использования, и с базой данных могут быть установлены соединения.

Восстановление поврежденных табличных пространств

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
- Восстановить табличное пространство каталога. (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.

Если поврежденное табличное пространство - единственное временное табличное пространство, следует создать новое временное табличное пространство, как только с базой данных можно будет установить 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.

Уменьшения 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.
- Используйте конфигурацию RAID (Redundant Array of Independent Disks - дублированный массив независимых дисков), например, RAID уровня 5. Дополнительную информацию о RAID смотрите в разделе “Защита от ошибок диска” на стр. 15.
- 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 - дублированный массив независимых 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: В аппаратном массиве дисков 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 несколько типов архитектуры RAID, отличающихся по функциям и производительности, однако на сегодня широко используются толькоRAID уровня 1 и уровня 5.

RAID уровня 1 называют также зеркальным копированием или дублированием дисков. *Зеркальное копирование дисков* дублирует данные (целые файлы) 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). При этом обеспечивается достаточная защита данных: при ошибке на одном диске данные остаются доступны на другом. При дублировании дисков возможная ошибка контроллера диска не ставит под угрозу защиту данных. Производительность при этом хорошая, однако число необходимых дисков возрастает вдвое.

В RAID уровня 5 по всем дискам используется чередование данных и информации о четности по секторам. Информация о четности чередуется с данными, а не сохраняется на особом дисководе. При этом обеспечивается достаточная защита данных: при ошибке любого диска данные все равно остаются доступными, так как можно использовать информацию с других

дисков, а также информацию о четности. Производительности чтения высокая, однако производительность записи низкая. Для конфигурации RAID уровня 5 требуется минимум три одинаковых диска. Величина дискового пространства, расходуемого на служебную информацию, зависит от количества дисков в массиве. Если в конфигурации RAID уровня 5 пять дисков, дополнительный расход дискового пространства составляет 20 процентов.

При использовании дискового массива RAID (кроме RAID уровня 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 уровня 5 ошибка случается на двух дисках одновременно, все данные теряются (но вероятность такой ошибки очень мала).

Для журналов можно рассмотреть возможность использования аппаратного или программного массива дисков RAID уровня 1 (смотрите раздел “Программные массивы дисков”); это обеспечит возможность восстановления до момента времени ошибки и высокую производительность записи (что важно для журналов). В случаях, где критична надежность (поскольку нельзя терять время на восстановление данных после ошибки), а производительность записи не так критична, рассмотрите возможность использования аппаратного массива дисков RAID уровня 5. Наоборот, если критична производительность записи, а стоимость дополнительного дискового пространства малозначима, рассмотрите возможность использования и для данных и для журналов аппаратного массива дисков RAID уровня 1.

Подробную информацию о возможных уровнях RAID смотрите на странице:

```
http://www.acnc.com/04_01_00.html
```

Программные массивы дисков: Программный массив дисков выполняет те же функции, что и аппаратный (смотрите раздел “Аппаратные массивы дисков” на стр. 15), но трафик дисков управляется 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” на стр. 15). Хотя для программного массива дисков и требуются дополнительные диски, реализация его обходится дешевле, так как не нужны дорогостоящие контроллеры дисков.

ОСТОРОЖНО:
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 разрешить допуск к диску. Загрузочный диск нужно отделить от массива дисков.

Уменьшения 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 многораздельной базы данных” на стр. 156).

Восстановление при ошибках транзакций в среде многораздельных баз данных

Если ошибка транзакции происходит в среде многораздельной базы данных, восстановление базы данных обычно необходимо и для сервера раздела базы данных, где произошла ошибка, и для любого другого сервера раздела базы данных, который участвовал в транзакции:

- Восстановление после отказа выполняется на отказавшем сервере раздела базы данных после устранения причины ошибки.
- *Восстановление раздела базы данных* на остальных серверах разделов базы данных (сохранивших активность) происходит сразу после обнаружения ошибки.

В среде многораздельных баз данных сервер раздела базы данных, на котором запущена прикладная программа, называется узлом координатора, а первый агент, работающий для этой прикладной программы, - агентом координатора. Агент координатора отвечает за распределение работы между другими серверами разделов базы данных и наблюдает за теми из них, которые принимают участие в транзакции. Когда прикладная программа выполняет для транзакции оператор COMMIT, агент координатора принимает транзакцию, используя протокол двухфазного принятия. На первой фазе узел координатора передает требование PREPARE всем остальным серверам разделов базы данных, участвующим в транзакции. Серверы отвечают одним из следующих способов:

READ-ONLY	На сервере не произошло изменение данных
YES	На сервере произошло изменение данных
NO	Сервер не готов к принятию из-за ошибки

Если один из серверов ответил NO, выполняется откат транзакции. В противном случае узел координатора начинает вторую фазу.

На второй фазе узел координатора записывает в журнал запись COMMIT, а затем передает требование COMMIT всем серверам, ответившим YES. После выполнения принятия всеми остальными серверами разделов базы данных они посылают узлу координатора подтверждение COMMIT. Транзакция завершается, когда агент координатора получит подтверждения COMMIT от всех участвующих в ней серверов. После этого агент координатора делает в журнал запись FORGET.

Более подробную информацию о двухфазном принятии смотрите в книге *Administration Guide: Planning*.

Восстановление после ошибки транзакции на активном сервере раздела базы данных

Если какой-либо из серверов разделов базы данных обнаружил, что другой сервер не работает, всякая работа, связанная с последним, останавливается:

- Если сохранивший активность сервер раздела базы данных является узлом координатора для прикладной программы, и эта программа была запущена на совершившем ошибку сервере раздела базы данных (который не готов к COMMIT), работа агента координатора прерывается для выполнения восстановления после ошибки. Если агент координатора находится на второй фазе обработки COMMIT, прикладная программа получает сообщение SQL0279N и теряет соединение с базой данных. В противном случае агент координатора выдает всем остальным участвующим в транзакции серверам требование ROLLBACK, и программе будет возвращено сообщение SQL1229N.
- Если сервер раздела базы данных, где произошла ошибка, являлся для этой программы узлом координатора, агенты, продолжающие работу для этой программы на активных серверах, прерываются для выполнения восстановления после ошибки. Откат текущей транзакции выполняется локально на каждом из серверов, если он не был подготовлен и не ожидает итогов транзакции. В такой ситуации транзакция остается неоднозначной на активных серверах разделов базы данных, и узел координатора об этом не знает (поскольку она недоступна).

  Дополнительную информацию о том, как разрешаются неоднозначные транзакции, смотрите в книге *Administration Guide: Planning*.
- Если прикладная программа была (до совершения ошибки) связана с сервером раздела базы данных, где произошла ошибка, но ни локальный сервер раздела базы данных, ни совершивший ошибку сервер раздела базы данных не являются узлом координатора, агенты, работающие для этой программы, прерываются. Узел координатора пошлет другим серверам разделов базы данных либо сообщение ROLLBACK, либо сообщение о

разрыве соединения. Если узел координатора возвратит сообщение SQL0279, транзакция будет неоднозначной только на сохранивших активность серверах разделов базы данных.

Если какой-либо процесс (такой, как агент или детектор тупиковых ситуаций) пытается послать требование совершившему ошибку серверу, ему будет сообщено о невозможности этого.

Восстановление после ошибки транзакции на сервере раздела базы данных, где произошла ошибка

Если ошибка транзакции приводит к ненормальному завершению работы менеджера баз данных,вы можете выполнить команду **db2start** с опцией RESTART для перезапуска менеджера баз данных после перезапуска процессора. При невозможности перезапуска процессора команду **db2start** можно использовать для перезапуска менеджера баз данных на другом процессоре. Дополнительную информацию смотрите в книге *Command Reference*.

Если работа менеджера баз данных завершается ненормально, разделы базы данных на этом сервере могут остаться в несогласованном состоянии. Чтобы сделать их пригодными к использованию, можно запустить восстановление после отказа на сервере раздела базы данных:

- Явным образом при помощи команды RESTART DATABASE
- Неявным образом при помощи требования CONNECT, если параметр конфигурации базы данных*autorestart* имеет значение ON

Восстановление после отказов использует записи журнала в активных файлах журнала, чтобы обеспечить отражение в базе данных действий всех завершенных транзакций. После применения изменений для всех непринятых транзакций (*кроме* неоднозначных) выполняется локальный откат. В среде многораздельной базы данных бывают два типа неоднозначных транзакций:

- На сервере раздела базы данных, не являющемся узлом координатора, транзакция неоднозначная, если она подготовлена, но еще не принята.
- На узле координатора транзакция неоднозначная, если она принята, но еще не записана в журнал как завершенная (то есть еще нет записи FORGET). Такая ситуация происходит, когда агент координатора не получил подтверждений COMMIT от всех серверов, работавших для прикладной программы.

Восстановление после отказов пытается разрешить все неоднозначные транзакции одним из следующих способов. Предпринимаемое действие зависит от того, был ли сервер раздела базы данных узлом координатора для прикладной программы:

- Если перезапущенный сервер не является для программы узлом координатора, он посылает агенту координатора запрос для выяснения результата транзакции.

- Если перезапущенный сервер *является* для программы узлом координатора, он посылает всем другим агентам (подчиненным) сообщение о том, что агент координатора все еще ожидает подтверждений COMMIT.

Возможно, что при восстановлении после отказа не удастся разрешить все неоднозначные транзакции (например, могут быть недоступны отдельные серверы разделов базы данных). В таком случае возвращается предупреждение SQL SQL1061W. Поскольку неоднозначные транзакции удерживают ресурсы,такие как блокировки и пространство активного журнала, есть возможность дойти до точки, когда изменения в базе данных станут невозможно из-за того, что пространство активного журнала удерживается неоднозначными транзакциями. Поэтому необходимо определить, остались ли после восстановления после отказа неоднозначные транзакции, и как можно быстрее восстановить все серверы разделов базы данных, необходимые для разрешения этих неоднозначных транзакций.

Если невозможно вовремя восстановить один или несколько серверов, необходимых для разрешения неоднозначной транзакции, и требуется доступ к разделам базы данных на других серверах, можно разрешить эту неоднозначную транзакцию вручную, приняв решение эвристически. Для запроса, принятия или отката неоднозначной транзакции на сервере можно использовать команду LIST INDOUBT TRANSACTIONS (смотрите книгу *Command Reference*).

Примечание: Команда LIST INDOUBT TRANSACTIONS используется также в среде распределенных транзакций. Чтобы различить эти два типа неоднозначных транзакций, в поле *источник* вывода, возвращенного командой LIST INDOUBT TRANSACTIONS, может быть показано следующее:

- DB2 Universal Database Enterprise - Extended Edition (указывает, что источником транзакции является среда многораздельных баз данных).
- XA (указывает, что источником транзакции является распределенная среда).

Более подробную информацию о распределенной среде смотрите в книге *Administration Guide: Planning*.

Определение сервера раздела базы данных, где произошла ошибка

При ошибке сервера раздела базы данных прикладная программа, скорее всего, получит один из следующих кодов SQLCODE. Способ определения менеджера баз данных, где произошла ошибка, зависит от полученного SQLCODE:

SQL0279N

Этот SQLCODE передается, когда сервер раздела базы данных, участвующий в транзакции, остановлен во время обработки COMMIT.

SQL1224N
Этот SQLCODE передается, когда совершивший ошибку сервер раздела базы данных является для транзакции узлом координатора.

SQL1229N
Этот SQLCODE передается, когда совершивший ошибку сервер раздела базы данных не является для транзакции узлом координатора.

Определение сервера раздела базы данных, где произошла ошибка, выполняется в два шага. В SQLCA, связанной с SQLCODE SQL1229N, в шестой позиции массива поля *sqlerrd* содержится номер узла сервера, обнаружившего ошибку. (Номер узла, записанный для этого сервера, соответствует номеру узла в файле `db2nodes.cfg`.) На сервере раздела базы данных, обнаружившем ошибку, в файл `db2diag.log` записывается сообщение, указывающее номер узла сервера, где произошла ошибка.

Примечание: Если на процессоре использовалось несколько логических узлов, ошибка на одном из них может привести к ошибкам на других узлах на этом же процессоре.

Для восстановления после ошибки на сервере раздела базы данных:

1. Устраните причину ошибки.
2. Перезапустите менеджер баз данных командой **db2start** с любого сервера раздела базы данных.
3. Перезапустите базу данных при помощи команды RESTART DATABASE на сервере или серверах раздела базы данных, где произошла ошибка.

Восстановление неоднозначных транзакций на хосте

Если ваша прикладная программа во время транзакции связалась с сервером базы данных хоста или AS/400, возникают определенные различия в восстановлении неоднозначных транзакций.

Для доступа к хосту или серверам базы данных AS/400 используется DB2 Connect. Шаги восстановления различаются, если у DB2 Connect сконфигурирован Менеджер точек синхронизации DB2.

Восстановление для случая, когда у DB2 Connect сконфигурирован Менеджер точек синхронизации DB2

Восстановление неоднозначных транзакций на серверах хоста или AS/400 обычно выполняется автоматически менеджером транзакций (Transaction Manager - TM) и менеджером точек синхронизации DB2 (SPM). Неоднозначная транзакция на сервере хоста или AS/400 не удерживает никаких ресурсов на локальной системе DB2, но удерживает ресурсы на сервере хоста или AS/400 все время, пока она остается неоднозначной. Если администратор хоста или сервера AS/400 обнаружил, что необходимо эвристическое решение, он может связаться с локальным администратором базы данных DB2 (например, по телефону),

чтобы решить, выполнять для этой транзакции на хосте или сервере AS/400 принятие или откат. В этом случае для определения состояния транзакции на экземпляре DB2 Connect можно использовать команду LIST DRDA INDOUBT TRANSACTIONS. В качестве руководства в большинстве ситуаций, где вовлечена коммуникационную среду SNA, можно использовать следующие шаги.

1. Соединитесь с SPM, как показано ниже:

```
db2 => connect to db2spm

 Database Connection Information

 Database product       = SPM0500
 SQL authorization ID   = CRUS
 Local database alias   = DB2SPM
```

2. Введите команду LIST DRDA INDOUBT TRANSACTIONS для вывода известных SPM неоднозначных транзакций. В показанном далее примере SPM известна одна неоднозначная транзакция. db_name ниже - это локальный алиас сервера хоста или AS/400. partner_lu - полное имя логического устройства сервера хоста или AS/400. Эти имена лучше идентифицируют сервера хоста или AS/400 и предоставляются вызывающей программой с сервера хоста или AS/400. luwid - уникальный идентификатор транзакции, который доступен на всех серверах хостов и AS/400. Если требуемая транзакция показана, для определения ее результата можно использовать поле uow_status, которое может содержать C (commit - принятие) или R (rollback - откат). Если ввести команду LIST DRDA INDOUBT TRANSACTIONS с параметром WITH PROMPTING, принимать, откатывать или отказываться от транзакции можно в интерактивном режиме. Дополнительную информацию смотрите в книге *Command Reference*.

```
db2 => list drda indoubt transactions
 DRDA Indoubt Transactions:
 1.db_name: DBAS3    db_alias: DBAS3    role: AR
   uow_status: C  partner_status: I  partner_lu: USIBMSY.SY12DQA
 corr_tok: USIBMST.STB3327L
 luwid: USIBMST.STB3327.305DFDA5DC00.0001
 xid: 53514C2000000017 00000000544D4442 0000000000305DFD A63055E962000000
      00035F
```

3. Если неоднозначная транзакция для partner_lu и для luwid не показана или если команда LIST DRDA INDOUBT TRANSACTIONS возвратила:

```
db2 => list drda indoubt transactions
SQL1251W  По эвристическому запросу не получено никаких данных.
```

это значит, что для транзакции выполнен откат.

Могла произойти и другая маловероятная, но возможная ситуация. Если показана неоднозначная транзакция с правильными luwid для partner_lu, но для uow_status показано ″I″, SPM не знает, будет ли для этой транзакции выполняться принятие или откат. В таком случае, чтобы принять или

откатить транзакцию на рабочей станции DB2 Connect, необходимо использовать параметр WITH PROMPTING. Затем позвольте DB2 Connect повторно синхронизироваться с сервером хоста или AS/400 на основе эвристического решения.

Восстановление, когда DB2 Connect не использует Менеджер точек синхронизации DB2

Информация этого раздела относится к случаю, когда связь по TCP/IP используется для обновления DB2 for OS/390 при обновлении нескольких узлов в DB2 Connect Personal Edition или DB2 Connect Enterprise Edition, а Менеджер точек синхронизации DB2 не используется. Восстановление неоднозначных транзакций в этом случае отличается от восстановления с использованием Менеджер точек синхронизации DB2. Когда в такой среде происходит неоднозначная транзакция, предупредительная запись генерируется на клиенте, на сервере базы данных и/или в базе данных менеджера транзакций (TM) в зависимости от того, кто из них обнаружил ошибку. Эта предупредительная запись помещается в файл `db2alert.log`. Дополнительную информацию об оповещениях смотрите в руководстве *Troubleshooting Guide*.

Ресинхронизация любой неоднозначной транзакции происходит автоматически, как только снова станут доступными менеджер транзакций, участвующие базы данных и соединения между ними. Следует позволять происходить автоматической повторной синхронизации, а не принимать принудительное эвристическое решение на сервере базы данных. Если, тем не менее, приходится это делать, в качестве руководства используйте следующие шаги.

Примечание: Поскольку Менеджер точек синхронизации DB2 не используется, команду LIST DRDA INDOUBT TRANSACTIONS использовать нельзя.

1. На хосте OS/390 введите команду DISPLAY THREAD TYPE(INDOUBT).

   Из полученного списка выберите транзакцию, которую вы хотите завершить эвристически. Подробности о команде DISPLAY смотрите в справочнике *DB2 for OS/390 Command Reference*. Показанный LUWID можно сопоставить с luwid в базе данных менеджера транзакций.

2. Введите команду RECOVER THREAD(<LUWID>) ACTION(ABORT|COMMIT) (в зависимости от того, что вы хотите сделать).

   Подробности о команде RECOVER смотрите в справочнике *DB2 for OS/390 Command Reference*.

Аварийное восстановление

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:

Аварийное восстановление

- 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, требуется 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. Более подробную информацию смотрите в разделе “Переопределение контейнеров табличных пространств при операции восстановления (перенаправленное восстановление)” на стр. 119.

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.

Восстановление версии

Восстановление версии - это восстановление предыдущей версии базы данных с использованием образа, созданного при резервном копировании. Этот метод восстановления используется для невосстановимых баз данных (то есть для таких, для которых нет архивированных журналов). При помощи опции WITHOUT ROLLING FORWARD команды RESTORE DATABASE этот метод можно использовать и для восстановимых баз данных. Этой операция восстановления воссоздает всю базу данных с использованием резервной копии,

созданной ранее. Резервная копия базы данных позволяет восстановить базу данных до состояния на момент создания данной резервной копии. Однако все единицы работы с момента резервного копирования до момента ошибки теряются (смотрите рис. 3).

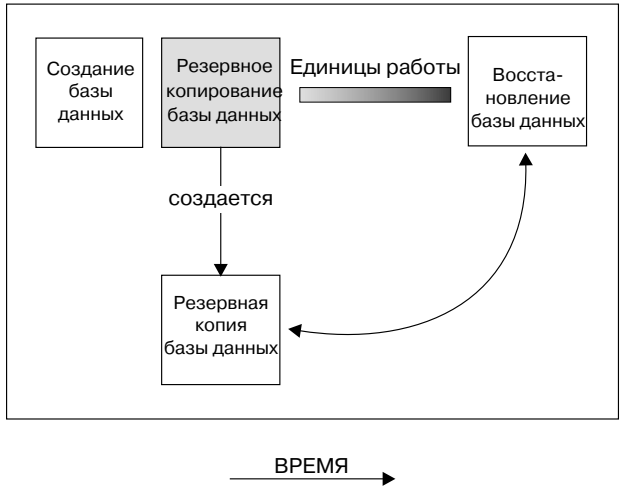


Рисунок 3. Восстановление версии. База данных восстанавливается из последней резервной копии, однако все единицы работы между моментом создания этой копии и моментом отказа теряются.

При использования 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 каждого раздела базы данных, выполняемое одновременно, называется *резервным копированием версии*.

Восстановление с повтором транзакций

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 “Что такое журналы восстановления” на стр. 33.) 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 с повтором транзакций после операции восстановления базы данных применяются записи транзакций журналов базы данных (смотрите рис. 4). 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. Если восстанавливается с повтором транзакций только один раздел базы данных, восстановление можно провести только до конца журналов. Восстановление до момента времени применяется ко *всем* разделам базы данных.

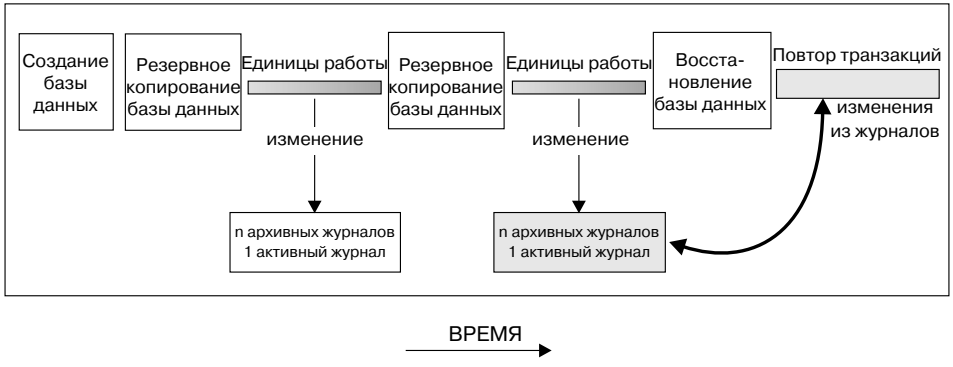


Рисунок 4. Восстановление базы данных с повтором транзакций. Для длительных транзакций может существовать несколько активных журналов.

- *Восстановление табличного пространства с повтором транзакций.* Если восстановление с повтором транзакций поддерживается базой данных, для ее табличных пространств также возможно резервное копирование, восстановление и повтор транзакций (смотрите раздел рис. 5 на стр. 28). Для выполнения восстановления и повтора транзакций табличного пространства требуется 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*).

Восстановление с повтором транзакций 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 раздел “Команда ROLLFORWARD DATABASE” на стр. 158).
- 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 восстановления базы данных. Например, при отключении питания табличное пространство может оказаться в состоянии отложенного повтора транзакций. 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 до конца журналов. 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.

  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.

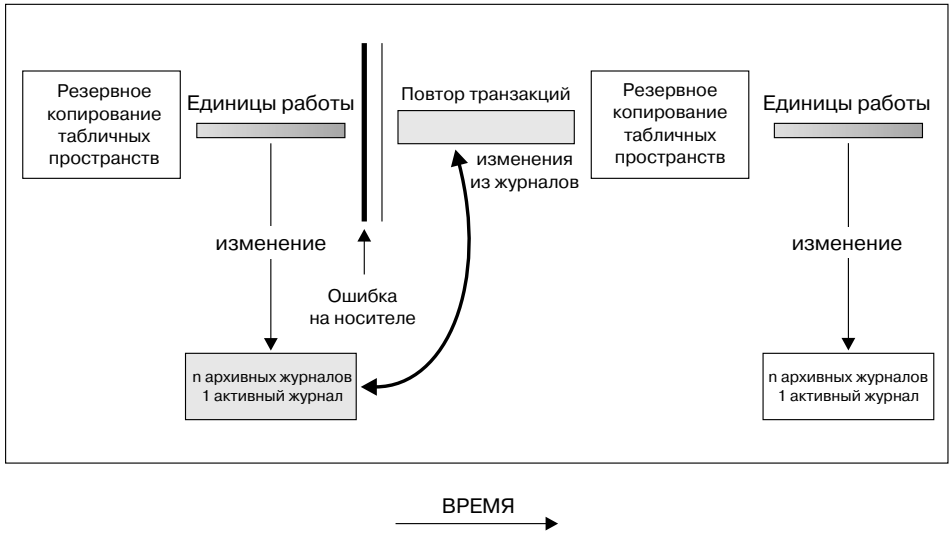


Рисунок 5. Восстановление с повтором транзакций табличного пространства. Для длительных транзакций может существовать несколько активных журналов.

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

Инкрементное резервное копирование и восстановление

По мере того, как размер баз данных, особенно хранилищ, продолжает расширяться до терабайтных и петабайтных масштабов, одновременно растут время и аппаратные ресурсы, требуемые для резервного копирования и восстановления этих баз данных. При работе с большими базами данных полное резервное копирование баз данных и табличных пространств не всегда оказывается лучшим подходом, поскольку требования к устройствам хранения для многочисленных копий таких баз данных оказываются непомерными. Рассмотрим следующие соображения:

- Когда меняется лишь малый процент данных в хранилище, должна быть возможность обходиться без резервного копирования всей базы данных.
- Добавление табличных пространств к существующим базам данных и последующее резервное копирование только табличного пространства рискованно, поскольку нет гарантии, что вне копируемых табличных пространств ничего за время между двумя копированиями не изменилось.

Теперь DB2 поддерживает инкрементное резервное копирование и восстановление (за исключением данных длинных полей и больших объектов). *Инкрементная резервная копия* - это резервная копия, содержащая лишь те страницы, которые были изменены после снятия предыдущей резервной копии.Помимо страниц с измененными данными и индексом, каждая инкрементная резервная копия содержит все метаданные первоначальной базы данных (такие как конфигурация базы данных, определения табличных пространств, хронологию базы данных и так далее), которые сохраняются при обычном полном резервном копировании.

Поддерживаются два типа инкрементного резервного копирования:

- *Инкрементный*. Инкрементная резервная копия - это копия всех данных базы данных, которые изменились со времени последней успешной операции полного резервного копирования. Ее называют также кумулятивной резервной копией, поскольку каждая следующая инкрементная резервная копия будет содержать информацию всех предыдущих. Предшественником инкрементной резервной копии всегда является последняя успешная полная резервная копия того же объекта.
- *Разностный*.Разностная (или инкрементная разностная) копия, - это копия всех данных базы данных, которые изменились со времени последней успешной операции резервного копирования (полного, инкрементного или типа дельта) рассматриваемого табличного пространства. Она называется также дифференциальной резервной копией. Предшественником разностной резервной копии является последняя успешная резервная копия со всеми табличными пространствами, входящими в данную разностную резервную копию.

Главное различие между инкрементной и разностной копиями заключается в их поведении при снятии последовательных резервных копий объекта, непрерывно менявшегося в течении какого-то времени. Каждая следующая инкрементная копия включает все содержимое предыдущей инкрементной копии плюс все новые или изменившиеся с момента изготовления предыдущей копии данные. Разностные резервные копии содержат только страницы, изменившиеся с момента изготовления предыдущей копии.

Допускается сочетание инкрементного резервного копирования базы данных и табличного пространства, как в оперативном режиме, так и в автономном. Тщательно планируйте стратегию резервного копирования, поскольку сочетание

инкрементного резервного копирования базы данных и табличного пространства приводит к тому, что предшественник резервной копии базы данных (или резервной копии нескольких табличных пространств) не всегда представляет собой единую копию, а может оказаться уникальным набором сделанных в разное время предыдущих резервных копий базы данных и табличных пространств.

Чтобы восстановить базу данных или табличное пространство до согласованного состояния, процесс восстановления должен начаться с согласованной резервной копии всего воссоздаваемого объекта (базы данных или табличного пространства)и должен затем использовать все соответствующие инкрементные копии в описанном ниже порядке (смотрите раздел “Восстановление из инкрементных резервных копий”).

Чтобы разрешить отслеживание изменений базы данных, DB2 поддерживает новый параметр конфигурации базы данных *trackmod*, который может иметь одно из двух допустимых значений:

- NO. Инкрементное резервное копирование в этой конфигурации не разрешено. Изменения страниц баз данных не отслеживаются и никак не регистрируются.
- YES. Инкрементное резервное копирование в этой конфигурации разрешено. Когда разрешается отслеживание изменений, новая конфигурация вступает в силу при первом успешном соединении с любой базой данных экземпляра. До первого инкрементного резервного копирования должно пройти полное резервное копирование базы данных.

Значение *trackmod* по умолчанию для существующих баз данных - `NO`; для новых баз данных - `YES`.

Для табличных пространств SMS уровень такого отслеживания - табличное пространство. Для табличных пространств DMS уровень отслеживания - экстент для страниц данных и страниц индексов, и табличное пространство - для других типов страниц.

Даже минимальное отслеживание изменений в базе данных может сказаться на производительности при выполнении транзакций, связанных с изменениями или вставкой данных.

Восстановление из инкрементных резервных копий

Операция восстановления из инкрементных резервных копий всегда содержит следующие шаги:

1. Определение инкрементной копии назначения.

   Сначала администратор базы данных должен найти последнюю копию, подлежащую восстановлению, и из утилиты восстановления DB2 запросить операцию инкрементного восстановления. Этот образ при инкрементном восстановлении называется целевым образом, поскольку он будет последним из восстановленных образов. Команда инкрементного восстановления для

этого образа может инициировать создание новой базы данных, конфигурация и определения табличных пространств которой будут взяты из этой целевой копии. Инкрементная целевая копия задается при помощи параметра TAKEN AT в команде RESTORE DATABASE.

2. Восстановление наиболее поздней полной базы данных или табличного пространства в качестве отправной точки, к которой можно применять последовательные инкрементные резервные копии.
3. Восстановление каждой из требуемых инкрементных резервных полных копий или копий табличного пространства в том порядке, в котором они создавались, начиная с исходной копии, восстановленной на шаге 2.
4. Повторение шага 3, пока не будет вторично прочитана копия назначения из шага 1. Всего в ходе операции инкрементного восстановления происходит два обращения к целевой копии. Во время первого обращения из копии считываются только начальные данные; никакие пользовательские данные не считываются. Полная копия считывается и обрабатывается только во время второго обращения.

   Два обращения к целевой копии при операции инкрементного восстановления нужны для того, чтобы убедиться в корректности изначально заданной хронологии, конфигурации базы данных и определений табличных пространств для базы данных, которая будет создана в ходе операции восстановления. В тех случаях, когда табличное пространство отбрасывалось после снятия первоначальной полной копии базы данных, данные табличного пространства для этой копии будут считываться из резервных копий, но игнорироваться в ходе инкрементного восстановления.

Чтобы восстановить набор инкрементных резервных копий, задайте опцию TAKEN AT *отметка_времени* в команде RESTORE DATABASE. Задайте отметку времени для последней из копий, которые нужно восстановить. Например:

```
db2 restore db sample incremental automatic taken at 20001228152133
```

При этом утилита восстановления DB2 автоматически выполнит все описанные далее шаги. На начальной фазе обработки будет прочитан образ резервной копии с отметкой времени 20001228152133, и утилита восстановления проверяет, что база данных, ее хронология и определения табличного пространства существуют и допустимы.

На второй фазе обработки по хронологии базы данных строится цепочка образов резервных копий, необходимых для требуемой операции восстановления. Если же по какой-либо причине это невозможно, и DB2 не может построить полную цепочку необходимых образов, операция восстановления прекращается и возвращается сообщение об ошибке. В таком случае автоматическое восстановление невозможно, и вам надо выполнить восстановление вручную.

Примечание: Настоятельно рекомендуется не использовать опцию FORCE в команде PRUNE HISTORY. По умолчанию эта команда не удаляет те записи хронологии, которые могут понадобиться для восстановления с самого свежего полного образа резервной копии, однако с опцией FORCE можно удалить записи, необходимые для автоматического восстановления.

Если хронология базы данных недоступна, вы можете выполнить инкрементное восстановление вручную по шагам, описанным в начале этого раздела. Например:

```
1. db2 restore database sample incremental taken at <отметка>

   где:
     <отметка> указывает на последнюю инкрементную резервную копию, которую
     нужно восстановить

2. db2 restore database sample incremental taken at <отметка1>

   где:
     <отметка1> указывает на начальную полную копию базы данных (или табличного
     пространства)

3. db2 restore database sample incremental taken at <отметкаX>

   где:
     <отметкаX> указывает на каждую из инкрементных резервных копий в порядке их
     создания

4. Повторить шаг 3, восстанавливая все инкрементные резервные копии вплоть до копии
<отметка>
```

Если предпринимается попытка операции восстановления базы данных и создавались инкрементные резервные образы табличного пространства, образы табличного пространства должны восстанавливаться в хронологическом порядке в соответствии их отметкам времени резервного копирования.

Ограничения автоматического инкрементного восстановления

1. Если вы переименовали табличное пространство после снятия резервной копии, которую хотите восстановить, и пытаетесь выполнить восстановление на уровне табличного пространства и новым именем, требуемая цепочка образов не будет сгенерирована по хронологии базы данных и произойдет ошибка.

   Пример:

   ```
   db2 backup db sample -> <ts1>
   db2 backup db sample incremental -> <ts2>
   db2 rename tablespace from userspace1 to t1
   db2 restore db sample tablespace ('t1') incremental automatic taken at <ts2>
   ```

   Предлагаемый обходной прием: используйте ручное восстановление.

2. Если вы отбрасываете базу данных, ее хронология будет удалена. Если затем восстановить отброшенную базу данных, ее хронология восстанавливается

на момент снятия восстанавливаемой резервной копии, все последующие записи хронологии будут утеряны. Если вы затем попытаетесь выполнить автоматическое инкрементное восстановление, для которого потребуются утерянные записи хронологии, утилита RESTORE построит неправильную цепочку резервных копий и будет получена ошибка ″out of sequence″.

Пример:

```
db2 backup db sample -> <ts1>
db2 backup db sample incremental -> <ts2>
db2 backup db sample incremental delta -> <ts3>
db2 backup db sample incremental delta -> <ts4>
db2 drop db sample
db2 restore db sample incremental automatic taken at <ts2>
db2 restore db sample incremental automatic taken at <ts4>
```

Рекомендуемые обходные приемы:

- Используйте ручное восстановление.
- Сначала восстановите файл хронологии из образа <ts4>, затем используйте автоматическое инкрементное восстановление.

Что такое журналы восстановления

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 три типа регистрации в журналах: *циклическая*, *регистрация захвата* и *архивная*, каждая из которых обеспечивает свой уровень возможностей восстановления.

- *Циклическая* запись выбирается по умолчанию, когда создается новая база данных. (Значение параметра конфигурации базы данных *logretain* - NO.) 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 для транзакций, выполненных после снятия последней полной резервной копии. 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. 6 на стр. 34 показано, что когда активна циклическая запись, активный журнал использует файлы журналов в циклическом порядке.

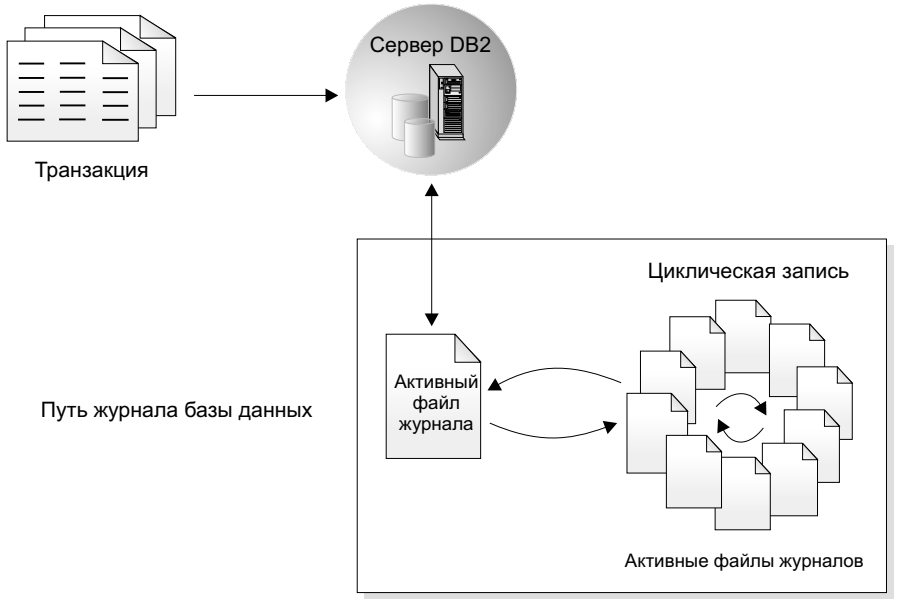


Рисунок 6. Циклическая запись журналов

Активные журналы используются во время восстановления после сбоя, чтобы не 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 отказа для изменений, записанных в журналы, но не еще принятых, 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. Активные журналы находятся в каталоге пути журналов базы данных.

- Регистрацию *захвата* можно сконфигурировать, задав для параметра конфигурации базы данных *logretain* значение CAPTURE. Регистрация захвата используется для обработки репликации. Файлы журнала сохраняются до завершения процесса репликации, после чего автоматически удаляются. Все утилиты DB2 работают с таким способом записи так же, как и с циклической записью;то есть при ней не разрешены ни оперативное снятие резервных копий, ни операции резервного копирования и восстановления на уровне табличного пространства, ни операции повтора транзакций, а операция загрузки выполняется с опцией RECOVERY NO и не оставляет табличные пространства в состоянии отложенного резервного копирования.
- *Архивная* регистрация используется специально для восстановления с повтором транзакций. Ее можно сконфигурировать, задав для параметра конфигурации базы данных *logretain* значение RECOVERY. Архивные журналы бывают двух типов:

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. 7).

автономные архивные журналы

Архивный журнал считается *автономным*, если он больше не находится в каталоге пути журналов базы данных (смотрите рис. 8 на стр. 36). Если используется обработчик пользователя, архивные журналы можно хранить и не в каталоге пути журналов базы данных. (Дополнительную информацию смотрите в разделе “Приложение H. Обработчик пользователя для восстановления баз данных” на стр. 471.)

Рисунок 7. Оперативные архивные журналы

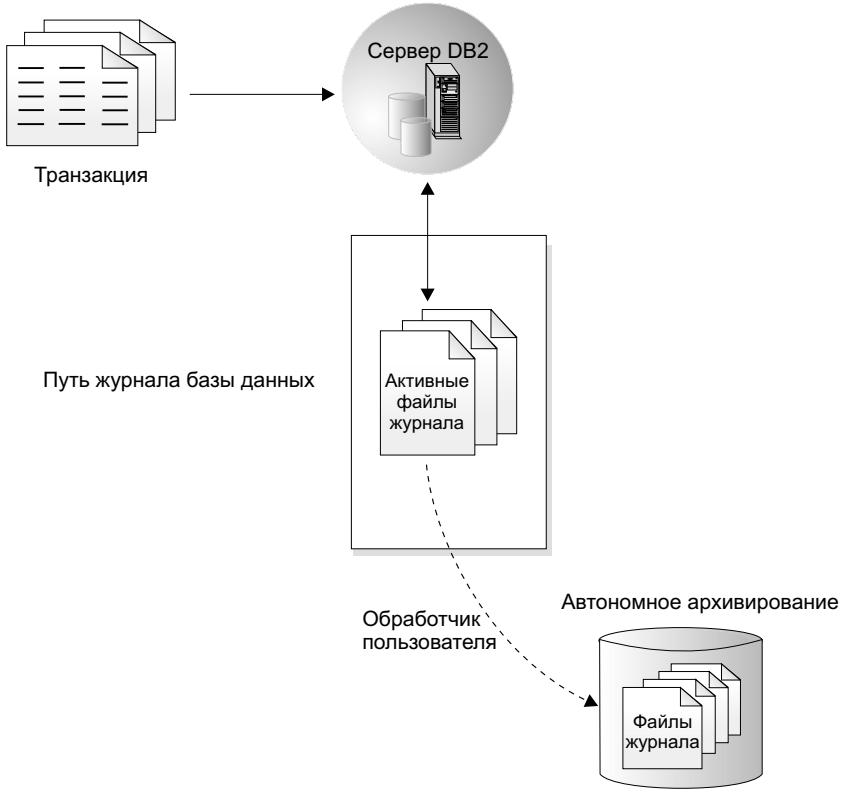


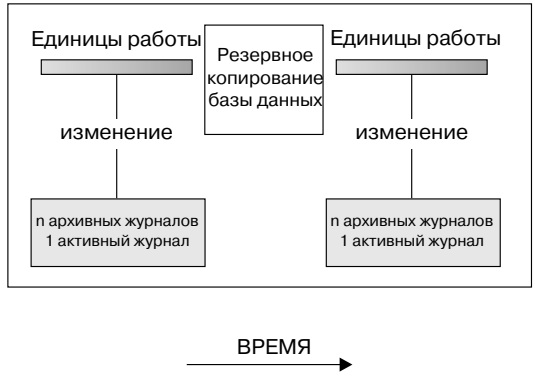
Рисунок 8. Автономные архивные журналы

При восстановлении с повтором транзакций для воссоздания базы данных до окончания журналов или до указанного момента времени могут использоваться и архивные, и активные журналы. Утилита 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. Для этого журналы должны быть заархивированы и к моменту восстановления базы данных сделаны доступными. После завершения оперативного резервного

копирования DB2 принудительно закрывает активный в данный момент журнал, и он архивируется. В результате у резервной копии будет полный комплект архивных журналов для ее восстановления.



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

Рисунок 9. Активные и архивные журналы базы данных при восстановлении с повтором транзакций. Для длительных транзакций может существовать несколько активных журналов.

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 руководстве *Administration Guide: Performance*.

Чтобы определить, какие *экстенты* журналов 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 руководстве *Administration Guide: Performance*.

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 того журнала, который был стерт.
2. Если вы опасаетесь, что ваши активные журналы могут быть повреждены (в результате аварии диска), вы можете обдумать использование зеркальной

копии - либо на уровне операционной системы, задав зеркальное копирование диска, на который записываются журналы, либо на уровне DB2 при помощи переменной реестра NEWLOGPATH2.

Создание зеркальной копии журнала

Теперь DB2 поддерживает создание зеркальной копии журнала на уровне базы данных. Создание зеркальной копии файлов журнала помогает защитить базу данных от таких опасностей, как:

- Случайное удаление активного журнала
- Разрушение данных из-за неисправности оборудования

Если вы опасаетесь, что ваши активные журналы могут быть повреждены (в результате аварии диска), вы можете при помощи новой переменной реестра DB2 - NEWLOGPATH2, - задать вторичный путь, чтобы база данных работала с копиями активного журнала, создавая зеркальные тома для записи журналов.

Переменная реестра NEWLOGPATH2 позволяет базе данных записывать идентичную вторую копию файлов журнала в другой путь. Рекомендуется разместить вторичный путь на другом физическом диске (желательно с другим контроллером диска). Тогда контроллер диска не будет уязвимой точкой.

Примечание: Поскольку Windows NT не позволяют “монтировать” устройство с произвольным именем пути, невозможно задать вторичный путь на отдельном устройстве.

NEWLOGPATH2 может быть включена (задана равной 1) или выключена (задана равной 0). Значение по умолчанию - ноль. Если этой переменной присвоена 1, имя вторичного пути равно текущему значению переменной LOGPATH плюс символ 2. Например, в среде SMP, если LOGPATH равен `/u/dbuser/sqllogdir/logpath`, вторичный путь журнала будет `/u/dbuser/sqllogdir/logpath2`. В среде MPP, если LOGPATH равен `/u/dbuser/sqllogdir/logpath`, DB2 присоединит к пути указатель узла, и в качестве первичного пути журнала будет использовать `/u/dbuser/sqllogdir/logpath/NODE0000`. В таком случае вторичным путем журнала будет`/u/dbuser/sqllogdir/logpath2/NODE0000`.

При первом включении NEWLOGPATH2 вторичный путь журнала не используется, пока не будет завершен текущий файл журнала при следующем запуске базы данных. Это аналогично тому, как сейчас используется LOGPATH и NEWLOGPATH.

Если возникает ошибка при записи в первичный или вторичный путь журнала, база данных пометит отказавший путь как “плохой”, запишет сообщение в файл `db2diag.log` и будет помещать следующие записи журнала только в оставшийся “хороший” путь журнала. DB2 не будет пытаться еще раз использовать “плохой” путь, пока не будет завершен текущий файл журнала. Когда DB2 потребуется

открыть следующий файл журнала, она проверит, допустим ли этот путь, и если да, начнет использовать его. Если нет, DB2 не будет пытаться еще раз использовать путь до первого обращения к следующему файлу журнала. Попыток синхронизировать пути журналов не производиться, но DB2 сохраняет информацию о происходящих ошибках доступа, чтобы использовать правильные пути при архивации файлов журнала. Если сбой произойдет при записи в оставшийся “хороший” путь, база данных прервет работу.

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 таблицы 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. (Для этого таблица должна быть создана с опцией NOT LOGGED INITIALLY.)

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. Подробности смотрите в разделе “Параллелизм” руководства *Administration Guide: Performance*.

Дополнительную информацию о создании таблиц смотрите в руководстве *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 все 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*.

Параметры конфигурации для записи в журнал базы данных

В файле конфигурации базы данных содержатся параметры, относящиеся восстановлению с повтором транзакций. Значения параметров по умолчанию не поддерживают такой тип восстановления, поэтому, если вы планируете его использование, некоторые из этих значений необходимо изменить. Дополнительную информацию о конфигурировании DB2 Universal Database смотрите в книге *Administration Guide: Performance*.

Первичные файлы журнала (logprimary)

Этот параметр показывает число создаваемых первичных файлов журнала.

Для первичного файла журнала вне зависимости от его заполненности требуется одно и то же дисковое пространство. Поэтому, если сконфигурировать больше файлов журнала, чем необходимо, дисковое пространство будет использоваться непроизводительно. Если

сконфигурировать слишком мало файлов журнала, можно столкнуться с состоянием заполненности журнала. При выборе числа конфигурируемых файлов журнала следует учитывать, какой размер вы задаете для каждого из файлов, и может ли ваша прикладная программа обрабатывать состояние заполнения журнала.

Если вы включаете восстановление с повтором транзакций для существующей базы данных, замените число первичных файлов журнала на сумму числа первичных и вторичных файлов плюс 1. В базе данных, для которой разрешено восстановление с повтором транзакций, в журнал записывается дополнительная информация для полей LONG VARCHAR и LOB.

Общее ограничение на размер файлов журнала составляет 32 Гбайта. Это означает, что число файлов журнала (*logprimary* + *logsecond*), умноженное на размер каждого файла в байтах (*logfilsiz* * 4096), не должно превышать 32 Гбайт.

Дополнительную информацию об этом параметре конфигурации смотрите в руководстве *Administration Guide: Performance*.

Вторичные файлы журнала (logsecond)

Этот параметр задает число вторичных файлов журнала, которые создаются и используются для файлов журнала восстановления при необходимости.

Если первичные файлы журнала заполняются, по мере необходимости по одному выделяются вторичные файлы журнала (с размером, указанным *logfilsiz*) вплоть до максимального числа, задаваемого этим параметром. Если требуется больше вторичных файлов журнала, чем задано, возвращается ошибка и работа с базой данных прекращается.

Дополнительную информацию об этом параметре конфигурации смотрите в руководстве *Administration Guide: Performance*.

Размер журнала (logfilsiz)

Этот параметр задает размер каждого сконфигурированного журнала в страницах по 4 Кбайта.

Всего можно сконфигурировать до 32 Гбайт пространства активного журнала. Эта цифра получается в результате умножения максимального значения*logfilsiz*, то есть 65535, на максимальное значение (*logprimary* + *logsecond*), то есть 128. Таким образом, ((*logprimary* + *logsecond*) * *logfilsiz*) < (32 Гбайта / 4096).

Размер файлов журнала непосредственно влияет на производительность. Если база данных сконфигурирована для сохранения файлов журнала, при заполнении одного из файлов выдается требование на размещение и инициализацию нового файла. Увеличение размера файла журнала уменьшает число запросов на выделение и

инициализацию новых файлов журнала. (Однако не забывайте, что при большом размере файлов журнала требуется больше времени на форматирование новых файлов). Форматирование новых файлов журнала прозрачно для прикладных программ, связанных с базой данных, поэтому на производительность базы данных не влияет.

Предположим, что у вас есть прикладная программа, поддерживающая базу данных в открытом состоянии для сокращения расходов процессорного времени на ее открытие (смотрите “Повышение производительности восстановления” на стр. 59); тогда размер файла журнала должен определяться количеством времени, необходимого для автономного снятия копий архивированных файлов журнала.

Скорость передачи данных устройства, используемого для хранения автономных архивированных файлов журнала, и программного обеспечения, используемого для изготовления копий, должны, как минимум, равняться средней скорости записи данных в файлы журнала менеджером баз данных. Если эта скорость передачи меньше скорости генерирования новых данных журнала, при достаточно продолжительном времени записи в журнал, определяемом количеством свободного места на диске, диск может полностью заполниться. В этом случае обработка базы данных останавливается.

Особенно критична скорость передачи данных при использовании магнитной ленты или оптических носителей. (Об использовании различных сред для хранения журналов смотрите раздел “Приложение H. Обработчик пользователя для восстановления баз данных” на стр. 471.) На некоторых ленточных устройствах для записи файла любого размера требуется одинаковое время. Необходимо знать производительность архивирующего устройства.

Ленточные устройства обладают и другими особенностями. Важна частота требований на архивирование. Например, если время для выполнения любой операции копирования равно пяти минутам, файла журнала должен вмещать данные за пять минут работы при пиковой нагрузке. У ленточного устройства могут быть конструктивные ограничения на число операций в день. Все эти факторы следует учитывать при определении размера файла журнала.

При определении размера файлов журнала также следует учитывать снижение потерь этих файлов. Файл журнала архивируется целиком. При использовании одного большого файла журнала увеличивается время между архивированиями. При ошибке носителя, на котором находятся файлы журнала, информация о некоторых транзакциях, возможно, будет утеряна. Уменьшение размера файла журнала увеличивает частоту архивирования, но может уменьшить потери информации при ошибках носителя, поскольку можно будет использовать маленькие файлы журнала, предшествующие утерянному.

Буфер журнала (logbufsz)

Этот параметр позволяет указывать величину совместно используемой памяти базы данных, отводимой под буфер записей журнала до их помещения на диск. Записи журнала записываются на диск в любой из следующих ситуаций:

- Принимается транзакция
- Заполняется буфер журнала
- Происходят некоторые внутренние события менеджера баз данных.

Увеличение размера буфера журнала повышает эффективность операций ввода/вывода при регистрации, поскольку записи журнала записываются на диск реже и большими порциями.

Число принятий для группировки (mincommit)

Этот параметр позволяет откладывать помещение записей журнала на диск до выполнения минимального числа принятий. Такая отложенная запись может помочь уменьшить излишнюю нагрузку на менеджер баз данных, связанную с записью в журнал, и таким образом улучшить производительность, когда несколько прикладных программ требуют много принятий за очень короткий отрезок времени.

Группировка принятий происходит только тогда, когда значение этого параметра больше 1 и число связанных с базой данных прикладных программ больше значения этого параметра. При группировке принятий выполнение требований прикладных программ на принятия откладывается либо до истечения одной секунды, либо до того времени, когда число требований на принятие сравняется со значением этого параметра (если это произойдет раньше).

Новый путь к журналу (newlogpath)

Файлы журнала базы данных изначально создаются в подкаталоге SQLOGDIR каталога, в котором расположена база данных. Можно изменить место размещения активных и будущих архивных файлов журнала, изменив значение этого параметра конфигурации так, чтобы он указывал на новый каталог или устройство. Если база данных сконфигурирована для восстановления с повтором транзакций, архивные файлы журнала, хранящиеся в исходном каталоге, не перемещаются в новое положение.

Поскольку можно изменять положение файлов журнала, те из них, которые необходимы для восстановления с повтором транзакций, могут находиться в разных каталогах или на разных устройствах. Значение этого параметра конфигурации можно изменять во время повтора транзакций для обеспечения доступа к файлам журнала, находящимся в различных положениях.

Необходимо следить за расположением архивных файлов.

Измененное значение не применяется до тех пор, пока база данных не перейдет в согласованное состояние. Состояние базы данных отражает параметр конфигурации*database_consistent*. Дополнительную информацию об этом параметре конфигурации смотрите в руководстве *Administration Guide: Performance*. Информацию о роли журналов баз данных для баз, которые находятся в несогласованном состоянии, смотрите в разделе “Управление файлами журнала”.

Сохранение журнала (logretain)

Если параметр logretain имеет значение RECOVERY, архивные журналы сохраняются в каталоге журналов базы данных, и база рассматривается как восстановимая, то есть для нее разрешен повтор транзакций.

Если параметр logretain имеет значение CAPTURE, программа репликации capture после выполнения захвата вызывает команду PRUNE LOGFILE для удаления файлов журнала. Если вы собираетесь выполнять для базы данных восстановление с повтором транзакций, не следует задавать для параметра *logretain* значение CAPTURE.

Обработчик пользователя (userexit)

Этот параметр задает, что менеджер баз данных вызывает программу обработчика пользователя для архивирования и восстановления файлов журнала. Файлы журнала при этом архивируются не в каталоге активных файлов журнала. Если параметр *userexit* имеет значение ON, повтор транзакций разрешен. Информацию об обработчиках пользователя смотрите в разделе “Приложение H. Обработчик пользователя для восстановления баз данных” на стр. 471.

Управление файлами журнала

При управлении журналами базы данных учтите следующие факторы:

- Нумерация архивированных файлов журнала начинается с S0000000.LOG и продолжается до S9999999.LOG, что допускает потенциально 10 миллионов файлов журнала. Менеджер баз данных возвращается к файлу S0000000.LOG, если:
  - Файл конфигурации базы данных изменяется, чтобы разрешить повтор транзакций.
  - Файл конфигурации базы данных изменяется, чтобы *запретить* повтор транзакций
  - Использован файл S9999999.LOG.

  DB2 повторно использует имена файлов журнала после восстановления базы данных (как с повтором транзакций, так и без). Менеджер баз данных подразумевает, что неправильный файл журнала не участвует в восстановлении с повтором транзакций, но он не может определить положение необходимого файла журнала. Вам необходимо проверить, что для восстановления с повтором транзакций доступны правильные файлы журнала.

Когда восстановление с повтором транзакций заканчивается успешно, последний файл журнала, использованный для повтора транзакций, усекается, и запись в журнал начинается со следующего последовательного номера. Повторно используются все файлы журнала в каталоге журнала с последовательным номером, большим, чем у последнего файла, использованного для восстановления с повтором транзакций. Все записи в усеченном журнале после точки усечения заполняются нулями. Прежде чем вызывать утилиту повтора транзакций, следует сделать копию журналов. (Можно вызвать программу обработчика пользователя, чтобы скопировать журналы в другое место. Информацию об обработчиках пользователя смотрите в разделе “Приложение H. Обработчик пользователя для восстановления баз данных” на стр. 471.)

- Если база данных не активирована (командой ACTIVATE DATABASE), DB2 усекает текущий файл журнала, когда все программы отсоединяются от этой базы данных. При следующем соединении программы с базой данных DB2 начинает запись в новый файл журнала. Если в вашей системе образуется много мелких файлов журнала, обдумайте использование команды ACTIVATE DATABASE. Это сэкономит не только расходы на активацию базы данных при подключении программ, но но и расходы на размещение большого файла журнала с последующим его усечением и размещением нового большого файла.
- Архивный журнал может быть связан с двумя или более различными *последовательностями журналов* для базы данных из-за повторного использования имен файлов журнала (смотрите раздел рис. 10 на стр. 46). Например, если вы хотите восстановить резервную копию Backup 2, существует две возможных используемых последовательности. Если при восстановлении всей базы данных вы произвели повтор транзакций до определенного момента времени с остановкой ранее конца журналов, вы создаете новую последовательность файлов журналов. Эти две последовательности файлов журнала нельзя объединять. Если у вас есть оперативная резервная копия, которая затрагивает первую последовательность файлов журнала, именно эту последовательность и надо использовать при восстановлении с повтором транзакций.

  Если после восстановления вы создали новую последовательность файлов журнала, все резервные копии табличных пространств в старой последовательности будут недействительными. Утилита восстановления не сможет распознать резервную копию табличного пространства в старой последовательности, если операция восстановления базы данных немедленно следует за операцией восстановления табличного пространства. До момента, когда для базы данных реально будет выполняться повтор транзакций, последовательность, которая при этом будет использована, неизвестна. Если табличное пространство находится в старой последовательности, оно должно быть “захвачено” при операции повтора транзакций для табличного пространства. Операция восстановления, использующая неправильную резервную копию, может завершиться успешно, однако операция повтора

транзакций для табличного пространства, завершится неудачно, и это табличное пространство останется в состоянии отложенного повтора транзакций.

Предположим, например, что операция снятия резервной копии табличного пространства,Backup 3, завершена между S0000013.LOG и S0000014.LOG верхней последовательности журналов (смотрите рис. 10). Если вы хотите восстановить с повтором транзакций базу с использованием резервной копии уровня базы данных, Backup 2, вам надо выполнить повтор транзакций до файла S0000012.LOG. После этого можно продолжать повтор, используя либо верхнюю, либо нижнюю (новую) последовательность. Если выполнять повтор транзакций с использованием нижней последовательности, вы не сможете использовать резервную копию уровня табличного пространства, Backup 3, для восстановления табличного пространства и повтора транзакций.

Чтобы завершить операцию повтора транзакций до конца журналов с использованием резервной копии уровня табличного пространства, Backup 3, надо восстановить резервную копию уровня базы данных, Backup 2, а затем выполнить повтор транзакций по верхней последовательности. Когда резервная копия уровня табличного пространства, Backup 3, будет восстановлена, можно начать операцию повтора транзакций до конца журналов.

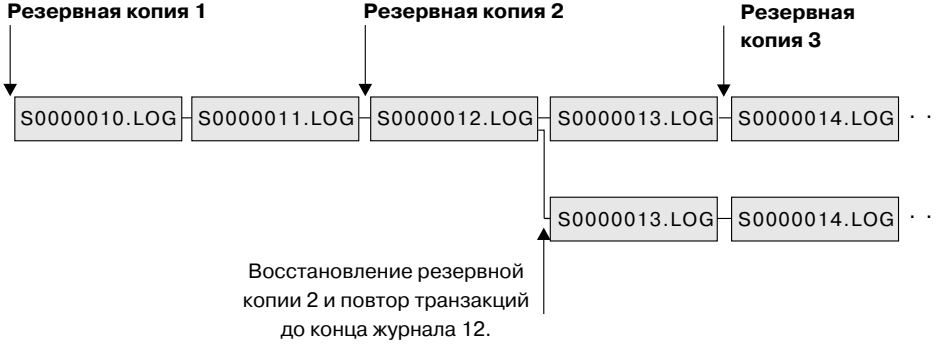


Рисунок 10. Повторное использование имен файлов журналов

Управление файлами журнала с программой обработчика пользователя

Вызов программы обработчика пользователя для архивирования и восстановления файлов журнала происходит так:

- Параметр файла конфигурации базы данных *userexit* задает, будет ли менеджер баз данных вызывать программу обработчика пользователя для архивирования файлов или для восстановления файлов журнала во время восстановления баз данных с повтором транзакций. Требование восстановить файл журнала выдается, когда утилите повтора транзакций необходим файл журнала, который не удалось найти в каталоге журнала.

Примечание: В Windows NT обработчик пользователя REXX использовать для архивирования файлов журнала нельзя.

- Во время архивирования файл журнала передается обработчику пользователя, когда он полон, даже если этот файл еще активен и требуется для обычной обработки. Это позволяет как можно быстрее переместить копии данных с временного носителя. Файл журнала, переданный обработчику пользователя, сохраняется в каталоге журнала, пока снова не потребуется для обычной обработки. При этом дисковое пространство используется повторно.
- DB2 при запуске обработчика пользователя для архивирования файла журнала открывает этот файл в режиме чтения. На некоторых платформах это не позволяет программе обработчика пользователя удалить файл журнала. На других платформах, например, AIX, процессам, в том числе программе обработчика пользователя, разрешается удалять файлы журнала. Программа обработчика пользователя никогда не должна удалять файл журнала после его архивирования, поскольку этот файл может быть еще активен и необходим для восстановления после отказа. DB2 управляет использованием дискового пространства при архивации файлов журнала.
- Когда файл журнала заархивирован и неактивен, DB2 не удаляет его, а переименовывает его, как следующий файл журнала, если такой файл необходим. Это повышает производительность, поскольку при создании нового файла журнала (вместо переименования файла) все страницы переписываются для сохранения места на диске. Нужные страницы лучше использовать повторно, чем освобождать и снова занимать ими место на диске.
- DB2 *не* вызывает обработчик пользователя ни при восстановлении при отказе, ни при откате.
- Программа обработчика пользователя не гарантирует восстановление с повтором транзакций в точке отказа, а лишь пытается уменьшить влияние ошибки. Файлы журнала ставятся на очередь в подпрограмму обработчика пользователя, когда они заполнены. Если заполняется диск, содержавший ошибку журнала перед файлом журнала, данные в таком файле журнала теряются. Кроме того, поскольку файлы для архивирования ставятся в очередь, на диске может возникнуть ошибка перед тем, как будут скопированы все файлы, что приведет к потере всех файлов журнала в очереди.
- Сконфигурированный размер каждого отдельного файла журнала имеет непосредственное отношение к обработчику пользователя. Если файлы журнала слишком велики, при ошибках на диске можно потерять большое количество данных. Если база данных сконфигурирована на использование небольших файлов, данные будут передаваться в подпрограмму обработчика пользователя чаще.

  Однако при записи данных на медленное устройство, например на ленту, лучше бы использовать более длинные файлы, чтобы предотвратить рост

очереди. После заполнения очереди требования на архивацию и восстановление обрабатываться не будут. Обработка возобновляется, когда в очереди освободится место. Необработанные требования не возвращаются в очередь автоматически.

- Требование на архивирование программой обработчика пользователя удовлетворяется, только если сконфигурирован параметр *userexit*, при каждом заполнении активного файла журнала. Возможна ситуация, что при последнем отсоединении от базы данных активный файл журнала оказался не полон, тогда программа обработчика пользователя вызывается для частично заполненного файла журнала.

  Примечание: Чтобы освободить неиспользуемое пространство журнала, файл журнала перед архивированием сокращается.
- Копию журнала следует создавать на другом физическом устройстве, чтобы для восстановления с повтором транзакций можно было воспользоваться автономным файлом журнала, если на устройстве с основным файлом журнала происходит ошибка носителя. Это устройство не должно быть тем же, где находятся файлы данных базы данных.
- В некоторых случаях, если база данных закроется до получения от программы обработчика пользователя положительного ответа на требование архивирования, менеджер баз данных пошлет другое требование при открытии базы данных. Таким образом, файл журнала может быть заархивирован несколько раз.
- Если программа обработчика пользователя получает требование на архивирование файла, который не существует (например, на него было несколько требований, и он был удален после первого успешного архивирования), или требование на восстановление файла, который не существует (так как он расположен в другом каталоге, или достигнут конец журналов), программа игнорирует это требование и передает успешный код возврата.
- Программа обработчика пользователя должна учитывать, что после восстановления до определенного момента времени могут существовать разные файлы журнала с одним именем; ее необходимо написать так, чтобы она сохраняла такие файлы журнала и связывала с ними верные пути восстановления (смотрите раздел “Управление файлами журнала” на стр. 44).
- Если несколько баз данных одновременно используют одно устройство, и одна из операций начинает повтор транзакций, файл журнала, необходимый для восстановления с повтором транзакций, может не существовать на носителе, установленном в настоящий момент в устройстве. Могут иметь место две ситуации:
  - Если программа обработчика пользователя возвращает менеджеру баз данных нулевой (успешный) код возврата, а требуемый файл журнала не был получен, менеджер баз данных считает, что повтор транзакций

выполняется до окончания журналов, и операция по повтору транзакций останавливается. Однако обработка повтора транзакций не будет выполнена до окончания журналов.

– Если возвращается ненулевой код возврата, база данных перейдет в состояние отложенного повтора транзакций, и вы должны либо продолжить, либо остановить обработку повтора транзакций.

Каждую из этих ситуаций можно предотвратить, убедившись, что на узле, который вызывает программу обработчика пользователя, во время операции по повтору транзакций не существует других открытых баз данных; либо написав программу обработчика пользователя, которая будет обрабатывать данную ситуацию.

Запрет транзакций при переполнении каталога журнала

Новая переменная реестра DB2, DB2_BLOCK_ON_LOG_DISK_FULL, позволяет предотвратить генерирование сообщений об ошибках заполнения диска (″disk full″), когда DB2 не может создать новый файл журнала по действующему пути журналов.

Вместо этого DB2 будет пытаться создавать файл журнала каждые 5 минут, пока операция не завершится успешно. Если в конфигурации базы данных для параметра *userexit* задано значение`ON`, DB2 проверяет также завершение архивирования файлов журнала. Если неактивный файл журнала успешно архивирован, DB2 может переименовать этот неактивный файл, дав ему имя нового файла журнала, и продолжить работу. После каждой попытки DB2 записывает сообщение в файл `db2diag.log`. Единственный способ убедиться, что программа зависает из-за переполнения диска - это следить за файлом `db2diag.log`.

Пока файл журнала не будет создан успешно, ни одна пользовательская программа, пытающаяся изменить табличные данные, не сможет осуществить принятие транзакций. На запросы только для чтения нельзя воздействовать непосредственно, однако если запросу нужен доступ к данным, которые блокируются требованием изменения или к странице данных которая исправляется в пуле буферов изменяющей программой, запросы только для чтения, скорее всего, также зависнут.

Архивирование журнала по требованию

Теперь DB2 поддерживает закрытие (и, если включена опция обработчика пользователя, то и архивирование) активного журнала восстановимой базы данных в любое время. Это дает возможность собрать полный комплект файлов журнала до известного момента, и затем использовать эти файлы журнала для изменения резервной базы данных.

Вы можете инициировать архивирование журнала по требованию, введя команду ARCHIVE LOG или вызвав API **db2ArchiveLog**; они описаны

соответственно в разделах “ARCHIVE LOG” на стр. 340 и “db2ArchiveLog - API архивирования активного журнала” на стр. 354.

Использование непосредственных устройств для журналов

Для записи журналов баз данных можно использовать непосредственное устройство. У этого решения есть свои достоинства и недостатки.

- Достоинства:
  - К системе можно подключить более 26 физических дисководов.
  - Длина пути ввода/вывода для файла меньше. Это может повысить производительность в вашей системе. Следует провести контрольные тесты производительности, чтобы понять, есть ли значимый выигрыш для вашей рабочей нагрузке.
- Недостатки:
  - Устройство не смогут использовать другие прикладные программы; оно *полностью* должно быть назначено DB2.
  - Устройство не сможет работать с утилитой операционной системы или независимого производителя для снятия резервных копий или копирования с этого устройства.
  - Можно с легкостью стереть файловую систему на существующем диске, если указать неверный номер диска.

Непосредственный журнал можно сконфигурировать посредством параметра конфигурации базы данных *newlogpath*. Пример синтаксиса, используемого для указания непосредственного устройства, смотрите в разделе “Непосредственный ввод/вывод” книги *Administration Guide: Implementation*. Перед работой с непосредственными устройствами ознакомьтесь с перечисленными выше их преимуществами и недостатками, а также с дополнительными замечаниями ниже:

- Допустимо только одно устройство. Это устройство можно определить на нескольких дисках на уровне операционной системы. DB2 произведет вызов операционной системы для определения размера устройства в страницах по 4 Кбайта.

  Если вы используете несколько дисков, это приведет к образованию большего по размеру устройства и к параллельной записи, что может увеличить производительность за счет ускорения ввода/вывода.
- DB2 будет пытаться производить запись в последнюю 4-Кбайтную страницу устройства. Если размер устройства больше 2 Гбайт, в операционных системах, которые не поддерживают такие устройства, попытка записи на последнюю страницу закончится неудачно. В этом случае DB2 попробует использовать все страницы до поддерживаемого предельного значения.

  Информация о размере этого устройства используется для указания размера устройства (в страницах по 4 Кбайта), доступного для DB2 через поддержку

операционной системы. Дисковое пространство, в которое DB2 может производить запись, обозначается как *доступный-размер-устройства*.

Первая страница устройства размером 4 Кбайта DB2 не используется (обычно это пространство используется операционной системой для других целей). Это означает, что общее пространство, доступное DB2, определяется по формуле *размер-устройства* = *доступный-размер-устройства - 1*.

- Параметр *logsecond* не используется. DB2 не будет размещать вторичные файлы журналов. Размер пространства активного журнала представляет собой число страниц по 4 Кбайта, получающееся в результате умножения *logprimary* на *logfilsiz*.
- Записи журнала группируются в экстенты журнала, каждый из которых имеет размер файла журнала (*logfilsiz*) в страницах по 4 Кбайта. Экстенты журнала помещаются на непосредственное устройство один за другим. В каждом экстенте также есть две дополнительные страницы для заголовка экстента. Это означает, что *число доступных экстентов журнала*, поддерживаемых устройством, равно *размер-устройства* / (*logfilsiz* + 2)
- Устройство должно быть достаточно большим для поддержки пространства активного журнала. Это значит, что *число доступных экстентов журнала* должно быть больше или равно значению, указанному для параметра конфигурации *logprimary*.
- Если вы используете циклическую запись в журнал, параметр конфигурации *logprimary* будет определять число экстентов журнала, записанных на устройство. Это может привести к появлению на этом устройстве неиспользуемого пространства.
- Если вы используете сохранение журнала (*logretain*) без программы обработчика пользователя, после полного использования *числа доступных экстентов журнала* все операции, приводящие к изменениям, будут вызывать ошибку переполнения журнала. В это время следует остановить базу данных и сделать ее резервную копию в автономном режиме для обеспечения восстановимости. После резервного копирования базы данных записи журнала, сделанные на устройство, теряются. Это означает, что нельзя использовать более раннюю резервную копию базы данных для восстановления базы данных и последующего повтора транзакций. Если сделать резервную копию базы данных до полного использования *числа доступных экстентов журнала*, базу данных можно восстанавливать из резервной копии и производить для нее повтор транзакций.
- Если вы используете сохранение журнала (*logretain*) с программой обработчика пользователя, эта программа вызывается для каждого экстента журнала, как только он заполняется записями журнала. Программа обработчика пользователя должна быть в состоянии выполнять чтение с устройства и сохранять архивированный журнал в виде файла. DB2 не будет вызывать программу обработчика пользователя для считывания файлов журнала на непосредственное устройство. Вместо этого во время восстановления с повтором транзакций DB2 будет читать заголовки

экстентов для определения того, содержится ли на непосредственном устройстве нужный файл журнала. Если необходимый файл журнала не обнаружен на этом непосредственном устройстве, DB2 будет производить поиск по пути переполнения журнала. Если файл журнала все еще не обнаружен, DB2 вызовет программу обработчика пользователя для считывания файла журнала по пути переполнения журнала. Если не указать путь переполнения журнала для операции повтора транзакций,DB2 не будет вызывать программу обработчика пользователя для считывания журнала. Дополнительную информацию о вызове программы обработчика пользователя смотрите в разделе “Формат вызова” на стр. 474.

- Если вы используете DPROP и записываете файлы журнала на непосредственное устройство, API чтения журнала не будет вызывать программу обработчика пользователя для считывания файлов журнала. Тем не менее запрошенные записи журнала будут возвращены, если они доступны на этом устройстве. Если вы запрашиваете файлы журнала с датой более ранней, чем есть на устройстве, они не будут возвращены (поведение аналогично тому случаю, когда DB2 не в состоянии найти файл журнала, содержащий необходимые записи).

  Примечания:

  1. При использовании для записи в журнал непосредственного устройства не рекомендуется использовать DPROP.
  2. При использовании API **sqlurlog** непосредственное устройство для записи в журнал использовать нельзя.

Потеря файлов журнала

Отбрасывание базы данных стирает все файлы журнала в текущем каталоге пути к журналу базы данных. Перед отбрасыванием базы данных может быть полезным сделать копии файлов журнала.

Если вы выполняете повтор транзакций базы данных до заданного момента времени, повторно используются последний использованный файл журнала и все следующие за ним существующие файлы журнала; вы теряете возможность восстановления до момента после этого заданного. Поэтому *перед* началом восстановления до определенного момента времени необходимо скопировать все файлы журнала в текущем каталоге журнала базы данных.

Когда операция восстановления с повтором транзакций заканчивается, файл журнала с последней принятой транзакцией усекается, и запись в журнал начинается со следующего последовательного номера. Если у вас нет копий отброшенного файла и файлов со следующими за ним номерами, вы не сможете восстанавливать базу данных до момента после заданного. (Когда возобновляется нормальная работа базы данных, создаются новые файлы журналов, которые можно использовать для последующих восстановлений.)

Если вы изменили каталог пути к журналу, а затем удалили этот подкаталог или стерли какие-либо из файлов журнала в подкаталоге, вызываемом по пути журнала, при открытии базы данных менеджер баз данных будет искать эти файлы журнала пути к журналу по умолчанию SQLOGDIR. Если файлы журнала не найдены, база данных переходит в состояние ожидания резервного копирования, и для ее дальнейшего использования надо сделать резервную копию базы данных. Эту резервную копию надо сделать даже в том случае, когда в этом подкаталоге содержатся пустые файлы журнала.

Если вы утеряли файл журнала, содержащий конечную точку резервной копии, сделанной в оперативном режиме, и производите повтор транзакций для соответствующего восстановленного образа, база данных будет непригодна к использованию. Чтобы сделать базу данных пригодной к использованию, необходимо восстановить ее из другой резервной копии и всех связанных с ней файлов журнала.

Вы можете столкнуться с ситуацией, аналогичной следующей: вы хотите сделать восстановление с повтором транзакций для целой базы данных, но боитесь в процессе восстановления потерять журнал. (Это может произойти, если у вас есть много архивных файлов журнала между временем образа последней резервной копии базы данных и моментом времени, до которого вы хотите восстановить базу данных.)

Прежде всего следует скопировать все применимые файлы журнала в “безопасное” место. Затем можно запустить команду RESTORE и использовать метод восстановления с повтором транзакций до необходимого момента времени. Если какие-нибудь из необходимых файлов будут повреждены или утеряны во время этого процесса, у вас есть резервная копия всех файлов журнала в другом месте.

Что такое файл хронологии восстановления

Файл хронологии восстановления создается с каждой из баз данных и автоматически обновляется всякий раз, когда:

- Для базы данных или табличных пространств снимается резервная копия
- База данных или табличные пространства восстанавливаются
- Для базы данных или табличных пространств выполняется повтор транзакций
- Создается табличное пространство
- Изменяется табличное пространство
- Стабилизируется табличное пространство
- Переименовывается табличное пространство
- Отбрасывается табличное пространство
- Загружается таблица

- Отбрасывается таблица
- Реорганизуется таблица
- Изменяется статистика таблицы

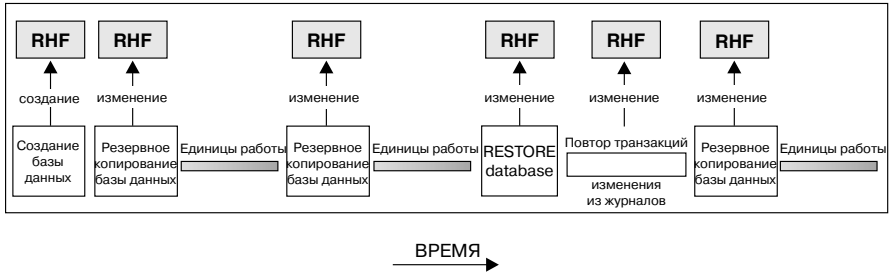


Рисунок 11. Создание и обновление файла хронологии восстановления

Сводную информацию о резервном копировании в этом файле можно использовать для восстановления всей или части базы данных до определенного момента времени. Информация в файле включает в себя:

- Поле идентификации (ID) для уникальной идентификации каждой записи
- Какая часть базы данных была скопирована и как
- Время снятия копии
- Положение копии (информация об устройстве и логическом пути доступа к копии)
- Время последнего восстановления из резервной копии
- Время переименования табличного пространства с показом его предыдущего и текущего имени
- Состояние операции резервного копирования: активная, неактивная, с истекшим сроком действия или удаленная
- Последний номер последовательности файлов журналов, сохраненный при резервном копировании базы данных или восстановлении с повтором транзакций.

Чтобы посмотреть записи в файле хронологии восстановления, используйте команду LIST HISTORY. Дополнительную информацию об этой команде смотрите в книге “LIST HISTORY” на стр. 343.

Примечание: Когда восстановление из резервной копии и повтор транзакций выполняются до конца файлов журнала, ID резервной копии, показанный после команды LIST HISTORY, указывает предельное

возможное время, то есть 99991231235959. ID резервной копии трансформируется таким образом только после выполнения повтора транзакций.

Каждая операция резервного копирования (базы данных, табличного пространства или инкрементная) включает в себя копирование файла хронологии восстановления. Этот файл хронологии восстановления связывается с базой данных. Отбрасывание базы данных удаляет файл хронологии восстановления. Восстановление базы данных в новом положении восстанавливает файл хронологии восстановления. Восстановление не переписывает существующий файл хронологии восстановления.

Если текущая база данных непригодна к использованию или недоступна, а связанный с ней файл хронологии восстановления поврежден или удален, опция команды RESTORE позволяет восстановить только файл хронологии восстановления. Затем этот файл хронологии восстановления можно будет просмотреть, чтобы получить информацию о том, какую резервную копию использовать для восстановления базы данных.

Размером этого файла управляет параметр конфигурации *rec_his_retentn*, указывающий время хранения (в днях) записей в файле. Даже если для этого параметра установлен нуль (0), сохраняется самая свежая резервная копия базы данных (и ее набор восстановления). (Эту копию можно удалить только путем использования PRUNE с опцией FORCE.) Период хранения по умолчанию - 366 дней. Можно задать неопределенное число дней хранения, указав значение -1. В этом случае сокращение файла надо выполнять явно. Дополнительную информацию об этом параметре конфигурации смотрите в руководстве *Administration Guide: Performance*.

Чистка мусора

Хотя удалять записи из файла хронологии можно в любой момент при помощи команды PRUNE HISTORY (смотрите раздел “PRUNE HISTORY/LOGFILE” на стр. 346), рекомендуется оставлять такую чистку DB2. За числом резервных копий баз данных DB2, записанных в файле хронологии восстановления автоматически следит функция DB2 под названием *чистка мусора*. Чистка мусора DB2 вызывается после успешного завершения операции резервного копирования базы данных, а также после успешного завершения восстановления базы данных. Число сохраняемых активных полных (не инкрементных) резервных копий определяет параметр конфигурации *num_db_backups*. Значение этого параметра конфигурации используется для просмотра файла хронологии, начиная с последней записи.

После каждого резервного копирования всей базы данных параметр конфигурации *rec_his_retentn* используется для удаления из файла хронологии записей с истекшим сроком.

Что такое файл хронологии восстановления

Активной резервной копией называется копия, при помощи которой можно восстановить текущее состояние базы данных, используя текущие файлы журнала для повтора транзакций. *Неактивная резервная копия базы данных* при восстановлении возвращает базу данных в более раннее состояние.

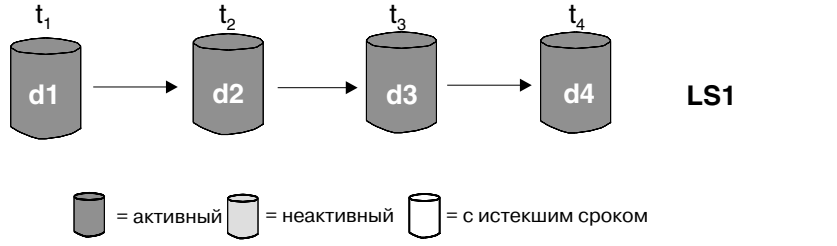


Рисунок 12. Активные резервные копии базы данных. Значение *num_db_backups* равно 4.

Все активные, но уже не нужные резервные копии базы данных помечаются как копии “с истекшим сроком”. Эти копии рассматриваются как ненужные, поскольку доступны более свежие резервные копии. Все резервные копии табличных пространств и резервные копии загрузки, сделанные до истечения срока годности резервной копии базы данных, также помечаются как копии “с истекшим сроком”.

Все резервные копии базы данных, помеченные как “неактивные” и сделанные раньше момента снятия резервной копии с истекшим сроком годности, также помечаются как копии “с истекшим сроком”. Все связанные неактивные

резервные копии табличных пространств и резервные копии загрузки также помечаются как копии “с истекшим сроком”.
Если восстановлена активная, но не самая последняя резервная копия базы

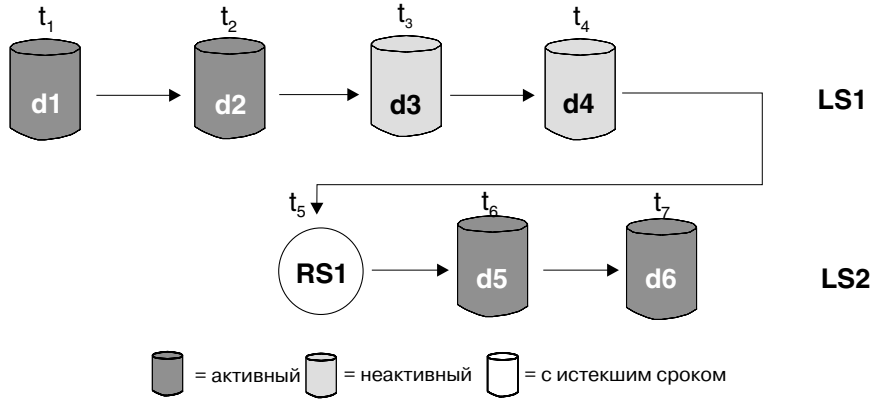


Рисунок 13. Неактивные резервные копии базы данных

данных, записанная в файле хронологии, все последующие резервные копии базы данных, принадлежащие той же самой последовательности файлов журнала, помечаются как “неактивные”.

Если восстановлена неактивная резервная копия базы данных, все неактивные резервные копии базы данных, принадлежащие к текущей последовательности файлов журнала, снова помечаются как “активные”. Все активные, но уже не находящиеся в текущей последовательности резервные копии базы данных помечаются как копии “с истекшим сроком”.

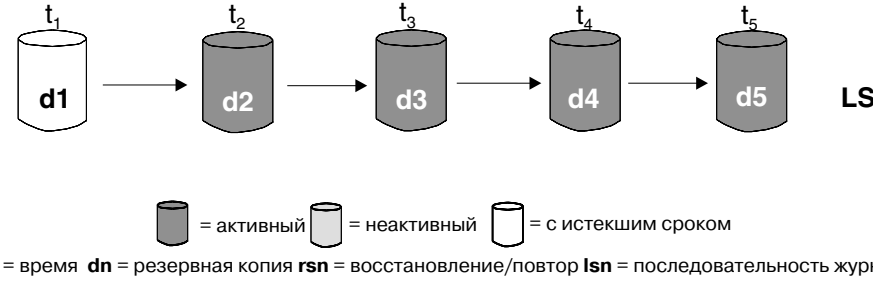


Рисунок 14. Резервные копии базы данных с истекшим сроком годности

Чистка мусора DB2 помечает также записи файла хронологии для резервной копии базы данных DB2 или табличного пространства как “неактивные”, если эта резервная копия не соответствует текущей *последовательности файлов журнала*. Текущая последовательность файлов журнала определяется восстановленной резервной копией базы данных DB2 и обработанными

файлами журнала. После восстановления резервной копии базы данных все резервные копии базы данных, сделанные после ее восстановления, становятся “неактивными”, поскольку восстановленная резервная копия начинает новую цепочку файлов журнала. (Это справедливо, если резервная копия восстановлена без повтора транзакций. Если же выполняется повтор транзакций, все резервные копии базы данных, снятые после прерывания цепочки журналов, помечаются как “неактивные”. Возможно восстановление более ранней резервной копии базы данных, поскольку утилита повтора транзакций может таким образом пройти последовательность журналов,содержащую поврежденный текущий образ резервной копии.)

Резервная копия табличного пространства становится “неактивной”, если после ее восстановления текущего состояния базы данных DB2 нельзя достичь с использованием текущей последовательности файлов журнала.

Если резервная копия содержит столбцы DATALINK, всем работающим серверам связей данных посылается требование выполнить чистку мусора. Чистка мусора DB2 затем удаляет резервные копии связанных файла серверов связей данных, которые содержались в резервной копии с истекшим сроком, но были отсоединены до следующей операции копирования базы данных.

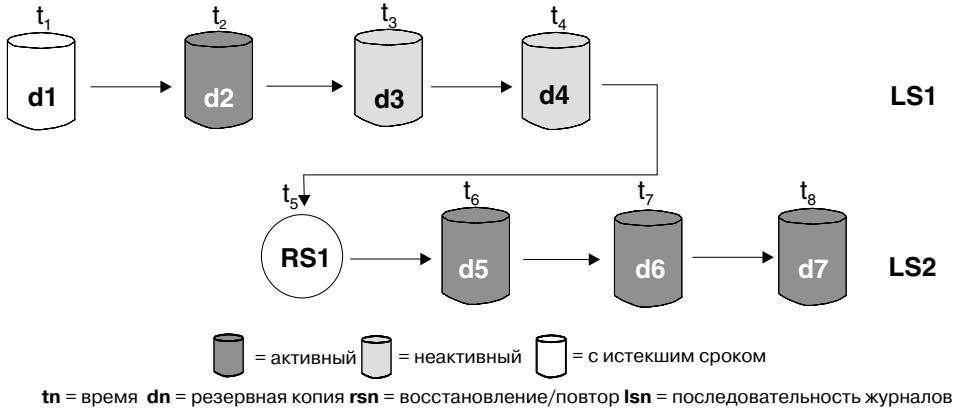


Рисунок 15. Резервные копии базы данных - активные, неактивные и с истекшим сроком

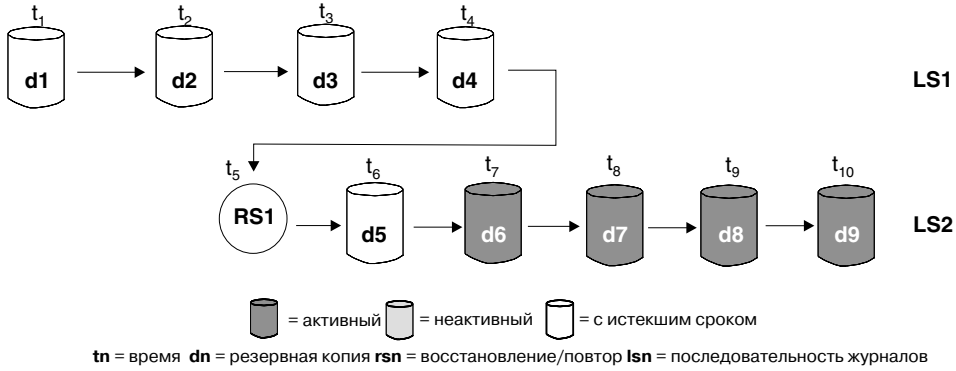


Рисунок 16. Последовательность файлов журнала с истекшим сроком годности

Что такое состояние табличного пространства

Текущий статус табличного пространства отражается его *состоянием*. С восстановлением тесно связаны следующие состояния табличного пространства:

- *Отложенный повтор транзакций*. Табличное пространство переводится в это состояние после восстановления или после ошибки ввода/вывода. После восстановления можно провести повтор транзакций до конца журналов или до момента времени. После ошибки ввода/вывода для табличного пространства должен быть выполнен повтор транзакций до конца журналов.
- *Выполняется повтор транзакций*. Табличное пространство переводится в это состояние, когда для него выполняется операция повтора транзакций. Когда операция повтора транзакций успешно завершается, табличное пространство выводится из этого состояния. Табличное пространство выводится из этого состояния также при отмене операции повтора транзакций.
- *Отложенное восстановление*. Табличное пространство переводится в это состояние, если для него отменена операция повтора транзакций, или при повторе транзакций происходит неисправимая ошибка, после которой требуется восстановить это табличное пространство и снова выполнить для него повтор транзакций.
- *Отложенное резервное копирование*. Табличное пространство переводится в это состояние после повтора транзакций до момента времени или после операции загрузки без опции копирования. Чтобы это табличное пространство можно было использовать, необходимо сделать его резервную копию. (Пока копия не будет сделана, изменения табличного пространства не допускаются,однако операции чтения разрешены.)

Повышение производительности восстановления

Решая вопрос повышения производительности восстановления, следует учитывать:

- 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, задайте значение, 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.

Параллельное восстановление

Теперь DB2 использует несколько агентов как при восстановлении после аварии, так и при восстановлении с повтором транзакций базы данных. Можно ожидать повышения производительности при этих операциях, особенно в симметричной многопроцессорной среде (SMP); использование нескольких агентов при восстановлении базы данных использует дополнительные процессоры, доступные на компьютерах SMP.

С этим усовершенствованием связан новый тип агента - db2agnsc. DB2 выбирает число агентов, используемых при восстановлении базы данных, на основе числа процессоров в компьютере. Для компьютеров SMP число используемых агентов равно числу процессоров + 1. На компьютерах с одним процессором несколько агентов используются для более производительного чтения журналов, обработки записей журналов и предварительной выборки страниц данных.

DB2 распределяет записи журнала между агентами так, чтобы они могли применять их одновременно, где возможно. Например, параллельно может идти обработка записей журнала, связанная с операциями вставки, удаления, изменения, добавления ключа и удаления ключа. Поскольку записи журнала

выполняются параллельно на уровне страниц (записи журнала на одной и той же странице данных обрабатываются одним агентом), производительность растет даже в том случае, когда вся работа выполняется на одной таблице.

Особенности менеджера связей данных DB2

В следующих разделах приводится информация для таблиц со столбцами DATALINK. Полное описание столбцов DATALINK смотрите в описании оператора CREATE TABLE в справочнике *SQL Reference*.

Особенности восстановления после отказов

Когда прикладная программа выдает запросы SQL, вовлекающие в работу серверы связей данных, на которых работает менеджер связей данных DB2 (используя столбцы DATALINK с атрибутом FILE LINK CONTROL), менеджер баз данных распределяет работу среди серверов связей данных. Он также следит за серверами связей данных, которые участвуют в транзакции. Когда прикладная программа требует принять транзакцию, менеджер баз данных выполняет принятие, используя протокол двухфазного принятия. На первой фазе менеджер баз данных заносит в журнал запись PREPARE и передает требование PREPARE всем серверам связей данных. Серверы связей данных отвечают одним из следующих способов:

- YES - сервер связей данных готов к принятию
- NO - из-за ошибки сервер связей данных не готов к принятию.

Первая фаза считается успешной, если все серверы связей данных ответили “YES”.

Обработка на второй фазе зависит от результата первой фазы. Если по хотя бы один сервер связей данных ответил “NO”, менеджер баз данных передает всем участвующим в транзакции серверам связей данных требование ABORT. Выполняется откат транзакции, а прикладной программе возвращается сообщение об ошибке SQL0903N с кодом причины “03”. В противном случае менеджер баз данных производит принятие транзакции, как он это делает обычно без участия серверов связей данных. В конце обработки транзакции он передает требование COMMIT всем участвующим в ней серверам связей данных.

Если на сервере связей данных произошла ошибка, оставившая какие-либо транзакции в состоянии PREPARED, такие транзакции называются *неоднозначными транзакциями*. Менеджер баз данных отвечает за отслеживание результата таких транзакций и за их периодическое разрешение на сервере связей данных. Как только менеджер баз данных определяет, что ошибка, возможно, создала на сервере связей данных неоднозначные транзакции, он помечает этот сервер, как нуждающийся в восстановлении после отказа. Пока сервер находится в этом состоянии, запрещены все требования SQL, вовлекающие его в работу. Прикладной программе, выдавшей такое требование SQL, возвращается сообщение SQL0357N с кодом причины “03”.

Во время обработки RESTART, ACTIVATE DATABASE или первой операции CONNECT менеджер баз данных пытается связаться с каждым из сконфигурированных серверов связей данных и разрешить неоднозначные транзакции путем отказа от них или их принятия. Состояние сервера связей данных помечается как доступное, если разрешены все его неоднозначные транзакции, за исключением тех, которые неоднозначны и на самом менеджере баз данных. В доступном состоянии разрешены требования SQL, затрагивающие этот сервер связей данных. В конце этой попытки разрешения неоднозначных транзакций, если менеджер баз данных определяет, что на сервере связей данных потенциально остались неоднозначные транзакции, он помечает этот сервер как нуждающийся в восстановлении после отказа. Это, например, может произойти, если сервер связей данных недоступен во время обработки RESTART, ACTIVATE DATABASE или первого операции CONNECT, а также в том случае, когда сервер связей данных во время этой обработки обнаружил ошибку.

Когда сконфигурированный для базы данных сервер связей данных находится в состоянии, требующем восстановления после отказа, менеджер баз данных не разрешает требования SQL, затрагивающие конкретно этот сервер связей данных. Требования SQL, затрагивающие остальные данные в базе данных, по-прежнему разрешены. Менеджер баз данных запускает процесс, который асинхронно пытается выполнить восстановление после отказа на всех серверах связей данных, на которых это необходимо. Когда этот процесс успешно выполнит восстановление после отказа, состояние сервера связей данных помечается как доступное, тем самым разрешая дальнейшие требования SQL, затрагивающие этот сервер.

Особенности утилиты резервного копирования

DB2 проверяет, что ко времени завершения операции резервного копирования для связанных файлов при работе менеджера связей данных DB2 на серверах связей данных также сделана резервная копия. (Утилита резервного копирования может быть запущена как без отключения, так и в автономном режиме, а образ резервной копии может относиться как к базе данных, так и к табличному пространству.) Дальнейшее описание касается только файлов, указанных в столбцах DATALINK, у которых для параметра RECOVERY задано YES. (Для файлов, указанных в столбцах DATALINK с RECOVERY=NO, резервная копия не делается.)

Когда на файлы есть ссылки, серверы связей данных планируют их асинхронное копирование на архивный сервер, например, TSM, или на диск. При работе утилиты резервного копирования DB2 проверяет, что скопированы все внесенные в расписание для копирования файлы. В начале резервного копирования DB2 связывается со всеми серверами связей данных, указанные в файле конфигурации DB2. Если для базы данных сконфигурирован один или несколько серверов связей данных, операция завершится успешно, даже если ни один сервер связей данных не доступен. После перезапуска сервера связей данных до того, как он станет доступным базе данных, на нем будет завершено

резервное копирование. Если у сервера связей данных есть один или несколько связанных файлов, и он не запущен или же остановлен во время резервного копирования, резервная копия не будет содержать полной информации DATALINK. Операция резервного копирования завершится успешно. В поле комментария файла хронологии для этой операции резервного копирования будет указан недоступный сервер связей данных, или, если недоступны более 4 серверов, число недоступных серверов. Когда для менеджеров связей данных будет успешно выполнено резервное копирование, в поле комментария будет записано его предыдущее значение.

Прежде чем сервер связей данных сможет быть снова помечен как доступный для базы данных, должен успешно завершиться процесс резервного копирования для всех отложенных резервных копий. (Это происходит автоматически асинхронным образом.) Если операция резервного копирования запущена, когда в очереди уже есть двукратное число *num_db_backups* (смотрите ниже) незавершенных сервером связей данных операций резервного копирования, эта операция завершается неудачно. Перед последующими операциями резервного копирования надо перезапустить этот сервер связей данных и завершить отложенные операции резервного копирования.

Когда ссылка на файл удаляется, сам файл либо удаляется, либо возвращается к предыдущему режиму разрешений в зависимости от значения, указанного для параметра ON UNLINK. Успешная операция резервного копирования может привести к тому, что серверы связей данных сотрут архивные версии файлов на сервере архивирования (на диске или TSM, смотрите раздел “Чистка мусора” на стр. 55). Параметр конфигурации базы данных *num_db_backups* указывает число резервных копирований базы данных DB2 до удаления архивированных версий файлов (на которые были удалены ссылки). Дополнительную информацию об этом параметре конфигурации смотрите в руководстве *Administration Guide: Performance*.

При удалении файлов, на которые удалены ссылки, информация об этих файлах также удаляется из регистрационных таблиц сервера связей данных.

Выбор метода резервного копирования для менеджера связей данных DB2 в AIX

Для резервного копирования файлов с сервера связей данных кроме копирования на диск и XBSA можно также использовать Tivoli Storage Manager (TSM).

Чтобы в качестве сервера архивов использовать Tivoli Storage Manager:

1. Установите Tivoli Storage Manager на сервере связей данных. Дополнительную информацию смотрите в документации по используемому продукту Tivoli Storage Manager.

2. Зарегистрируйте прикладную программу клиента связей данных на сервере Tivoli Storage Manager. Дополнительную информацию смотрите в документации по используемому продукту Tivoli Storage Manager.
3. В файлы сценариев `db2profile` или `db2cshrc` администратора менеджера связей данных добавьте следующие переменные среды:

```
(для оболочки Bash, Bourne или Korn)
export DSMI_DIR=/usr/tivoli/tsm/client/api/bin
export DSMI_CONFIG=$HOME/tsm/dsm.opt
export DSMI_LOG=$HOME/dldump
export PATH=$PATH:$DSMI_DIR

(для оболочки C)
setenv DSMI_DIR /usr/tivoli/tsm/client/api/bin
setenv DSMI_CONFIG ${HOME}/tsm/dsm.opt
setenv DSMI_LOG ${HOME}/dldump
setenv PATH=${PATH}:$DSMI_DIR
```

4. Убедитесь, что файл системных опций TSM `dsm.sys` находится в каталоге `$DSMI_DIR`.
5. Убедитесь, что файл пользовательских опций TSM `dsm.opt` находится в каталоге *INSTHOME*`/tsm`, где *INSTHOME* - начальный каталог администратора менеджера связей данных.
6. В файле системных опций Tivoli Storage Manager `/usr/tivoli/tsm/client/api/bin/dsm.sys` задайте для опции *PASSWORDACCESS* значение `generate`.
7. Зарегистрируйте пароль TSM с опцией `generate` *перед* первым запуском менеджера файлов связей данных. Теперь не нужно будет задавать пароль при установлении соединения менеджера файлов связей данных с сервером TSM. Дополнительную информацию смотрите в документации по вашему продукту TSM.
8. Для переменной реестра DLFM_BACKUP_TARGET задайте значение TSM. В этом случае значение переменной реестра DLFM_BACKUP_DIR_NAME будет игнорироваться. Для резервного копирования будет использоваться Tivoli Storage Manager.

   Примечания:

   a. Имейте в виду, что если во время работы изменить значение переменной реестра DLFM_BACKUP_TARGET с TSM на положение на диске, созданные файлы архивов не будут перемещены в новое положение. Например, если менеджер файлов связей данных был запущен со значением TSM переменной реестра DLFM_BACKUP_TARGET и затем значение этой переменной реестра было изменено на положение на диске, вновь создаваемые файлы архивов будут сохраняться в этом новом положении на диске. Файлы архивов, сохраненные ранее в TSM, не будут перемещены в это новое положении на диске.

b. Чтобы переопределить класс управления TSM по умолчанию, можно использовать новую переменную реестра DLFM_TSM_MGMTCLASS. Если эта переменная реестра не задана, используется класс управления TSM по умолчанию.

9. Остановите менеджер файлов связей данных командой **dlfm stop**.
10. Запустите менеджер файлов связей данных командой **dlfm start**.

Выбор метода резервного копирования для менеджера связей данных DB2 в операционной среде Solaris

Для резервного копирования файлов с сервера связей данных кроме копирования на диск и XBSA можно также использовать Tivoli Storage Manager (TSM).

Чтобы в качестве сервера архивов использовать Tivoli Storage Manager:

1. Установите Tivoli Storage Manager на сервере связей данных. Дополнительную информацию смотрите в документации по используемому продукту Tivoli Storage Manager.
2. Зарегистрируйте прикладную программу клиента связей данных на сервере Tivoli Storage Manager. Дополнительную информацию смотрите в документации по используемому продукту Tivoli Storage Manager.
3. В файлы сценариев `db2profile` или `db2cshrc` администратора менеджера связей данных добавьте следующие переменные среды:

```
(для оболочки Bash, Bourne или Korn)
export DSMI_DIR=/opt/tivoli/tsm/client/api/bin
export DSMI_CONFIG=$HOME/tsm/dsm.opt
export DSMI_LOG=$HOME/dldump
export PATH=$PATH:/opt/tivoli/tsm/client/api/bin

(для оболочки C)
setenv DSMI_DIR /opt/tivoli/tsm/client/api/bin
setenv DSMI_CONFIG ${HOME}/tsm/dsm.opt
setenv DSMI_LOG ${HOME}/dldump
setenv PATH=${PATH}:/opt/tivoli/tsm/client/api/bin
```

4. Убедитесь, что файл системных опций TSM `dsm.sys` находится в каталоге `/opt/tivoli/tsm/client/api/bin`.
5. Убедитесь, что файл пользовательских опций TSM `dsm.opt` находится в каталоге *INSTHOME*`/tsm`, где *INSTHOME* - начальный каталог администратора менеджера связей данных.
6. В файле системных опций Tivoli Storage Manager `/opt/tivoli/tsm/client/api/bin/dsm.sys` задайте для опции *PASSWORDACCESS* значение `generate`.
7. Зарегистрируйте пароль TSM с опцией `generate` *перед* первым запуском менеджера файлов связей данных. Теперь не нужно будет задавать пароль при установлении соединения менеджера файлов связей данных с сервером TSM. Дополнительную информацию смотрите в документации по вашему продукту TSM.

8. Для переменной реестра DLFM_BACKUP_TARGET задайте значение TSM. В этом случае значение переменной реестра DLFM_BACKUP_DIR_NAME будет игнорироваться. Для резервного копирования будет использоваться Tivoli Storage Manager.

   Примечания:

   a. Имейте в виду, что если во время работы изменить значение переменной реестра DLFM_BACKUP_TARGET с TSM на положение на диске, созданные файлы архивов не будут перемещены в новое положение. Например, если менеджер файлов связей данных был запущен со значением TSM переменной реестра DLFM_BACKUP_TARGET и затем значение этой переменной реестра было изменено на положение на диске, вновь создаваемые файлы архивов будут сохраняться в этом новом положении на диске. Файлы архивов, сохраненные ранее в TSM, не будут перемещены в это новое положении на диске.

   b. Чтобы переопределить класс управления TSM по умолчанию, можно использовать новую переменную реестра DLFM_TSM_MGMTCLASS. Если эта переменная реестра не задана, используется класс управления TSM по умолчанию.

9. Остановите менеджер файлов связей данных командой **dlfm stop**.
10. Запустите менеджер файлов связей данных командой **dlfm start**.

Выбор метода резервного копирования для менеджера связей данных DB2 в Windows NT

При вставке значения DATALINK в таблицу со столбцом DATALINK, для которого определено восстановление, в расписание вносится операция резервного копирования на сервере архивов соответствующих файлов DATALINK с сервера связей данных. В настоящее время поддерживаются две опции резервного копирования файлов на сервере архивов: копирование на диск (метод по умолчанию) и Tivoli Storage Manager. В будущих выпусках менеджера связей данных DB2 для Windows NT будут поддерживаться другие носители и программы резервного копирования.

Копирование на диск (метод по умолчанию)

Когда на сервере DB2 вызвана утилита резервного копирования, для заданных в базе данных файлов на сервере связей данных создается резервная копия в каталоге, задаваемом переменной среды DLFM_BACKUP_DIR_NAME. По умолчанию эта переменная имеет значение `c:\dlfmbackup`, где `c:\` обозначен диск резервной установки менеджера связей данных.

Чтобы задать для этой переменной значение c:\dlfmbackup, введите следующую команду:

```
db2set -g DLFM_BACKUP_DIR_NAME=c:\dlfmbackup
```

Положение, задаваемое в переменной среды DLFM_BACKUP_DIR_NAME, *не* должно определяться в файловой системе с помощью фильтра файловой системы связей данных и каталог, заданный для резервного копирования файлов, должен содержать достаточно свободного пространства.

Задайте также для переменной DLFM_BACKUP_TARGET значение LOCAL, используя следующую команду:

```
db2set -g DLFM_BACKUP_TARGET=LOCAL
```

После задания или изменения этих переменных остановите и перезапустите менеджер файлов связей данных с помощью команд **dlfm stop** и **dlfm start**.

Tivoli Storage Manager

Чтобы в качестве сервера архивов использовать Tivoli Storage Manager:

1. Установите Tivoli Storage Manager на сервере связей данных. Дополнительную информацию смотрите в документации по используемому продукту Tivoli Storage Manager.
2. Зарегистрируйте прикладную программу клиента связей данных на сервере Tivoli Storage Manager. Дополнительную информацию смотрите в документации по используемому продукту Tivoli Storage Manager.
3. Нажмите кнопку **Пуск** и выберите **Настройка –> Панель управления –> Система**. Откроется окно Свойства системы. Выберите закладку **Среда** и введите следующие переменные среды и их значения:

Переменная	Значение
DSMI_DIR	c:\tsm\baclient
DSMI_CONFIG	c:\tsm\baclient\dsm.opt
DSMI_LOG	c:\tsm\dldump

4. Убедитесь, что файл системных опций TSM `dsm.sys` находится в каталоге `c:\tsm\baclient`.
5. Убедитесь, что файл пользовательских опций TSM `dsm.opt` находится в каталоге `c:\tsm\baclient`.
6. В файле системных опций Tivoli Storage Manager `c:\tsm\baclient\dsm.sys` задайте для опции *PASSWORDACCESS* значение `generate`.
7. Зарегистрируйте пароль TSM с опцией `generate` *перед* первым запуском менеджера файлов связей данных. Теперь не нужно будет задавать пароль при установлении соединения менеджера файлов

связей данных с сервером TSM. Дополнительную информацию смотрите в документации по вашему продукту TSM.

8. Для переменной среды DLFM_BACKUP_TARGET задайте значение TSM, используя следующую команду:

```
db2set -g DLFM_BACKUP_TARGET=TSM
```

В этом случае значение переменной среды DLFM_BACKUP_DIR_NAME будет игнорироваться. Для резервного копирования будет использоваться Tivoli Storage Manager.

Примечания:

a. Имейте в виду, что если во время работы изменить значение переменной среды DLFM_BACKUP_TARGET с TSM на LOCAL, созданные файлы архивов не будут перемещены в новое положение. Например, если менеджер файлов связей данных был запущен со значением TSM переменной среды DLFM_BACKUP_TARGET и затем значение этой переменной среды было изменено на LOCAL, вновь создаваемые файлы архивов будут сохраняться в этом новом положении на диске. Файлы архивов, сохраненные ранее в TSM, не будут перемещены в это новое положении на диске.

b. Чтобы переопределить класс управления TSM по умолчанию, можно использовать новую переменную среды DLFM_TSM_MGMTCLASS. Если эта переменная не задана, используется класс управления TSM по умолчанию.

9. Остановите менеджер файлов связей данных командой **dlfm stop**.
10. Запустите менеджер файлов связей данных командой **dlfm start**.

Резервное копирование файловой системы с журналами в AIX

В AIX, остановив менеджер связей данных, можно выполнить автономное резервное копирование файловой системы с журналами. Для пользователей, которым требуется большая доступность системы, предлагается следующий подход, позволяющий избежать необходимости останавливать менеджер связей данных.

1. Возьмите исходный файл CLI `quiesce.c` и сценарий оболочки `online.sh`. Эти файлы находятся в каталоге `/samples/dlfm`.
2. Скомпилируйте `quiesce.c`:

```
xlC -o quiesce  -L$HOME/sqllib/lib -I$HOME/sqllib/include -c quiesce.c
```

3. Как пользователь root, выполните этот сценарий на узле с файловой системой DLFS.

В сценарии оболочки `online.sh` предполагается, что на узле менеджера связей данных есть запись каталога для каждой базы данных, зарегистрированной на

менеджере связей данных. Подразумевается также, что /etc/filesystems содержит полную запись для файловой системы DLFS. Этот сценарий оболочки делает следующее:

- Стабилизирует все таблицы в базах данных, зарегистрированных на менеджере связей данных. Это предотвращает все новые операции с ними.
- Размонтирует файловую систему и вновь монтирует ее как файловую систему только для чтения.
- Выполняет резервное копирование файловой системы.
- Размонтирует файловую систему и вновь монтирует ее как файловую систему только для чтения и записи.
- Выводит таблицы DB2 из состояния стабилизации.

Этот сценарий нужно изменить в соответствии с вашей средой:

1. Выберите команду резервного копирования и задайте ее в функции do_backup в этом сценарии.
2. Задайте в этом сценарии следующие переменные среды:
   - В качестве значения DLFM_INST задайте имя экземпляра DLFM.
   - В качестве значения PATH_OF_EXEC задайте путь выполняемого файла ″quiesce″.

Запустите этот сценарий:

```
online.sh <имя_файловой_системы>
```

Особенности утилиты восстановления и повтора транзакций

Приведенная ниже информация касается случая, когда есть столбцы DATALINK, определенные для таблицы с опцией RECOVERY=YES. Если у таблицы есть столбец DATALINK, определенный с опцией RECOVERY=NO, по окончании операции восстановления эта таблица переводится в состояние ожидания синхронизации связей данных. Дополнительную информацию смотрите в разделе “Синхронизация связей данных” на стр. 84.

Во время операций восстановления из резервной копии таблицы со столбцами DATALINK могут быть переведены в одно из двух состояний:

- *Согласование связей данных невозможно*

  Когда таблица находится в состоянии невозможности синхронизации связей данных, для нее возможны без ограничений операции над столбцами, которые не являются столбцами DATALINK. Если столбец DATALINK включается в оператор SELECT, возвращается предупреждение. В отношении столбцов DATALINK можно выполнять вызовы UPDATE (с некоторыми ограничениями - подробности смотрите в разделе “Вывод таблицы из состояния невозможности согласования связей данных” на стр. 83). Нельзя выполнять операторы INSERT и DELETE, поскольку они затрагивают столбец DATALINK.

- *Согласование связей данных отложено*

  Когда таблица находится в состоянии отложенной синхронизации связей данных, для нее возможны без ограничений операции над столбцами, которые не являются столбцами DATALINK. Если столбец DATALINK включается в оператор SELECT, возвращается предупреждение. Нельзя выполнять любые операторы DML, такие как UPDATE, INSERT или DELETE.

Информация об этих состояниях отображается в файле `db2diag.log` при работе утилиты восстановления из резервной копии или повтора транзакций. Для получения этой информации можно использовать также команду **db2dart**.

Когда вы восстанавливаете базу данных или табличное пространство, для успешного завершения операции восстановления необходимы следующие условия:

- Если не работает любой из серверов связей данных, записанных в файле резервной копии, операция восстановления все равно завершится успешно.

  Таблицы с информацией о столбце DATALINK, пострадавшие из-за отсутствия сервера связей данных, после операции восстановления будут переведены в состояние отложенного согласования связей данных (или отложенного повтора транзакций - если выполнялся повтор транзакций). Прежде чем серверы связей данных смогут быть снова помечены как доступные для базы данных, этот процесс восстановления должен успешно завершиться. Асинхронный процесс, который завершает резервное копирование, как только менеджер связей данных станет доступным (смотрите раздел “Особенности утилиты резервного копирования” на стр. 63), завершает также обработку восстановления.
- Если любой из серверов связей данных, записанных в файле резервной копии, остановится во время операции восстановления, операция восстановления завершится неудачно. Восстановление можно перезапустить при неработающем сервере связей данных (смотри выше).
- Если предыдущая операция восстановления остается незавершенной на каких-либо серверах связей данных, последующие операции восстановления базы данных или табличных пространств будут завершаться неуспешно до тех пор, пока не будут перезапущены эти серверы связей данных и не будет завершено незавершенное восстановление.
- Информация о всех столбцах DATALINK, записанных в файл резервной копии, должна существовать в соответствующих регистрационных таблицах серверов связей данных.

  Если в регистрационных таблицах не записано всей информации о столбцах DATALINK, после завершения операции восстановления (или отложенного повтора транзакций - если выполнялся повтор транзакций) таблица с отсутствующим столбцом DATALINK переводится в состояние невозможности согласования связей данных.

Если резервная копия не записана в регистрационные таблицы, это может означать, что предоставленный файл резервной копии более ранний, чем значение для num_db_backups, и уже был помечен для ″чистки мусора″. Архивированные файлы из этой более ранней резервной копии были удалены и не могут быть восстановлены. Все таблицы, у которых есть столбцы DATALINK, переводятся в состояние отложенного согласования связей данных.

Если резервная копия не записана в регистрационные таблицы, это может означать, что процесс резервного копирования до сих пор не завершен, поскольку не работает сервер связей данных. Все таблицы, у которых есть столбцы DATALINK, переводятся в состояние отложенного согласования связей данных. Когда сервер связей данных будет перезапущен, процесс резервного копирования будет завершен до процесса восстановления.

Таблица остается доступной для пользователей, но ссылки в столбцах DATALINK могут быть неверными (например, файл, заданный значением столбца DATALINK, может быть не найден). Если вы не хотите этого, можно перевести таблицу в состояние ожидания проверки, введя оператор ″SET CONSTRAINTS for имя_таблицы TO DATALINK RECONCILE PENDING″.

Если после операции восстановления из резервной копии таблица находится в состоянии невозможности согласования связей данных, данные столбца DATALINK можно исправить одним из способов, предлагаемых в разделе “Вывод таблицы из состояния невозможности согласования связей данных” на стр. 83 .

Примечание: В процессе задания ссылки на файл может потребоваться считывание этого файла в файловую систему с архивного сервера. Если во время этого процесса произойдет ошибка (например, файл нельзя скопировать в файловую систему из-за наличия повторения имен), соответствующая таблица переводится в состояние отложенного согласования связей данных.

Настоятельно рекомендуется заархивировать файл `datalink.cfg` на случай некоторых нестандартных сценариев восстановления, поскольку файл `datalink.cfg` в резервной копии базы данных лишь отражает файл `datalink.cfg`, каким он был на момент резервного копирования. В некоторых случаях восстановления требуется иметь наиболее поздний файл `datalink.cfg`. Таким образом, необходимо создавать резервную копию файла `datalink.cfg` после каждого вызова команды ADD DATALINKS MANAGER или DROP DATALINKS MANAGER. Это поможет получить наиболее поздний файл `datalink.cfg`, если самый свежий файл `datalink.cfg` на диске недоступен.

Если наиболее поздний файл `datalink.cfg` на диске недоступен, замените существующий файл `datalink.cfg` (восстановленный из резервной копии)

наиболее поздним файлом `datalink.cfg`, заархивированным до запуска восстановления повтором транзакций. Сделайте это после восстановления базы данных.

Восстановление баз данных из резервных копий, сделанных в автономном режиме, без повтора транзакций

Восстановление без повтора транзакций можно выполнять только на уровне базы данных, но не на уровне табличного пространства. Восстановить базу данных без повтора транзакций можно, либо задав базу данных, для которой невозможен повтор транзакций (то есть использующую циклическую запись журнала), либо указав параметр WITHOUT ROLLING FORWARD в команде RESTORE DATABASE.

Если вы используете утилиту восстановления с опцией WITHOUT DATALINK, все таблицы со столбцами DATALINK переводятся в состояние отложенного согласования связей (DRP), а во время операции восстановления никакого согласования с серверами связей данных не выполняется.

Если вы не используете опцию WITHOUT DATALINK, а сервер связей данных, записанный в файл резервной копии, больше не определен для базы данных (то есть был отброшен при помощи команды DROP DATALINKS MANAGER), те таблицы, что содержали ссылки DATALINK на отброшенный сервер связей данных, утилитой восстановления переводятся в состояние DRP.

Если вы не используете опцию WITHOUT DATALINK, все серверы связей данных доступны и вся информация о столбцах DATALINK полностью записана в регистрационные таблицы, то для всех серверов связей данных, записанных в файл резервной копии, происходит следующее:

- Все файлы, которые были связаны после образа резервной копии, использованной для операции восстановления базы данных, помечаются как несвязанные (поскольку они не записаны в образ резервной копии как связанные).
- Все файлы, ссылки на которые были удалены после снятия образа резервной копии, но имелись до снятия образа резервной копии, помечаются как связанные (поскольку они записаны в образ резервной копии как связанные). Если файл впоследствии был связан с другой таблицей в другой базе данных, восстановленная таблица переводится в состояние отложенного согласования связей данных.

Примечание: Описанные шаги невозможны, если резервная копия, использованная для операции восстановления базы данных, была снята в момент, когда не работал хотя бы один сервер связей данных, поскольку информация DATALINK в резервной копии не полна. Описанные шаги также невыполнимы, если резервная копия, использованная для операции восстановления базы данных, была снята после восстановления базы данных с повтором

транзакций или без него. В обоих случаях все таблицы со столбцами DATALINK переводятся в состояние отложенного согласования связей, а во время операции восстановления никакого согласования с серверами связей данных не выполняется.

Восстановление баз данных и табличных пространств и повтор транзакций до конца журналов

При восстановлении базы данных или табличных пространств с последующим повтором транзакций до конца журналов (при условии, что все журналы доступны) проверка синхронизации не требуется, если только во время операции восстановления нет ни одного неработающего сервера связей данных, записанного в файле резервной копии. Если вы не уверены, что для операции повтора транзакций есть все файлы журнала, или полагаете, что может потребоваться синхронизация значений DATALINK, сделайте следующее:

1. Введите оператор SQL для затронутых таблиц:

   ```
   SET CONSTRAINTS FOR имя_таблицы
   TO DATALINK RECONCILE PENDING
   ```

   Это переведет таблицу в состояние отложенного согласования связей данных и в состояние отложенной проверки.

2. Если вы не хотите, чтобы таблица находилась в состоянии отложенной проверки, введите следующий оператор SQL:

   ```
   SET CONSTRAINTS FOR имя_таблицы IMMEDIATE CHECKED
   ```

   Это выведет таблицу из состояния отложенной проверки, но оставит ее в состоянии отложенного согласования связей данных. Для вывода таблицы из этого состояния необходимо использовать утилиту согласования.

Может оказаться, что файл резервной копии содержит ссылки DATALINK на менеджер связей DB2 (то есть менеджер связей DB2 был зарегистрирован в базе данных во время резервного копирования), который в дальнейшем был отброшен от базы данных. Для каждого табличного пространства, для которого выполняется восстановление с повтором транзакций и которое содержит хотя бы одну таблицу со ссылками DATALINK на отброшенный менеджер связей DB2, утилита повтора транзакций переводит все таблицы со столбцами DATALINK в состояние отложенного согласования связей данных.

Восстановление баз данных и табличных пространств и повтор транзакций до определенного момента времени

При работе с таблицами связей данных можно провести восстановление с повтором транзакций до конца файлов журнала или до заданного момента времени.

В конце операции повтора транзакций таблицы в табличных пространствах, для которых был выполнен повтор до определенного момента времени, переводятся в состояние отложенного согласования связей данных. Чтобы вывести их из

этого состояния, необходимо использовать утилиту согласования. Дополнительную информацию смотрите в разделе “Синхронизация связей данных” на стр. 84.

Пример повтора транзакций до определенного момента времени

Далее приводится простой сценарий, показывающий файлы, которые следует сохранить, чтобы выполнять восстановление из резервной копии и повтор транзакций. В этом примере показаны изменения значения одной строки в столбце типа DATALINK вместе с файлами, которые менеджер связей данных DB2 должен сохранить для поддержки восстановления. В этом примере предполагается, что отсутствует поддержка для восстановления этих файлов до времени раньше последнего резервного копирования. У серверов связей данных, на которых запущен менеджер связей данных DB2, такого ограничения нет. Заметьте, что файл A существует до момента 3, когда он удаляется, поскольку не был связан в момент 2, и политика в отношении базы данных в этом примере задает сохранение несвязанных файлы до следующего резервного копирования (то есть параметр конфигурации базы данных *num_db_backups* имеет значение 1).

Время	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Действие	Создание	Изменение	Резервное копирование	Изменение	Изменение	Удаление	Восстановление до 5
Значение столбца	значение A	значение B	значение B	значение C	значение D	-	значение D
Связанный файл	файл A	файл B	файл B	файл C	файл D	-	файл D
Дополнительные файлы, которые сохраняет DataLinks File Manager		файл A		файл B	файл B, файл C	файл B, файл C, файл D	файл B, файл C

Примечание: Восстановление связанных файлов всегда выполняется в сочетании с остальной базой данных.

Взаимодействие менеджера связей данных DB2 с восстановлением

В следующей таблице показаны разные типы восстановления, которые можно выполнить; действия менеджера связей данных DB2 во время восстановления и повтора транзакций, и необходимо ли запускать утилиту согласования после завершения восстановления:

Особенности менеджера связей данных DB2

Тип восстановления	Работа менеджера связей данных DB2 при восстановлении	Работа менеджера связей данных DB2 при повторе транзакций	Согласование
Невосстановимая база данных			
Восстановление базы данных из полной резервной копии при работе всех серверов связей данных	Выполняется быстрое согласование	Нет	Может быть запущена дополнительно при подозрении на ошибки связывания файлов
Восстановление базы данных с использованием опции WITHOUT DATALINK	Таблицы переводятся в состояние отложенного согласования связей данных	Нет	Обязательно
Восстановление базы данных из полной резервной копии, когда хотя бы один из серверов связей данных не работает	Быстрое согласование выполняется лишь на тех таблицах в табличных пространствах, у которых нет связей с неработающим сервером связей данных, тогда как остальные таблицы переводятся в состояние отложенного согласования связей данных	Нет	Требуется для таблиц в табличных пространствах со связями с неработающим сервером связей данных
Восстановление базы данных из неполной резервной копии при работе всех серверов связей данных	Быстрое согласование не производится, все таблицы со столбцами DATALINK переводятся в состояние отложенного согласования связей данных	Нет	Обязательно
Восстановимая база данных			

Тип восстановления	Работа менеджера связей данных DB2 при восстановлении	Работа менеджера связей данных DB2 при повторе транзакций	Согласование
Восстановление базы данных с использованием опции WITHOUT ROLLING FORWARD из полной резервной копии при работе всех серверов связей данных	Выполняется быстрое согласование	Нет	Необязательно
Восстановление базы данных с использованием опций WITHOUT ROLLING FORWARD и WITHOUT DATALINK из полной или неполной резервной копии при работающих или неработающих серверах связей данных	Таблицы переводятся в состояние отложенного согласования связей данных	Нет	Обязательно
Восстановление базы данных с использованием опции WITHOUT ROLLING FORWARD из полной резервной копии, когда хотя бы один из серверов связей данных не работает	Быстрое согласование выполняется лишь на тех таблицах в табличных пространствах, у которых нет связей с неработающими серверами связей данных, тогда как остальные таблицы переводятся в состояние отложенного согласования связей данных	Нет	Требуется на таблицах в табличных пространствах со связями с неработающими серверами связей данных

Тип восстановления	Работа менеджера связей данных DB2 при восстановлении	Работа менеджера связей данных DB2 при повторе транзакций	Согласование
Восстановление базы данных с использованием опции WITHOUT ROLLING FORWARD из неполной резервной копии при работающих или неработающих серверах связей данных	Быстрое согласование не производится, все таблицы со столбцами DATALINK переводятся в состояние отложенного согласования связей данных	Нет	Обязательно
Восстановление базы данных и повтор транзакций до конца журналов из полной резервной копии при работе всех серверов связей данных	Никаких действий	Никаких действий	Необязательно
Восстановление базы данных и повтор транзакций до конца журналов из полной резервной копии, когда при выполнении повтора транзакций хотя бы один из серверов связей данных не работает	Никаких действий	Никаких действий	Необязательно
Восстановление базы данных и повтор транзакций до конца журналов из полной или неполной резервной копии, когда во время восстановления любой из серверов связей данных не работает	Никаких действий	Все таблицы со столбцами DATALINK переводятся в состояние отложенного согласования связей данных	Требуется для всех таблиц со столбцами DATALINK

Тип восстановления	Работа менеджера связей данных DB2 при восстановлении	Работа менеджера связей данных DB2 при повторе транзакций	Согласование
Восстановление базы данных и повтор транзакций до конца журналов из неполной резервной копии при работе всех серверов связей данных во время восстановления	Никаких действий	Никаких действий	Необязательно
Восстановление базы данных и повтор транзакций до конца журналов из полной или неполной резервной копии при работе всех серверов связей данных, когда на любом из серверов связей данных резервная копия неизвестна	Никаких действий	Все таблицы в табличных пространствах со связями с сервером связей данных, на котором резервная копия неизвестна, переводятся в состояние отложенного согласования связей данных	Обязательно
Восстановление табличного пространства и повтор транзакций до конца журналов из полной резервной копии при работе всех серверов связей данных	Никаких действий	Никаких действий	Необязательно
Восстановление табличного пространства и повтор транзакций до конца журналов из полной резервной копии, когда при выполнении повтора транзакций хотя бы один из серверов связей данных не работает	Никаких действий	Никаких действий	Необязательно

Тип восстановления	Работа менеджера связей данных DB2 при восстановлении	Работа менеджера связей данных DB2 при повторе транзакций	Согласование
Восстановление табличного пространства и повтор транзакций до конца журналов из полной или неполной резервной копии, когда во время восстановления любой из серверов связей данных не работает	Никаких действий	Все таблицы в табличных пространствах со связями с любым неработающим сервером связей данных переводятся в состояние отложенного согласования связей данных.	Требуется для таблиц в табличных пространствах со связями с любым неработающим сервером связей данных
Восстановление табличного пространства и повтор транзакций до конца журналов из неполной резервной копии при работе всех серверов связей данных	Никаких действий	Никаких действий	Необязательно
Восстановление базы данных и повтор транзакций до определенного момента времени из полной или неполной резервной копии при работающих или не работающих серверах связей данных во время процесса восстановления и/или повтора транзакций	Никаких действий	Таблицы переводятся в состояние отложенного согласования связей данных	Обязательно

Тип восстановления	Работа менеджера связей данных DB2 при восстановлении	Работа менеджера связей данных DB2 при повторе транзакций	Согласование
Восстановление табличного пространства и повтор транзакций до определенного момента времени из полной или неполной резервной копии при работающих или не работающих серверах связей данных во время процесса восстановления и/или повтора транзакций	Никаких действий	Таблицы переводятся в состояние отложенного согласования связей данных	Обязательно
Восстановление базы данных с другим именем, алиасом, именем хоста или экземпляром без повтора транзакций (смотрите примечание 1 на стр. 83)	Таблицы переводятся в состояние невозможности согласования связей данных	Нет	Необязательно, но таблицы в состоянии невозможности согласования связей данных нужно исправить вручную
Восстановление базы данных с другим именем базы данных, алиасом, именем хоста или экземпляра и повтор транзакций	Никаких действий	Таблицы переводятся в состояние невозможности согласования связей данных	Необязательно, но таблицы в состоянии невозможности согласования связей данных нужно исправить вручную

Тип восстановления	Работа менеджера связей данных DB2 при восстановлении	Работа менеджера связей данных DB2 при повторе транзакций	Согласование
Восстановление базы данных из непригодной к использованию резервной копии (образ был удален при чистке мусора на сервере связей данных) без повтора транзакций (смотрите примечание 1 на стр. 83), с опцией WITHOUT DATALINK или без нее	Таблицы переводятся в состояние отложенного согласования связей данных	Никаких действий	Обязательно
Восстановление базы данных из непригодной к использованию резервной копии (образ был удален при чистке мусора на сервере связей данных) и повтор транзакций, с опцией WITHOUT DATALINK или без нее	Никаких действий	Таблицы переводятся в состояние отложенного согласования связей данных	Обязательно
Восстановление табличного пространства из непригодной к использованию резервной копии (образ был удален при чистке мусора на сервере связей данных) и повтор транзакций	Никаких действий	Таблицы переводятся в состояние отложенного согласования связей данных	Обязательно

Примечания:

1. Восстановление с использованием оперативной резервной копии и опции WITHOUT ROLLING FORWARD или восстановление с использованием автономной резервной копии.
2. *Полная* резервная копия - это резервная копия, снятая при всех работающих серверах связей данных. *Неполная* резервная копия - это резервная копия, снятая, когда хотя бы один из серверов связей данных не работал.
3. Процесс быстрого согласования не производится, если резервная копия, использованная для операции восстановления базы данных, была снята после восстановления базы данных - с повтором транзакций или без него. В этом случае все таблицы со столбцами DATALINK переводятся в состояние отложенного согласования связей данных.

Вывод таблицы из состояния невозможности согласования связей данных

Восстановленная таблица (или таблицы) со столбцом DATALINK переводится в состояние невозможности согласования связей данных, если табличное пространство восстановлено из резервной копии, сделанной раньше, чем значение, указанное для параметра конфигурации *num_db_backups*. Дополнительную информацию об этом параметре конфигурации смотрите в руководстве *Administration Guide: Performance*.

DB2 продолжает допускать доступ к таблице даже в том случае, когда данные столбца DATALINK могут быть неверными. Если вы хотите предотвратить доступ к таблице, в которой значения столбца DATALINK могут быть несовместимыми, введите SET CONSTRAINTS для *имя_таблицы* в команде DATALINK RECONCILE PENDING. Значения DATALINK можно исправить так:

- При помощи оператора SQL UPDATE установите для части столбца DATALINK с положениями данных URL нулевой длины, если этот столбец не допускает пустых значений, и NULL, если допускает.
- Восстановите файлы на необходимых серверах связей данных. Затем запустите прикладную программу, которая выполняет операторы SELECT для чтения значений столбца DATALINK и операторы UPDATE для обновления столбца DATALINK при помощи тех же самых значений. Обратите внимание на то, что при изменении значений столбцов DATALINK таблица должна находиться в состоянии невозможности согласования связей данных. После завершения операции обновления файлы будут помечены, как связанные на соответствующих серверах связей данных.

Затем отмените состояние невозможности согласования связей данных, введя следующую команду:

```
SET CONSTRAINTS FOR имя_таблицы DATALINK RECONCILE PENDING IMMEDIATE UNCHECKED
```

Синхронизация связей данных

Для синхронизации связей данных используется утилита синхронизации. Эта утилита запускается из DB2 и затрагивает все работающие серверы связей данных для менеджера связей данных DB2, на которые есть ссылки в значениях столбцов DATALINK. Она проверяет, что упомянутые файлы либо существуют на сервере связей данных, либо эти связи могут быть переустановлены. В следующих разделах описано, как DB2 определяет, надо ли вам синхронизировать связи данных, и как это сделать.

Если ссылка на файл на сервере связей данных не существует или не может быть переустановлена, утилита синхронизации помещает строки с ошибками, а также причины всех ошибок в таблицу исключений (если она указана), а затем модифицирует эти строки. Если таблица исключений не указана, значения столбцов DATALINK, для которых ссылка на файл не может быть восстановлена, копируются в файл отчета об исключениях вместе с ID столбца и причиной. Информацию из таблицы исключений (если она указана) или из отчета можно использовать для необходимого исправления строк. Таблица исключений, используемая с утилитой синхронизации, идентична таблице исключений, используемой утилитой загрузки. Дополнительную информацию об утилите загрузки смотрите в руководстве *Data Movement Utilities Guide and Reference*. Имя отчета имеет вид *отчет*.exp (расширение `.exp` добавляет утилита синхронизации). Например, можно вызвать утилиту синхронизации при помощи следующего оператора:

```
db2 RECONCILE dept DLREPORT  /u/scottba/report FOR EXCEPTION excptab
```

Эта команда синхронизирует таблицу с именем `dept` и записывает исключения в таблицу исключений `excptab`, которая была создана пользователем.Информация о файлах, для которых во время синхронизации были разорваны связи, записывается в файл `report.ulk`, создаваемый в каталоге `/u/scottba`.Если не указано *FOR EXCEPTION excptab*, информация об исключениях записывается в файл `report.exp`, который создается в каталоге `/u/scottba`. Дополнительную информацию об утилите согласования смотрите в руководстве *Command Reference*.

Как обнаружить, что требуется синхронизация

Далее приводятся несколько ситуаций, в которых может потребоваться запуск утилиты согласования:

- Вся база данных восстановлена, и для нее произведен повтор транзакций до определенного момента времени. Поскольку для всей базы данных был выполнен повтор транзакций до принятой транзакции, ни одна из таблиц не будет находиться в состоянии ожидания проверки (из-за реляционных или проверочных ограничений). Все данные в базе данных приведены в согласованное состояние. Тем не менее, столбцы DATALINK могут быть не синхронизированы с метаданными в Менеджере связей данных DB2, и требуется согласование.

В этом случае таблицы с данными столбцов DATALINK уже будут в состоянии DRP (ожидание согласования связей данных). Для каждой из этих таблиц необходимо вызвать утилиту согласования.

- Один из серверов связей данных, на котором запущен менеджер связей DB2, теряет след своих метаданных. Это может произойти по нескольким причинам. Например:
  - После холодного старта сервера связей данных.
  - После восстановления метаданных сервера связей данных до более раннего состояния.

  В некоторых ситуациях, например, при выполнении операторов SQL UPDATE и DELETE, DB2 может быть в состоянии обнаружить ошибку метаданных на сервере связей. В таких ситуациях DB2 откажется обработать оператор SQL. Вам придется перевести таблицу в состояние DRP (ожидание согласования связей данных) при помощи оператора SET CONSTRAINTS, а затем запустить утилиту согласования для этой таблицы.
- Файловая система недоступна (например, из-за отказа диска) и не восстанавливается до текущего состояния. В этой ситуации могут отсутствовать отдельные файлы.
- Менеджер связей DB2 отброшен в базе данных, но в столбце DATALINK FILE LINK CONTROL есть ссылки на этот менеджер связей DB2. Для таких таблиц следует запустить утилиту согласования.

Обзор процедуры синхронизации

При необходимости синхронизации связей данных из-за восстановления до определенного момента времени или из-за несоответствия управляющей информации на серверах связей данных с запущенным менеджером связей данных DB2 и в DB2:

1. Вам придется перевести таблицу в состояние ожидания согласования связей данных при помощи оператора SET CONSTRAINTS. (В некоторых случаях за вас это сделает DB2.)
2. Используйте утилиту синхронизации для разрешения связей и выполните необходимые действия в отношении исключений в таблице исключений или в отчете об исключениях.

Глава 2. Резервное копирование базы данных

В этом разделе описывается утилита резервного копирования DB2 UDB, используемая для создания резервных копий базы данных или табличных пространств.

Рассматриваются следующие темы:

Обзор резервного копирования

В простейшей форме команды DB2 BACKUP DATABASE требуется задать только алиас базы данных, для которой нужно создать резервную копию. Например:

```
db2 backup db sample
```

Если команда выполнена успешно, вы получите новый образ резервной копии в каталоге, откуда была запущена эта команда. Он будет расположен в том же каталоге, поскольку в этом примере положение образа резервной копии не было задано явно. Например, в Windows NT/2000 эта команда (запущенная из корневого каталога) создаст образ резервной копии, который будет выглядеть в списке файлов каталога так:

© Copyright IBM Corp. 2001

```
 Directory of D:\SAMPLE.0\DB2\NODE0000\CATN0000\20010320

03/20/2001  12:26p      <DIR>          .
03/20/2001  12:26p      <DIR>          ..
03/20/2001  12:27p          12,615,680 122644.001
```

Если при запуске утилиты резервного копирования задана опция, указывающая положение назначения, образы резервных копий создаются в указанном месте. Это положение может быть:

- Каталогом (для резервных копирований на диск или дискету)
- Устройством (для резервных копирований на ленту)
- Сервером Tivoli Storage Manager (TSM) (смотрите раздел “Приложение G. Tivoli Storage Manager” на стр. 465)
- Сервером другого поставщика
- Задаваться программой обработчика пользователя (только в OS/2).

При каждом выполнении резервного копирования целой базы данных в файле хронологии восстановления автоматически обновляется сводка информации. Этот файл создается в том же каталоге, что и файл конфигурации базы данных. Дополнительную информацию о файле хронологии восстановления смотрите в разделе “Что такое файл хронологии восстановления” на стр. 53.

В системах на основе UNIX имена файлов для созданных образов резервных копий состоят из нескольких элементов, разделенных точками:

```
алиас_базы_данных.тип.имя_экземпляра.NODEnnnn.CATNnnnn.отметка_времени
.порядковый_номер
```

Например:

```
STAFF.0.DB201.NODE0000.CATN0000.19950922120112.001
```

На других платформах используется четырехуровневое дерево подкаталогов:

```
алиас_базы_данных.тип\имя_экземпляра.NODEnnnn\CATNnnnn\ггггммдд\ччммсс
.порядковый_номер
```

Например (Windows NT/2000):

```
SAMPLE.0\DB2\NODE0000\CATN0000\20010320\122644.001
```

Алиас базы данных — Алиас базы данных (от 1 до 8 символов), заданный при запуске утилиты резервного копирования.

Тип — Тип операции резервного копирования; `0` - полное резервное копирование уровня базы данных, `3` - резервное копирование уровня табличного пространства, `4` - образ резервной копии, созданный командой LOAD...COPY TO.

Имя экземпляра — Имя текущего экземпляра (от 1 до 8 символов), взятое из переменной среды **DB2INSTANCE**.

Номер узла — Номер узла. В одноразделных системах баз данных это всегда `NODE0000`. В многораздельных системах баз данных это `NODE`*xxxx*, где *xxxx* - номер узла, заданный в файле `db2nodes.cfg`.

Номер узла каталога — Номер узла каталога базы данных. В одноразделных системах баз данных это всегда `CATN0000`. В многораздельных системах баз данных это `CATN`*xxxx*, где *xxxx* - номер узла, заданный в файле `db2nodes.cfg`.

Отметка времени — 14-символьное представление даты и времени выполнения резервного копирования в виде *ггггммддччннсс*, где:

- *гггг* - год (с 1995 до 9999)
- *мм* - месяц (от 01 до 12)
- *дд* - день месяца (от 01 до 31)
- *чч* - час (от 00 до 23)
- *нн* - минуты (от 00 до 59)
- *сс* - секунды (от 00 до 59)

Порядковый номер — Трехзначный номер, используемый в качестве расширения файла.

Если образ резервной копии записывается на ленту:

- Имена файлов не создаются, однако описанная выше информация сохраняется в заголовке резервной копии для целей проверки.
- Ленточное устройство должно быть доступно через стандартный интерфейс операционной системы. Однако в системе большой многораздельной базы данных может оказаться непрактичным назначать каждому серверу раздела базы данных отдельное ленточное устройство. Можно подключить ленточные устройства к одному или нескольким серверам TSM, чтобы обеспечить доступ к этим устройствам всем серверам разделов базы данных. Дополнительную информацию о TSM смотрите в разделе “Приложение G. Tivoli Storage Manager” на стр. 465.
- В системе многораздельных баз данных можно также использовать программные продукты, предоставляющие функции виртуальных ленточных устройств, такие как REELlibrarian 4.2 или CLIO/S. Эти программные продукты используются для доступа к ленточным устройствам, подключенным к другим узлам (серверам разделов базы данных) через эмуляцию ленточного устройства. Доступ к удаленным ленточным

устройствам производится прозрачным образом, а доступ к эмуляциям ленточных устройств производится через стандартный интерфейс операционной системы.

Привилегии, полномочия и авторизация, необходимые для использования резервного копирования

Привилегии позволяют пользователям создавать ресурсы баз данных и обращаться к ним. Уровни полномочий дают способ группировки привилегий и высокоуровневой поддержи менеджера баз данных и операций утилит. Все это вместе позволяет управлять доступом к менеджеру баз данных и его объектам баз данных. Пользователям доступны только объекты, для которых у них есть соответствующие права - то есть требуемые привилегии или полномочия.

Для использования утилиты резервного копирования у вас должны быть полномочия SYSADM, SYSCTRL или SYSMAINT.

Использование резервного копирования

Прежде, чем использовать резервное копирование

Вы не должны иметь соединение с базой данных, для которой делается резервная копия: утилита резервного копирования автоматически установит соединение с указанной базой данных и, выполнив резервное копирование, завершит это соединение.

База данных может быть локальной или удаленной. Образ резервной копии остается на сервере базы данных, если только не используется программный продукт управления хранением, например, Tivoli Storage Manager (TSM).

В системе многораздельных баз данных резервное копирование выполняется отдельно для каждого раздела базы данных. Эта операция является локальной для того сервера раздела базы данных, на котором запущена эта утилита. Однако можно ввести команду **db2_all** с одного из серверов разделов базы данных в экземпляре для запуска утилиты резервного копирования на списке серверов, заданных номерами их узлов. (Для задания списка узлов или серверов разделов базы данных, на которых есть пользовательские таблицы, используйте команду LIST NODES. Информацию о команде LIST NODES смотрите в справочнике *Command Reference*.) В этом случае сначала следует сделать резервную копию узла каталога, а затем - других разделов базы данных. Для резервного копирования разделов базы данных можно также использовать Командный центр. Поскольку при таком подходе не поддерживается восстановление с повтором транзакций, регулярно создавайте резервные копии базы данных, расположенной на таких узлах. Кроме того, вместе с каждой созданной резервной копией необходимо сохранять копию файла `db2nodes.cfg` - для защиты от возможного повреждения этого файла.

В системе с распределенными требованиями операции резервного копирования применяются к базе данных с распределенными требованиями и к метаданным, хранящимся в каталоге этой базы данных (оболочки, серверы, псевдонимы и т.п.) Если объекты источника данных (таблицы и производные таблицы) не хранятся в базе данных с распределенными требованиями, их резервное копирование не выполняется.

Если база данных была создана при помощи предыдущего выпуска менеджера баз данных и впоследствии не перенастроена, перед ее резервным копированием необходимо произвести перенастройку. Информацию о перенастройке баз данных смотрите в книге *Administration Guide: Planning*.

Запуск резервного копирования

Утилиту резервного копирования можно запустить:

- Из командной строки.

  Пример команды BACKUP DATABASE, введенной в командной строке:

  ```
  db2 backup database sample to c:\DB2Backups
  ```

- Из записной книжки Резервное копирование базы данных или из мастера по резервному копированию базы данных в Центре управления. Чтобы открыть эту записную книжку или мастер:

  1. В Центре управления раскрывайте дерево объектов, пока не найдете папку Базы данных.
  2. Щелкните по папке Базы данных. Все существующие базы данных будут показаны на панели содержимого в правой части окна.
  3. Щелкните правой кнопкой мыши по нужной базе данных на панели содержимого и выберите из всплывающего меню Резервное копирование базы данных или Резервное копирование базы данных при помощи мастера. Откроется записная книжка Резервное копирование базы данных или мастер по резервному копированию базы данных.

  Общую информацию о Центре управления смотрите в книге *Administration Guide*. Подробную информацию можно получить в электронной справке Центра управления.

- При помощи API **sqlubkp**. Информацию об этом API смотрите в разделе “API резервного копирования базы данных” на стр. 99. Общую информацию о создании прикладных программ с API управления DB2 смотрите в книге *Application Building Guide*.

Вывод информации о резервной копии

Команду **db2ckbkp** можно использовать для вывода информации о существующих образах резервных копий. Эта утилита позволяет:

- Проверить целостность образа резервной копии и определять, можно ли произвести восстановление.

Вывод информации о резервной копии

- Вывести информацию, хранящуюся в заголовке резервной копии.

Подробную информацию об этой утилите смотрите в разделе “db2ckbkp - Проверка резервной копии” на стр. 330.

Резервное копирование на ленту

При резервном копировании базы данных или табличного пространства необходимо правильно задать размер блока и размер буфера. Особая необходимость в этом возникает при использовании переменного размера блока (например, когда в AIX задан нулевой размер блока).

При резервном копировании в качестве фиксированного размера блока можно использовать только некоторые значения. Это ограничение вызвано тем, что DB2 записывает заголовок образа резервной копии как блок размером 4 Кбайта. DB2 поддерживает только фиксированные блоки с размерами 512, 1024, 2048 и 4096 байт. При использовании фиксированного размера блока для резервной копии можно использовать любой размер буфера. Однако если фиксированный размер блока не равен одному из поддерживаемых DB2 размеров, резервное копирование может завершиться неудачно.

При использовании фиксированного размера блока для большой базы данных операции резервного копирования будут занимать много времени. В этом случае, возможно, лучше использовать переменный размера блока.

Примечание: Использование переменного размера блока в настоящее время *не* поддерживается. Если вам необходимо использовать такие блоки, вы должны хорошо протестировать процедуры восстановления, чтобы быть уверенными, что сможете успешно использовать для восстановления образы резервных копий, созданные с переменным размером блока.

При использовании переменного размера блока необходимо задать размер буфера для резервного копирования - меньший или равный предельному значению для используемых ленточных устройств. Максимальную производительность дает размер буфера, равный максимальному размеру блока для используемого устройства.

При восстановлении из образа резервной копии с переменным размером блока может возникнуть ошибка. В таком случае, возможно, придется переписать этот образ с использованием подходящего размера блока. Пример для AIX:

```
tcl -b 0 -Bn -f /dev/rmt0 read > backup_filename.file
dd if=backup_filename.file of=/dev/rmt0/ obs=4096 conv=sync
```

Образ резервной копии записывается в файл с именем `backup_filenam.file`. Команда **dd** выгружает этот образ на ленту, используя размер блока 4096.

Такой подход может быть неприменим, если образ резервной копии слишком велик для выгрузки в файл. Одно из возможных решений - использовать команду **dd** для выгрузки образа с одного ленточного устройства на другое. Это возможно, если образ занимает не более одной магнитной ленты. При использовании двух ленточных устройств команда команда **dd** выглядит так:

```
dd if=/dev/rmt1 of=/dev/rmt0 obs=4096
```

Если использование двух ленточных устройств невозможно, можно попробовать при помощи команды **dd** записать образ на непосредственное устройство, а затем - с него на ленту. Трудность использования этого метода в том, что команда **dd** *должна* следить за числом блоков, записываемых на непосредственное устройство. Это число должно быть указано при обратном перемещении образа на ленту. Если для выгрузки образа с непосредственного устройства используется команда **dd**, на ленту выгружается все содержимое непосредственного устройства. Команда **dd** не может определить, какая часть непосредственного устройства использована для сохранения образа.

При использовании утилиты резервного копирования необходимо знать максимальные размеры блоков для используемых ленточных устройств. Несколько примеров:

Устройство	Подключение	Максимальный размер блока	Максимальный размер буфера DB2 (в страницах по 4 Кбайта)
8 mm	scsi	131072	32
3420	s370	65536	16
3480	s370	65536	16
3490	s370	65536	16
3490E	s370	65536	16
7332 (4 mm)[1]	scsi	262144	64
3490e	scsi	262144	64
3590[2]	scsi	2097152	512
3570 (magstar MP)		262144	64

Примечания:

1. Устройство 7332 не использует максимальный размер блока. 256 Кбайт - просто предлагаемое значение. Максимальный размер блока определяется родительским адаптером.
2. Поскольку 3590 не поддерживает блоки размером 2 Мбайта, можно попробовать меньшие значениям (например, 256 Кбайт), обеспечивающие требуемую производительность.

3. Максимальный размер блока для конкретного устройства можно узнать в документации на это устройство или у поставщика.

Резервное копирование в именованные конвейеры

В системах на основе UNIX теперь поддерживается резервное копирование в локальные именованные конвейеры и восстановление базы данных из таких копий. Процесс, записывающий данные в именованный конвейер, и процесс, читающий из него данные, должны находиться на одном компьютере. Конвейер должен существовать и должен находиться в локальной файловой системе. Поскольку именованный конвейер воспринимается как локальное устройство, не нужно специально указывать, что данные записываются в конвейер. Пример для AIX:

1. Создаем именованный конвейер:

   ```
   mkfifo /u/dmcinnis/mypipe
   ```

2. Используем этот конвейер в качестве назначения для операции резервного копирования базы данных:

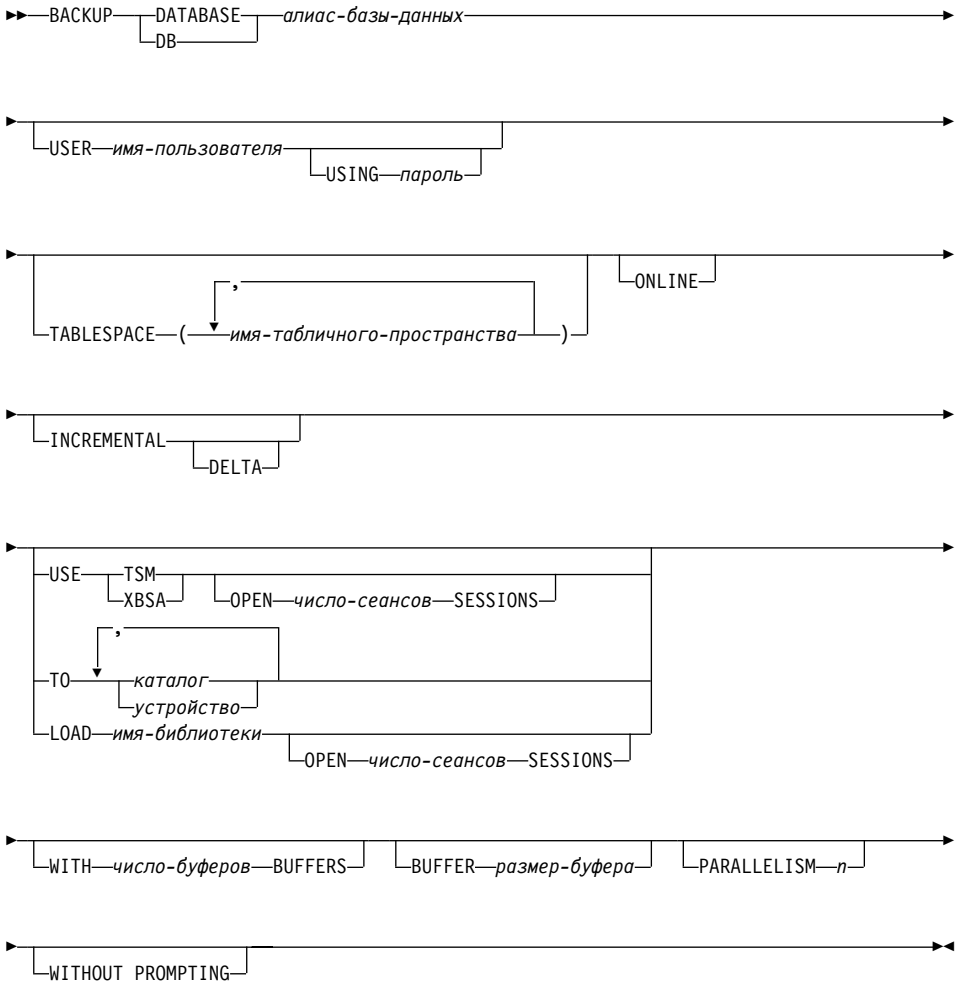
   ```
   db2 backup db sample to /u/dmcinnis/mypipe
   ```

3. Если этот образ резервной копии должен использоваться утилитой восстановления, операция восстановления должна быть запущена *перед* операцией резервного копирования, чтобы она получила все данные:

   ```
   db2 restore db sample into mynewdb from /u/dmcinnis/mypipe
   ```

Команда BACKUP DATABASE

Синтаксис команды



Параметры команды

DATABASE алиас-базы-данных
Задает алиас базы данных для резервного копирования.

USER имя-пользователя
Задает имя пользователя для резервного копирования базы данных.

USING пароль
Пароль, используемый для аутентификации имени пользователя. Если пароль отсутствует, пользователю будет предложено ввести его.

TABLESPACE имя-табличного-пространства

Список имен, используемых для указания табличных пространств для резервного копирования.

ONLINE

Задает оперативное резервное копирование. По умолчанию используется автономное резервное копирование. Оперативное резервное копирование доступно только для баз данных, в конфигурации которых включен параметр *logretain* или *userexit*.

Примечание: Операция оперативного резервного копирования может продолжаться дольше заданного времени, если для `sysibm.systables` есть блокировка IX, поскольку утилите резервного копирования DB2 требуется блокировка S для объектов, содержащих большие объекты.

INCREMENTAL

Задает инкрементную резервную копию. Инкрементная резервная копия - это копия всех данных базы данных, которые изменились со времени последней успешной операции полного резервного копирования.

DELTA

Задает разностную резервную копию. Разностная резервная копия - копия всех данных базы данных, которые изменились со времени последней успешной операции резервного копирования любого типа.

USE TSM

Указывает, что резервная копия будет использовать вывод Tivoli Storage Manager (прежнее название - ADSM).

OPEN число-сеансов SESSIONS

Число сеансов ввода-вывода, которые будут созданы между DB2 и TSM или другим устройством поставщика для резервного копирования.

Примечание: Этот параметр не учитывается при резервном копировании на ленту, диск или другое локальное устройство.

USE XBSA

Указывает, что будет использоваться интерфейс XBSA. API служб резервного копирования (XBSA) - открытый интерфейс прикладного программирования для прикладных программ или утилит, которым для целей резервного копирования или архивирования требуется управление хранением данных. Legato NetWorker - менеджер хранения, в настоящее время поддерживающий интерфейс XBSA.

TO каталог/устройство

Список имен каталогов или ленточных устройств хранения. Необходимо указывать полный путь к каталогу. Место назначения

должно находиться на сервере базы данных. Этот параметр можно повторять для указания каталогов и устройств назначения, на которых будет расположена резервная копия. Если указано несколько устройств назначения (например, target1, target2 и target3), первым будет открыто target1. Заголовок носителя и специальные файлы (включая файл конфигурации, таблицу табличных пространств и файл хронологии) помещаются на target1. Все остальные устройства открываются и затем используются параллельно во время операции резервного копирования. Поскольку в операционных системах OS/2 и Windows нет общей поддержки ленточных устройств хранения, для каждого из них требуется свой драйвер. При резервном копировании в файловую систему FAT в операционных системах OS/2 и Windows пользователи должны соблюдать ограничение на имена 8.3.

При использовании ленточных устройств хранения или дисководов для дискет могут генерироваться сообщения и подсказки для действий пользователя. Допустимые опции ответа:

c Продолжить. Продолжить использовать устройство, сгенерировавшее предупреждение (например, когда установлена новая магнитная лента)

d Завершить работу с устройством. Прекратить использовать *только* то устройство, которое сгенерировало предупреждающее сообщение (например, когда закончились магнитные ленты)

t Завершить. Прекратить операцию резервного копирования.

Ленточные устройства не поддерживаются в OS/2. В OS/2 можно указать `0` или `0:`, чтобы операция резервного копирования вызывала программу обработчика пользователя (смотрите раздел “Приложение H. Обработчик пользователя для восстановления баз данных” на стр. 471). База данных должна быть стабилизирована до запуска операции оперативного резервного копирования с программой обработчика пользователя. Утилита резервного копирования ожидает принятия или отката всех транзакций. Во время работы этой утилиты все новые транзакции ожидают завершения операции резервного копирования.

Если ленточная система хранения не поддерживает возможность уникальной ссылки на резервную копию, рекомендуется не хранить на одной ленте несколько резервных копий одной и той же базы данных.

LOAD имя-библиотеки

Имя совместно используемой библиотеки (в ø/2; или в Windows - библиотеки DLL), содержащей нужные функции ввода-вывода резервного копирования и восстановления поставщика. Можно вводить

полное имя. Если не вводить полное имя, по умолчанию используется путь, по которому расположена программа обработчика пользователя.

WITH число-буферов BUFFERS

Число буферов для использования. По умолчанию предполагается 2 буфера. Однако при создании резервной копии в нескольких положениях для повышения производительности можно использовать больше буферов.

BUFFER размер-буфера

Размер в страницах по 4 кбайта буфера, используемого при построении резервной копии. Минимальное значение для этого параметра - 8 страниц; по умолчанию используется 1024 страницы. Если указан нулевой размер буфера, для размера буфера используется значение параметра конфигурации менеджера баз данных *backbufsz*.

При использовании магнитной ленты с переменным размерами блоков уменьшите размер блока до диапазона, поддерживаемого устройством хранения на ленте. В противном случае операция резервного копирования может закончиться успешно, но полученная резервная копия не будет пригодна для восстановления.

При использовании устройств хранения на ленте с SCO UnixWare 7 задайте размер буфера 16.

В большинстве версий Linux использование устанавливаемого по умолчанию размера буфера DB2 для операций резервного копирования на SCSI-ленточные устройства приводит к ошибке SQL2025N с кодом причины 75. Для предотвращения переполнения внутренних буферов SCSI Linux рассчитайте число страниц по формуле:

```
(число-страниц-буфера) <= ST_MAX_BUFFERS * ST_BUFFER_BLOCKS / 4
```

где *число-страниц-буфера* - значение *backbufsz* или *restbufsz*, а `ST_MAX_BUFFERS` и `ST_BUFFER_BLOCKS` определены в ядре Linux в каталоге `drivers/scsi`.

PARALLELISM n

Определяет число табличных пространств, которые могут быть прочитаны параллельно утилитой резервного копирования. По умолчанию устанавливается 1.

WITHOUT PROMPTING

Указывает, что резервное копирование будет проходить без внешнего вмешательства и все действия, которые обычно требуют вмешательства пользователя, будут возвращать сообщение об ошибке.

API резервного копирования базы данных

Синтаксис API на языке C

```
/* Файл: sqlutil.h */
/* API: Резервное копирование базы данных */
/* ... */
SQL_API_RC SQL_API_FN
  sqlubkp (
    char * pDbAlias,
    sqluint32     BufferSize,
    sqluint32     BackupMode,
    sqluint32     BackupType,
    sqluint32     CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    sqluint32     NumBuffers,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaTargetList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    sqluint32 * pBackupSize,
    void * pReserved4,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

API резервного копирования базы данных

Общий синтаксис API

```
/* Файл: sqlutil.h */
/* API: Резервное копирование базы данных */
/* ... */
SQL_API_RC SQL_API_FN
  sqlgbkp (
    unsigned short DbAliasLen,
    unsigned short UserNameLen,
    unsigned short PasswordLen,
    unsigned short * pReserved1,
    char * pDbAlias,
    sqluint32 BufferSize,
    sqluint32 BackupMode,
    sqluint32 BackupType,
    sqluint32 CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    sqluint32 NumBuffers,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaTargetList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    sqluint32 * pBackupSize,
    void * pReserved4,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

Параметры API

DbAliasLen

Входной. 2-байтное целое без знака - длина алиаса базы данных в байтах.

UserNameLen

Входной. Двухбайтное целое число без знака - длина в байтах имени пользователя. Задайте значение ноль, если имя пользователя не указано.

PasswordLen

Входной. Двухбайтное целое число без знака - длина пароля в байтах. Задайте значение ноль, если пароль не указан.

pReserved1.

Зарезервирован для будущего использования.

pDbAlias

Входной. Строка, содержащая алиас базы данных (указанный в системном каталоге баз данных), для которой нужно выполнить резервное копирование.

BufferSize

Входной. Размер буфера резервного копирования в страницах по 4 Кбайта. Минимальное значение - 8 страниц. Значение по умолчанию - 1024 страницы.

BackupMode

Входной. Задает режим резервного копирования. Допустимые значения (определены в `sqlutil`):

SQLUB_OFFLINE

В автономном режиме; только утилита резервного копирования может соединяться с базой данных.

SQLUB_ONLINE

В оперативном режиме; при выполнении резервного копирования другим прикладным программам разрешено соединяться с базой данных.

Примечание: Если при выполнении резервного копирования в рабочем режиме в `sysibm.systables` есть блокировка IX, может истечь срок ожидания, так как утилита резервного копирования DB2 устанавливает блокировки S для больших объектов SMS и блокировки IN для всех остальных объектов.

BackupType

Входной. Задает требуемый тип резервного копирования. Допустимые значения (определены в `sqlutil`):

SQLUB_FULL

Задает полное (неинкрементное) копирование базы данных. Это значение не будет поддерживаться в будущем. Для задания полного копирования базы данных используйте значение `SQLUB_DB`.

SQLUB_DB

Задает резервное копирование всех табличных пространств базы данных.

SQLUB_TABLESPACE

Задает резервное копирование уровня табличного пространства. В параметре *pTablespaceList* задайте список табличных пространств, для которых нужно сделать резервные копии.

SQLUB_INCREMENTAL
Задает создание образа инкрементной резервной копии. Образ инкрементной резервной копии - это копия всех данных в базе данных, измененных после выполнения последней успешной операции полного резервного копирования.

SQLUB_DELTA
Задает создание образа разностной резервной копии. Образ разностной резервной копии - это копия всех данных в базе данных, измененных после выполнения последней успешной операции резервного копирования любого типа.

CallerAction
Входной. Задает выполняемое действие. Допустимые значения (определены в `sqlutil`):

SQLUB_BACKUP
Запустить операцию резервного копирования.

SQLUB_NOINTERRUPT
Запустить операцию резервного копирования. Задает, что операция резервного копирования будет выполняться в автоматическом режиме. Для сценариев, в которых обычно требуется вмешательство пользователя, будет предпринята попытка выполнения без возвращения к вызывающей программе или будет выдана ошибка. Это значение параметра CallerAction можно использовать, например, когда вы знаете, что все необходимые носители для операции резервного копирования смонтированы, а подсказки утилиты не желательны.

SQLUB_CONTINUE
Продолжить операцию резервного копирования после того, как пользователь выполнил действия, запрошенные утилитой резервного копирования (например, продолжить после монтирования новой ленты).

SQLUB_TERMINATE
Прервать операцию резервного копирования после того, как пользователь не смог выполнить действия, запрошенные утилитой резервного копирования.

SQLUB_DEVICE_TERMINATE
Удалить конкретное устройство из списка устройств, используемых утилитой резервного копирования. Когда один из носителей переполнен, утилита резервного копирования возвращает предупреждение вызывающей программе (продолжая выполнять обработку с использованием оставшихся устройств). Снова запустите утилиту резервного копирования с

этим значением параметра CallerAction, чтобы удалить устройство, для которого выдано такое предупреждение, из списка используемых устройств.

SQLUB_PARM_CHECK
Используется для проверки параметров без выполнения операции резервного копирования. Эта опция не вызывает завершения соединения с базой данных после возвращения из вызова. После успешного возврата из этого вызова ожидается, что пользователь выдаст вызов со значением SQLUB_CONTINUE для выполнения действия.

SQLUB_PARM_CHECK_ONLY
Используется для проверки параметров без выполнения операции резервного копирования. Перед возвращением из вызова завершается установленное этим вызовом соединение с базой данных и последующие вызовы не требуются.

pApplicationId
Выходной. Буфер длины SQLU_APPLID_LEN+1 (это значение определено в sqlutil). В этом буфере API возвращает строку, которая указывает агента, обслуживающего эту программу. Может использоваться для получения информации о процессе операции резервного копирования с помощью монитора базы данных.

pTimestamp
Выходной. Буфер длины SQLU_TIME_STAMP_LEN+1 (это значение определено в sqlutil). В этом буфере API возвращает значение отметки времени для образа резервной копии.

NumBuffers
Входной. Задает число используемых буферов резервного копирования.

pTablespaceList
Входной. Список табличных пространств, для которых нужно сделать резервные копии. Требуется только для операций резервного копирования уровня табличного пространства. Смотрите раздел “Структура данных: SQLU-TABLESPACE-BKRST-LIST” на стр. 112.

pMediaTargetList
Входной. В этой структуре вызывающая программа может задать назначение для операции резервного копирования. Передаваемая информация зависит от значения поля *media_type*. Допустимые значения *media_type* (определенные в sqlutil):

SQLU_LOCAL_MEDIA
Локальные устройства (набор лент, дисков или дискет). Передается список структур *sqlu_media_entry*. В операционных системах OS/2 или Windows этот список может содержать только имена каталогов, но не имена ленточных устройств.

SQLU_TSM_MEDIA
TSM. Если для задания пути для образа резервной копии не используется структура *sqlu_media_entry*, присвойте значение NULL указателю *media* в структуре *sqlu_media_list_targets*. Применяется поставляемая с DB2 совместно используемая библиотека TSM. Если нужно использовать другую версию совместно используемой библиотеки TSM, задайте имя совместно используемой библиотеки в SQLU_OTHER_MEDIA.

SQLU_OTHER_MEDIA
Продукт другого поставщика. Задайте имя этой совместно используемой библиотеки в структуре *sqlu_vendor*.

SQLU_USER_EXIT
Обработчик пользователя. Дополнительные входные параметры не требуются (можно использовать только в OS/2).

Смотрите раздел “Структура данных: SQLU-MEDIA-LIST” на стр. 108.

pUserName
Входной. Строка с именем пользователя, используемым при установке соединения.

pPassword
Входной. Строка с паролем, используемым с этим именем пользователя.

pReserved2
Зарезервирован для будущего использования.

VendorOptionsSize
Входной. Длина поля *pVendorOptions*; не должна превышать 65535 байт.

pVendorOptions
Входной. Используется для передачи информации от прикладной программы к функции поставщика. Это должна быть плоская структура данных (то есть косвенные уровни не поддерживаются). Имейте в виду, что обращение байтов не выполняется, а кодовая страница для этих данных не проверяется.

Parallelism
Входной. Степень параллелизма (число манипуляторов буферов).

pBackupSize
Выходной. Размер образа резервной копии (в Мбайтах). Может иметь пустое значение (NULL).

pReserved4
Зарезервирован для будущего использования.

pReserved3
Зарезервирован для будущего использования.

pSqlca Выходной. Указатель на структуру *sqlca*. Дополнительную информацию об этой структуре смотрите в публикациях *Administrative API Reference* или *SQL Reference*.

Синтаксис API REXX

```
BACKUP DATABASE алиас-базы-данных USING :значение [USER имя-пользователя
USING пароль]

[TABLESPACE :имена-табличных-пространств] [ONLINE]

[LOAD библиотека-поставщика [OPTIONS опции-поставщика] [OPEN число-сеансов
SESSIONS] | TO :область-назначения |
USE TSM [OPEN число-сеансов SESSIONS] |
USER_EXIT]

[ACTION действие] [WITH число-буферов BUFFERS] [BUFFERSIZE размер-буфера]
[PARALLELISM степень-параллелизма]
```

Параметры API REXX

алиас-базы-данных
Алиас базы данных, для которой выполняется резервное копирование.

значение
Составная переменная хоста REXX, в которую возвращается информация о резервном копировании базы данных. Далее XXX будет обозначать имя этой переменной хоста:

XXX.0 Число элементов в этих переменных (всегда 2)

XXX.1 Отметка времени для образа резервной копии

XXX.2 ID программы, указывающий агента, обслуживающего эту прикладную программу.

имя-пользователя
Указывает имя пользователя, под которым выполняется резервное копирование базы данных.

пароль Пароль, используемый для аутентификации имени пользователя.

имена-табличных-пространств
Составная переменная хоста REXX, содержащая список табличных пространств, для которых нужно сделать резервные копии. Далее XXX будет обозначать имя этой переменной хоста:

XXX.0 Число табличных пространств, для которых нужно сделать резервные копии

XXX.1 Имя первого табличного пространства

XXX.2 Имя второго табличного пространства

XXX.3 и так далее.

библиотека-поставщика
Имя совместно используемой библиотеки (DLL в операционных системах Windows или OS/2), содержащей используемые функции поставщика для операций резервного копирования и восстановления. Может содержать полный путь. Если полный путь не задан, по умолчанию используется путь программ обработчиков пользователя.

опции-поставщика
Информация, требуемая функциям поставщика.

число-сеансов
Число сеансов ввода-вывода, используемых TSM или продуктом поставщика.

область-назначения
Локальные устройства. Может задавать набор лент, дисков или дискет. Задайте список, как указано в разделе “Структура данных: SQLU-MEDIA-LIST” на стр. 108. В операционных системах OS/2 или Windows этот список может содержать только имена каталогов, но не имена ленточных устройств.

действие
Задает выполняемое действие. Допустимые значения:

SQLUB_BACKUP
Запустить операцию резервного копирования.

SQLUB_NOINTERRUPT
Запустить операцию резервного копирования. Задает, что операция резервного копирования будет выполняться в автоматическом режиме. Для сценариев, в которых обычно требуется вмешательство пользователя, будет предпринята попытка выполнения без возвращения к вызывающей программе или будет выдана ошибка. Это значение параметра CallerAction можно использовать, например, когда вы знаете, что все необходимые носители для операции резервного копирования смонтированы, а запросы утилиты не желательны.

SQLUB_CONTINUE
Продолжить операцию резервного копирования после того, как пользователь выполнил действия, запрошенные утилитой резервного копирования (например, продолжить после монтирования новой ленты).

SQLUB_TERMINATE
Прервать операцию резервного копирования после того, как

пользователь не смог выполнить действия, запрошенные утилитой резервного копирования.

SQLUB_DEVICE_TERMINATE
Удалить конкретное устройство из списка устройств, используемых утилитой резервного копирования. Когда один из носителей переполнен, утилита резервного копирования возвращает предупреждение вызывающей программе (продолжая выполнять обработку с использованием оставшихся устройств). Снова запустите утилиту резервного копирования с этим значением параметра CallerAction, чтобы удалить устройство, для которого выдано такое предупреждение, из списка используемых устройств.

SQLUB_PARM_CHECK
Используется для проверки параметров без выполнения операции резервного копирования.

число-буферов
Число используемых буферов резервного копирования.

размер-буфера
Размер буфера резервного копирования в страницах по 4 Кбайта. Минимальное значение - 8 страниц.

степень-параллелизма
Степень параллелизма (число манипуляторов буферов).

Структура данных: SQLU-MEDIA-LIST

Эта структура используется, чтобы:

- Хранить список носителей *назначения* резервной копии (смотрите раздел “API резервного копирования базы данных” на стр. 99).
- Хранить список носителей *источника* резервной копии (смотрите раздел “API восстановления базы данных” на стр. 129).
- Передавать информацию утилите загрузки DB2.

Таблица 1. Поля структуры SQLU-MEDIA-LIST

Имя поля	Тип данных	Описание
MEDIA_TYPE	CHAR(1)	Символ, обозначающий тип носителя.
SESSIONS	INTEGER	Обозначает число элементов в массиве, на который указывает поле *target* данной структуры.
TARGET	Union	Это поле - указатель на один из трех типов структур. Тип структуры, на которую он указывает, определяется значением поля *media_type*. Дополнительную информацию о заполнении этого поля смотрите в соответствующем API.

Таблица 2. Поля структуры SQLU-MEDIA-LIST-TARGETS

Имя поля	Тип данных	Описание
MEDIA	Указатель	Указатель на структуру *sqlu_media_entry*.
VENDOR	Указатель	Указатель на структуру *sqlu_vendor*.
LOCATION	Указатель	Указатель на структуру *sqlu_location_entry*.

Таблица 3. Поля структуры SQLU-MEDIA-ENTRY

Имя поля	Тип данных	Описание
RESERVE_LEN	INTEGER	Длина поля *media_entry*. Для языков, отличных от C.
MEDIA_ENTRY	CHAR(215)	Путь к резервной копии, используемый утилитами резервного копирования и восстановления.

Таблица 4. Поля структуры SQLU-VENDOR

Имя поля	Тип данных	Описание
RESERVE_LEN1	INTEGER	Длина поля *shr_lib*. Для языков, отличных от C.
SHR_LIB	CHAR(255)	Имя совместно используемой библиотеки поставщика для сохранения или получения данных.

Таблица 4. Поля структуры SQLU-VENDOR (продолжение)

Имя поля	Тип данных	Описание
RESERVE_LEN2	INTEGER	Длина поля *filename*. Для языков, отличных от C.
FILENAME	CHAR(255)	Имя файла, задающее источник для загрузки при применении совместно используемой библиотеки.

Таблица 5. Поля структуры SQLU-LOCATION-ENTRY

Имя поля	Тип данных	Описание
RESERVE_LEN	INTEGER	Длина поля *location_entry*. Для языков, отличных от C.
LOCATION_ENTRY	CHAR(256)	Имя файлов входных данных для утилиты загрузки.

Допустимые значения *MEDIA_TYPE* (определенные в `sqlutil`):

SQLU_LOCAL_MEDIA
Локальные устройства (ленты, диски или дискеты)

SQLU_SERVER_LOCATION
Устройства на сервере (ленты, диски или дискеты; только для загрузки). Задается только для параметра *pDataFileList*.

SQLU_TSM_MEDIA
TSM

SQLU_OTHER_MEDIA
Библиотека поставщика

SQLU_USER_EXIT
Обработчик пользователя (только для OS/2)

SQLU_PIPE_MEDIA
Именованный конвейер (только для API поставщика)

SQLU_DISK_MEDIA
Диск (только для API поставщика)

SQLU_DISKETTE_MEDIA
Дискета (только для API поставщика)

SQLU_TAPE_MEDIA
Лента (только для API поставщика)

Структура данных: SQLU-MEDIA-LIST

Синтаксис языка

Структура на языке C

```
/* Файл: sqlutil.h */
/* Структура: SQLU-MEDIA-LIST */
/* ... */
typedef SQL_STRUCTURE sqlu_media_list
{
  char            media_type;
  char            filler[3];
  sqlint32            sessions;
  union sqlu_media_list_targets target;
} sqlu_media_list;
/* ... */

/* Файл: sqlutil.h */
/* Структура: SQLU-MEDIA-LIST-TARGETS */
/* ... */
union sqlu_media_list_targets
{
  struct sqlu_media_entry        *media;
  struct sqlu_vendor             *vendor;
  struct sqlu_location_entry     *location;
};
/* ... */

/* Файл: sqlutil.h */
/* Структура: SQLU-MEDIA-ENTRY */
/* ... */
typedef SQL_STRUCTURE sqlu_media_entry
{
  sqluint32       reserve_len;
  char            media_entry[SQLU_DB_DIR_LEN+1];
} sqlu_media_entry;
/* ... */

/* Файл: sqlutil.h */
/* Структура: SQLU-VENDOR */
/* ... */
typedef SQL_STRUCTURE sqlu_vendor
{
  sqluint32       reserve_len1;
  char            shr_lib[SQLU_SHR_LIB_LEN+1];
  sqluint32       reserve_len2;
  char            filename[SQLU_SHR_LIB_LEN+1];
} sqlu_vendor;
/* ... */
```

```
/* Файл: sqlutil.h */
/* Структура: SQLU-LOCATION-ENTRY */
/* ... */
typedef SQL_STRUCTURE sqlu_location_entry
{
  sqluint32       reserve_len;
  char            location_entry[SQLU_MEDIA_LOCATION_LEN+1];
} sqlu_location_entry;
/* ... */
```

Структура на языке COBOL

```
* Файл: sqlutil.cbl
01 SQLU-MEDIA-LIST.
    05 SQL-MEDIA-TYPE         PIC X.
    05 SQL-FILLER             PIC X(3).
    05 SQL-SESSIONS           PIC S9(9) COMP-5.
    05 SQL-TARGET.
        10 SQL-MEDIA          USAGE IS POINTER.
        10 SQL-VENDOR         REDEFINES SQL-MEDIA
        10 SQL-LOCATION       REDEFINES SQL-MEDIA
        10 FILLER             REDEFINES SQL-MEDIA
*

* Файл: sqlutil.cbl
01 SQLU-MEDIA-ENTRY.
    05 SQL-MEDENT-LEN         PIC 9(9) COMP-5.
    05 SQL-MEDIA-ENTRY        PIC X(215).
    05 FILLER                 PIC X.
*

* Файл: sqlutil.cbl
01 SQLU-VENDOR.
    05 SQL-SHRLIB-LEN         PIC 9(9) COMP-5.
    05 SQL-SHR-LIB            PIC X(255).
    05 FILLER                 PIC X.
    05 SQL-FILENAME-LEN       PIC 9(9) COMP-5.
    05 SQL-FILENAME           PIC X(255).
    05 FILLER                 PIC X.
*

* Файл: sqlutil.cbl
01 SQLU-LOCATION-ENTRY.
    05 SQL-LOCATION-LEN       PIC 9(9) COMP-5.
    05 SQL-LOCATION-ENTRY     PIC X(255).
    05 FILLER                 PIC X.
*
```

Структура данных: SQLU-TABLESPACE-BKRST-LIST

Эта структура используется для вывода списка имен табличных пространств.

Таблица 6. Поля структуры SQLU-TABLESPACE-BKRST-LIST

Имя поля	Тип данных	Описание
NUM_ENTRY	INTEGER	Число записей в списке, на который указывает поле *tablespace*.
TABLESPACE	Указатель	Указатель на структуру *sqlu_tablespace_entry*.

Таблица 7. Поля структуры SQLU-TABLESPACE-ENTRY

Имя поля	Тип данных	Описание
RESERVE_LEN	INTEGER	Длина символьной строки в поле *tablespace_entry*. Для языков, отличных от C.
TABLESPACE_ENTRY	CHAR(19)	Имя табличного пространства.

Синтаксис языка

Структура на языке C

```
/* Файл: sqlutil.h */
/* Структура: SQLU-TABLESPACE-BKRST-LIST */
/* ... */
typedef SQL_STRUCTURE sqlu_tablespace_bkrst_list
{
  long            num_entry;
  struct sqlu_tablespace_entry *tablespace;
} sqlu_tablespace_bkrst_list;
/* ... */

/* Файл: sqlutil.h */
/* Структура: SQLU-TABLESPACE-ENTRY */
/* ... */
typedef SQL_STRUCTURE sqlu_tablespace_entry
{
  sqluint32       reserve_len;
  char            tablespace_entry[SQLU_MAX_TBS_NAME_LEN+1];
  char            filler[1];
} sqlu_tablespace_entry;
/* ... */
```

Структура на языке COBOL

```
* Файл: sqlutil.cbl
01 SQLU-TABLESPACE-BKRST-LIST.
    05 SQL-NUM-ENTRY          PIC S9(9) COMP-5.
    05 SQL-TABLESPACE         USAGE IS POINTER.
*
```

Структура данных: SQLU-TABLESPACE-BKRST-LIST

```
* Файл: sqlutil.cbl
01 SQLU-TABLESPACE-ENTRY.
    05 SQL-TBSP-LEN           PIC 9(9) COMP-5.
    05 SQL-TABLESPACE-ENTRY   PIC X(18).
    05 FILLER                 PIC X.
    05 SQL-FILLER             PIC X(1).
*
```

Примеры сеансов резервного копирования

Примеры использования процессора командной строки

```
db2 backup database sample use tsm open 2 sessions with 4 buffers

db2 backup database payroll tablespace syscatspace, userspace1 to
   /dev/rmt0, /dev/rmt1 with 8 buffers without prompting
```

Следующий пример иллюстрирует стратегию еженедельного инкрементного резервного копирования восстановимой базы данных. Он включает в себя операцию еженедельного полного резервного копирования базы данных, операцию ежедневного разностного резервного копирования и операцию резервного копирования дважды в неделю:

```
(Sun) db2 backup db kdr use tsm
(Mon) db2 backup db kdr online incremental delta use tsm
(Tue) db2 backup db kdr online incremental delta use tsm
(Wed) db2 backup db kdr online incremental use tsm
(Thu) db2 backup db kdr online incremental delta use tsm
(Fri) db2 backup db kdr online incremental delta use tsm
(Sat) db2 backup db kdr online incremental use tsm
```

Пример командного сценария DB2 и информацию о его использовании можно найти в разделе “Приложение F. Сценарий восстановления CLP” на стр. 457.

Примеры использования API

Примеры программ, содержащих вызовы API DB2 и встроенные вызовы SQL, и информацию об их использовании можно найти в разделе “Приложение E. Программы примеров восстановления” на стр. 389.

Оптимизация производительности резервного копирования

Чтобы сократить время операции резервного копирования:

- Используйте резервное копирование уровня табличного пространства.

  С помощью опции TABLESPACE команды BACKUP DATABASE можно создать резервную копию часть базы данных (и впоследствии восстановить эту часть). Это облегчает управление данными таблиц, индексами и данными длинных полей или больших объектов в отдельных табличных пространствах.

- Увеличьте значение параметра PARALLELISM команды BACKUP DATABASE, чтобы он соответствовал числу табличных пространств, для которых создаются резервные копии.

  Параметр PARALLELISM определяет число процессов или потоков, запущенных для чтения из базы данных. Каждый такой процесс или поток работает с одним табличным пространством. После завершения резервного копирования этого табличного пространства процесс или поток запрашивает другое табличное пространство. Однако имейте в виду, что для каждого процесса или потока требуются дополнительные ресурсы памяти и

процессора, поэтому для загруженных систем следует оставить значение параметра PARALLELISM по умолчанию - 1.

- Увеличьте размер буфера резервного копирования.

  В идеале размер буфера резервного копирования должен быть кратен размеру экстента табличного пространства. Если у вас несколько табличных пространств с разными размерами экстентов, задайте значение, кратное самому большому размеру экстента.

- 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.

- Используйте несколько устройств назначения.

Ограничения на использование резервного копирования

Ограничения на применение утилиты резервного копирования:

- Операцию резервного копирования табличного пространства и операцию восстановления табличного пространства нельзя выполнять одновременно, даже если в них участвуют разные табличные пространства.
- Чтобы в среде многораздельной базы данных иметь возможность восстановления с повтором транзакций, необходимо регулярно делать резервные копии базы данных на узлах из списка, и нужно иметь по крайней мере одну резервную копию остальных узлов в системе (даже тех, на которых нет пользовательских данных для этой базы данных). Образ резервной копии раздела базы данных на сервере раздела базы данных, не содержащем пользовательских данных для этой базы данных, требуется в двух случаях:
  - Если после создания последней резервной копии в систему базы данных добавлен сервер раздела базы данных, и нужно выполнить на этом сервере восстановление с повтором транзакций.
  - Если используется восстановление до определенного момента времени, для которого требуется, чтобы все разделы базы данных в системе находились в состоянии отложенного повтора транзакций.

Устранение неисправностей при резервном копировании

Нельзя производить резервное копирование базы данных в состоянии, исключающем ее использование, за исключением базы данных в состоянии отложенного резервного копирования. Если какое-либо табличное пространство в базе данных находится в ненормальном состоянии, сделать резервную копию такой базы данных или табличного пространства нельзя, если только это не состояние отложенного резервного копирования.

Если из-за сбоя системы во время операции восстановления база данных или табличное пространство остались в частично восстановленном состоянии, перед

резервным копированием необходимо успешное завершить восстановление этой базы данных или этого табличного пространства.

Операция резервного копирования не будет выполнена успешно, если в списке табличных пространств, для которых делаются копии, содержится временное табличное пространство.

Утилита резервного копирования обеспечивает управление одновременностью для нескольких процессов, выполняющих резервное копирование разных баз данных. Это управление одновременностью удерживает устройства назначения резервного копирования открытыми до завершения всех операций резервного копирования. Если в операции резервного копирования возникла ошибка и открытый контейнер не удалось закрыть, другие операции резервного копирования, записывающие данные на то же устройство назначения, могут столкнуться с ошибками доступа. Для исправления таких ошибок доступа необходимо прекратить операцию резервного копирования, вызвавшую ошибки, и отключиться от устройства назначения. При использовании утилиты резервного копирования для одновременных операций резервного копирования на ленту надо задать для этих процессов запись данных на разные ленты.

Глава 3. Восстановление базы данных

В этом разделе описывается утилита восстановления DB2 UDB, предназначенная для воссоздания испорченных или поврежденных баз данных или табличных пространств, для которых были ранее сделаны резервные копии.

Рассматриваются следующие темы:

Обзор процедуры восстановления

В простейшей форме команды DB2 RESTORE DATABASE требуется лишь указать алиас базы данных, которую вы хотите восстановить. Например:

```
db2 restore db sample
```

В этом примере, так как база данных SAMPLE существует, будет возвращено следующее сообщение:

```
SQL2539W  Предупреждение! Восстанавливается существующая база, которая совпадает
с базой резервной копии.
Файлы базы данных будут удалены.
Хотите продолжить? (д/н)
```

Если вы ответите `д` и резервная копия базы данных SAMPLE существует, операция восстановления должна завершиться успешно.

Для операции восстановления базы данных требуется монопольное соединение, то есть при запуске этой операции никакие прикладные программы не могут

© Copyright IBM Corp. 2001

работать с базой данных и утилита восстановления запрещает другим прикладным программам доступ к базе данных до завершения восстановления. Однако восстановление табличного пространства может быть выполнено в оперативном режиме.

Табличное пространство нельзя использовать, пока операция восстановления с последующим повтором транзакций не завершиться успешно.

Если есть таблицы, занимающие несколько табличных пространств, резервное копирование и восстановление таких наборов табличных пространств необходимо производить совместно.

При выполнении частичного восстановления или восстановления поднабора можно использовать либо резервную копию уровня табличных пространств, либо полную копию уровня базы данных, выбрав из нее одно или несколько табличных пространств. Все файлы журналов, связанные с этими табличными пространствами, должны существовать со времени создания резервной копии.

Привилегии, полномочия и авторизация, необходимые для выполнения восстановления

Привилегии позволяют пользователям создавать ресурсы баз данных и обращаться к ним. Уровни полномочий дают способ группировки привилегий и высокоуровневой поддержи менеджера баз данных и операций утилит. Все это вместе позволяет управлять доступом к менеджеру баз данных и его объектам баз данных. Пользователям доступны только объекты, для которых у них есть соответствующие права - то есть требуемые привилегии или полномочия.

Для восстановления в *существующую* базу данных из полной резервной копии базы данных у вас должны быть полномочия SYSADM, SYSCTRL или SYSMAINT. Для восстановления в *новую* базу данных необходимы полномочия SYSADM или SYSCTRL.

Выполнение процедуры восстановления

Перед выполнением восстановления

Когда происходит восстановление в *существующую* базу данных, соединяться с базой, которую будут восстанавливать, не надо: утилита восстановления автоматически установит соединение с указанной базой данных и разорвет это соединение по завершении процедуры восстановления. Когда происходит восстановление в *новую* базу данных, для создания этой базы данных требуется подключение к экземпляру. Когда происходит восстановление в *новую удаленную* базу данных, сначала надо подсоединиться к экземпляру, где будет храниться новая база данных. После этого создать новую базу данных, указав кодовую страницу и территорию сервера.

База данных может быть локальной или удаленной.

Запуск восстановления

Утилиту восстановления можно вызвать:

- Из командной строки.

  Пример команды RESTORE DATABASE, введенной в командной строке:

  ```
  db2 restore db sample from D:\DB2Backups taken at 20010320122644
  ```

- Из записной книжки Восстановление базы данных или мастера из Центра управления. Чтобы открыть записную книжку Восстановление базы данных или мастер:

  1. В Центре управления раскрывайте дерево объектов, пока не найдете папку Базы данных.
  2. Щелкните по папке Базы данных. На правой панели окна (панели содержимого) будут показаны все существующие базы данных.
  3. Щелкните по нужной базе данных на панели содержимого правой кнопкой мыши и выберите из всплывающего меню Восстановить базу данных или Восстановить базу данных с помощью мастера. Откроется записная книжка Восстановление базы данных или мастер по восстановлению базу данных.

  Общую информацию о центре управления смотрите в книге *Administration Guide*. Подробную информацию можно получить в электронной справке Центра управления.

- при помощи интерфейса прикладного программирования (API) **sqlrestore**. Информацию об этом API смотрите в разделе “API восстановления базы данных” на стр. 129. Общую информацию о создании прикладных программ с API управления DB2 смотрите в книге *Application Building Guide*.

Переопределение контейнеров табличных пространств при операции восстановления (перенаправленное восстановление)

Во время операции резервного копирования базы данных записываются все контейнеры, связанные с копируемыми табличными пространствами. Во время операции восстановления для всех контейнеров, перечисленных в резервной копии, проверяется их существование и доступность. Если один или несколько контейнеров недоступны из-за ошибки носителя (или по какой-либо другой причине), операция восстановления закончится ошибкой. В этом случае, чтобы операция восстановления была выполнена успешно, ее надо перенаправить в другие контейнеры. DB2 поддерживает добавление, изменение и удаление контейнеров табличных пространств.

Для переопределения контейнеров табличных пространств можно указать параметр REDIRECT в команде RESTORE DATABASE или воспользоваться страницей Контейнеры в записной книжке Восстановление базы данных в

Центре управления. Примеры перенаправленного восстановления, использующие процессор командной строки, командный сценарий или интерфейс прикладного программирования, смотрите в разделе “Примеры сеансов восстановления” на стр. 140.

Во время операции перенаправленного восстановления контейнеры каталогов и файлов создаются автоматически, если они еще не существуют. Менеджер баз данных не создает автоматически контейнеры устройств.

Перенаправление контейнеров обеспечивает значительную гибкость в управлении контейнерами табличных пространств. Например, несмотря на то, что добавление контейнеров к табличным пространствам SMS не поддерживается, эту процедуру можно выполнить, указав дополнительный контейнер при вызове операции восстановления с перенаправлением. Аналогично можно переместить табличное пространство DMS из контейнеров файлов в контейнеры устройств.

Восстановление в существующую базу данных

Можно восстановить резервную копию всей базы данных в существующую базу данных. Резервная копия может отличаться от существующей базы данных именем алиаса, именем базы данных или уникальным номером базы данных.

Уникальный номер - это идентификатор базы данных, который не изменяется в течении всего времени жизни базы данных. Уникальный номер назначается менеджером баз данных при создании базы данных. Он не меняется после восстановления резервной копии, даже если резервная копия имела другой уникальный номер базы данных. DB2 всегда использует уникальный номер из образа резервной копии.

При восстановлении в существующую базу данных утилита восстановления:

- Стирает данные таблиц, индексов и длинных полей существующей базы данных и заменяет их данными из резервной копии.
- Заменяет записи таблиц для всех восстанавливаемых табличных пространств.
- Сохраняет файл хронологии восстановления, если он не поврежден. Если файл хронологии восстановления поврежден, менеджер баз данных копирует его из резервной копии.
- Сохраняет тип аутентификации для существующей базы данных.
- Сохраняет каталоги базы данных для существующей базы данных. Эти каталоги определяют, где находится база данных и как она каталогизирована.
- Сравнивает уникальные номера баз данных. Если уникальные номера различны:
  - Удаляет журналы, связанные с существующей базой данных.
  - Копирует файл конфигурации базы данных из резервной копии.

– Если в команде RESTORE DATABASE был задан параметр NEWLOGPATH, задает значение параметра *logpath* конфигурации базы данных.

Если уникальные номера баз данных совпадают:

– Удаляет журналы, если копия сделана с невосстановимой базы данных.
– Сохраняет текущий файл конфигурации базы данных, если этот файл не поврежден; в противном случае этот файл копируется из резервной копии.
– Если в команде RESTORE DATABASE было указано NEWLOGPATH, задает значение параметра *logpath* конфигурации базы данных; в противном случае копирует текущий путь журналов в файл конфигурации базы данных. Проверяет путь к журналу: если база данных не может использовать этот путь, изменяет в конфигурации базы данных путь к журналу на устанавливаемый по умолчанию.

Восстановление в новую базу данных

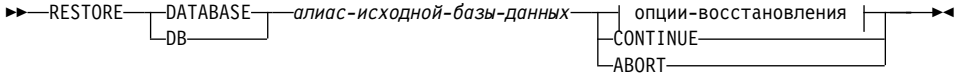
Можно создать новую базу данных, а потом восстановить в нее резервную копию всей базы данных. Кодовые страницы резервной копии и базы данных назначения должны совпадать.

При восстановлении в новую базу данных утилита восстановления:

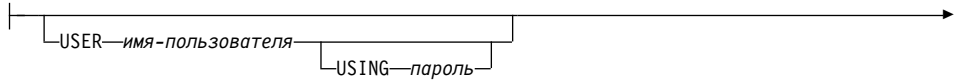
- Создает новую базу данных, используя имя алиаса базы данных, заданное в параметре "алиас базы данных назначения". (Если алиас базы данных назначения не был указан, утилита восстановления создает базу данных с алиасом, который задан в параметре "алиас исходной базы данных".)
- Копирует файл конфигурации базы данных из резервной копии.
- Если в команде RESTORE DATABASE был указан параметр NEWLOGPATH, задает значение параметра *logpath* конфигурации базы данных. Проверяет путь к журналу: если база данных не может использовать этот путь, изменяет в конфигурации базы данных путь к журналу на устанавливаемый по умолчанию.
- Восстанавливает тип авторизации из резервной копии.
- Восстанавливает комментарии из каталогов баз данных резервной копии.
- Восстанавливает файл хронологии восстановления для базы данных.

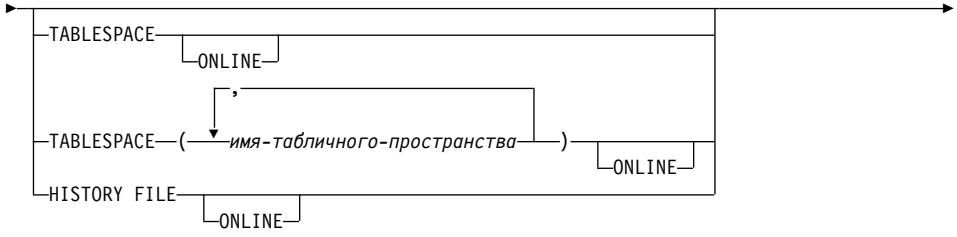
команда RESTORE DATABASE

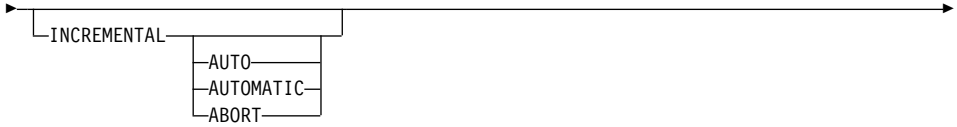
Синтаксис команды

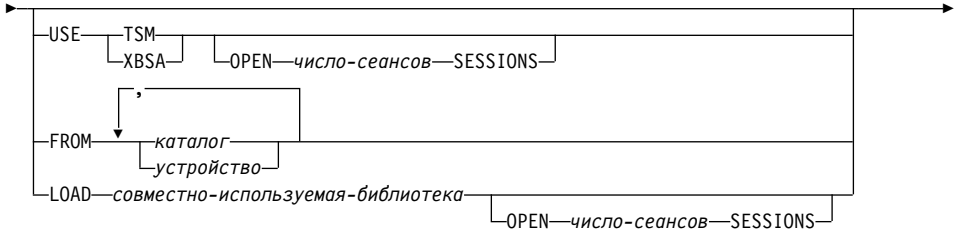


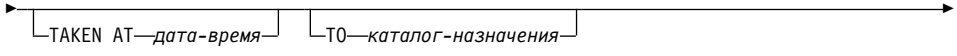
опции-восстановления:

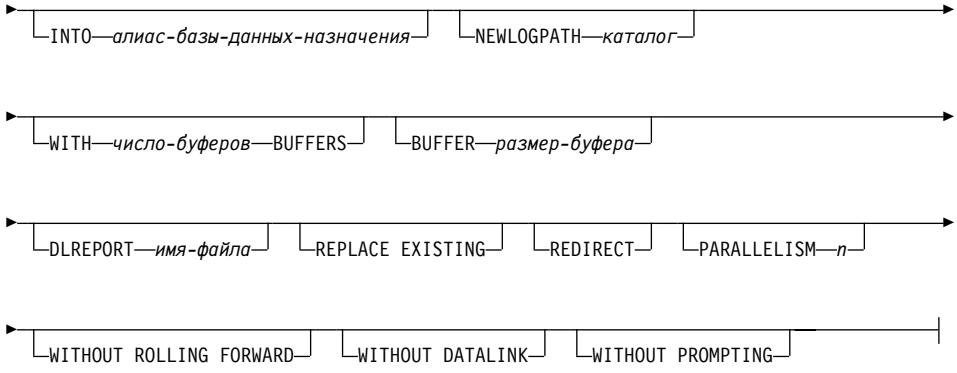












Параметры команды

DATABASE алиас-исходной-базы-данных
Алиас исходной базы данных, с которой была снята резервная копия.

CONTINUE
Указывает, что контейнеры были переопределены и что нужно выполнить последний шаг операции перенаправленного восстановления.

ABORT
Этот параметр:

- Останавливает операцию перенаправленного восстановления. Это удобно, если возникла ошибка, для которой требуется повторить один или несколько шагов. Выдав команду RESTORE DATABASE с опцией ABORT, нужно повторить все шаги операции перенаправленного восстановления, включая RESTORE DATABASE с опцией REDIRECT.
- Прерывает операцию инкрементного восстановления до ее завершения.

USER имя-пользователя
Указывает имя пользователя, под которым должно выполняться восстановление базы данных.

USING пароль
Пароль для аутентификации имени пользователя. Если пароль отсутствует, пользователю будет предложено ввести его.

TABLESPACE имя-табличного-пространства
Список имен табличных пространств, для которых требуется восстановление.

ONLINE
Это ключевое слово можно использовать только для операции восстановления уровня табличного пространства; оно разрешает

оперативное восстановление образа резервной копии. Это означает, что во время восстановления образа резервной копии другие агенты смогут соединяться с базой данных и что данные в других табличных пространствах будут доступны во время восстановления указанных табличных пространств.

HISTORY FILE
Это ключевое слово задается, чтобы восстановить из образа резервной копии только файл хронологии.

INCREMENTAL
Задает операцию инкрементного восстановления в ручном режиме. Все команды восстановления должны вводиться пользователем вручную.

AUTOMATIC/AUTO
Задает операцию инкрементного восстановления в автоматическом режиме.

USE TSM
Задает, что база данных должна быть восстановлена из выходных данных, управляемых TSM.

OPEN число-сеансов SESSIONS
Число сеансов ввода-вывода, используемых TSM или продуктом поставщика.

USE XBSA
Задает, что должен использоваться интерфейс XBSA. API служб резервного копирования (XBSA) - открытый интерфейс прикладного программирования для прикладных программ или утилит, которым для целей резервного копирования или архивирования требуется управление хранением данных. В настоящее время интерфейс XBSA поддерживается менеджером хранения Legato NetWorker.

FROM каталог/устройство
Каталог или устройство, где находятся образы резервных копий. Если опции USE TSM, FROM и LOAD опущены, по умолчанию используется текущий каталог.

В операционных системах Windows или OS/2 задаваемый каталог не должен быть каталогом, созданным DB2. Например, для следующих команд:

```
db2 backup database sample to c:\backup
db2 restore database sample from c:\backup
```

DB2 генерирует подкаталоги в каталоге `c:\backup`, который поэтому будет проигнорирован. Чтобы задать точное положение образа резервной копии для восстановления, используйте параметр TAKEN AT. В одном пути может храниться несколько образов резервных копий.

Если задано несколько устройств и последним задано ленточное устройство, пользователь получит предложение заменить ленту. Возможные ответы на запрос:

c Продолжить. Продолжить использование устройства, для которого было выдано предупреждение (например, когда смонтирована новая лента).

d Прекратить использовать устройство. Перестать использовать *только* то устройство, для которого было выдано предупреждение (например, если больше нет лент).

t Прекратить работу. Прервать операцию восстановления, если пользователем не смог по требованию утилиты выполнить какое-либо действие.

Ленточные устройства не поддерживаются в OS/2. В OS/2 можно задать `0` или `0:`, чтобы утилита восстановления вызывала программу обработчика пользователя. (Это возможно только в том случае, если для резервного копирования базы данных была использована программа обработчика пользователя.) При восстановлении с помощью программы обработчика пользователя для задания положения контейнеров используется только путь базы данных; поэтому восстанавливаются все контейнеры этой базы данных.

При использовании обработчика пользователя нельзя выполнять перенаправленное восстановление.

LOAD совместно-используемая-библиотека
Имя совместно используемой библиотеки (DLL в операционных системах Windows или OS/2), содержащей используемые функции поставщика для операций резервного копирования и восстановления. Это имя может содержать полный путь. Если полный путь не задан, по умолчанию используется путь программ обработчиков пользователя.

TAKEN AT дата-время
Отметка времени образа резервной копии базы данных. Это время выводится после успешного завершения резервного копирования и является частью имени пути созданного образа резервной копии. Оно задается в виде *гггг*мм*дд*чч*мм*сс. Это значение можно задавать не полностью. Например, если есть два образа резервных копий с отметками времени 19971001010101 и 19971002010101 и вы зададите для этого параметра значение `19971002`, будет использован образ резервной копии с отметкой времени `19971002010101`. Если значение этого параметра не задано, на исходном носителе должна быть только одна резервная копия.

TO каталог-назначения

Каталог базы данных назначения. Этот параметр игнорируется, если утилита выполняет восстановление в существующую базу данных.

Примечание: В операционных системах Windows или OS/2 при использовании этого параметра задавайте только букву диска. Если указать более длинный путь, будет возвращена ошибка.

INTO алиас-базы-данных-назначения

Алиас базы данных назначения. Если эта база данных не существует, она будет создана.

NEWLOGPATH каталог

Полное имя каталога, который будет использоваться для активных файлов журналов после операции восстановления. Этот параметр действует так же, как параметр конфигурации базы данных *newlogpath*, но влияет только на операцию восстановления, в которой он задан. Этот параметр можно использовать, когда путь журналов в образе резервной копии не должен использоваться после восстановления; например, когда этот путь более неверен или используется другой базой данных.

WITH число-буферов BUFFERS

Число используемых буферов. Значение по умолчанию - 2. Однако можно использовать больше буферов для улучшения производительности при чтении с нескольких источников или если было увеличено значение параметра PARALLELISM.

BUFFER размер-буфера

Размер буфера (в страницах) для операции восстановления. Минимальное значение этого параметра - 8 страниц; значение по умолчанию - 1024 страницы. Если задан нулевой размер буфера, для размера буфера будет использоваться значение параметра конфигурации менеджера баз данных *restbufsz*.

Размер буфера восстановления должен быть положительным целым числом, кратным размеру буфера, заданному для операции резервного копирования. Если задан неправильный размер буфера, будут использоваться буферы наименьшего приемлемого размера.

При использовании ленточных устройств в SCO UnixWare 7 задайте размер буфера 16.

DLREPORT имя-файла

Если имя файла задается, оно должно быть полным. В этот файл выводится информация о файлах, ставших несвязанными в результате быстрого согласования при операции восстановления. Эту опцию нужно

использовать, только если восстанавливаемая таблица содержит столбцы типа DATALINK и связанные файлы.

REPLACE EXISTING

Этот параметр задает, что если уже существует база данных с тем же алиасом, что и алиас базы данных назначения, то нужно заместить эту существующую базу данных восстановленной базой данных. Его удобно использовать в сценариях, запускающих утилиту восстановления, так как процессор командной строки не будет запрашивать у пользователя подтверждения удаления существующей базы данных. Если задан параметр WITHOUT PROMPTING, не обязательно задавать параметр REPLACE EXISTING, но в таком случае если возникнет ситуация, требующая вмешательства пользователя, будет выдана ошибка.

REDIRECT

Задает операцию перенаправленного восстановления. Для завершения операции перенаправленного восстановления после этой команды должны идти одна или несколько команд SET TABLESPACE CONTAINERS и затем команда RESTORE DATABASE с опцией CONTINUE.

Примечание: Все команды для одной операции перенаправленного восстановления должны быть выданы в одном окне или сеансе процессора командной строки.

WITHOUT ROLLING FORWARD

Задает, что после успешного завершения восстановления база данных не должна переводиться в состояние отложенного повтора транзакций.

Если после успешного завершения восстановления база данных находится в состоянии отложенного повтора транзакций, перед использованием этой базы данных необходимо выполнить “Команда ROLLFORWARD DATABASE” на стр. 158.

WITHOUT DATALINK

Задает, что все таблицы со столбцами DATALINK должны быть переведены в состояние отложенного согласования связей данных и что согласование связанных файлов не должно выполняться.

PARALLELISM n

Задает число использующих буферы процессов, запускаемых при операции восстановления. Значение по умолчанию - 1.

WITHOUT PROMPTING

Задает, что операция восстановления выполняется в автоматическом режиме. В ситуациях, в которых требуется вмешательство пользователя, будет выдано сообщение об ошибке. При использовании устройств со

сменным носителем (например, для лент или дискет) в конце чтения данных с носителя пользователь получит подсказку, даже если задана эта опция.

API восстановления базы данных

Синтаксис API на языке C

```
/* Файл: sqlutil.h */
/* API: Восстановить базу данных */
/* ... */
SQL_API_RC SQL_API_FN
  sqlurestore (
    char * pSourceDbAlias,
    char * pTargetDbAlias,
    sqluint32 BufferSize,
    sqluint32 RollforwardMode,
    sqluint32 DatalinkMode,
    sqluint32 RestoreType,
    sqluint32 RestoreMode,
    sqluint32 CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    char * pTargetPath,
    sqluint32 NumBuffers,
    char * pReportFile,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaSourceList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    void * pRestoreInfo,
    void * pContainerPageList,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

Общий синтаксис API

```
/* Файл: sqlutil.h */
/* API: Восстановить базу данных */
/* ... */
SQL_API_RC SQL_API_FN
  sqlgrestore (
    unsigned short SourceDbAliasLen,
    unsigned short TargetDbAliasLen,
    unsigned short TimestampLen,
    unsigned short TargetPathLen,
    unsigned short UserNameLen,
    unsigned short PasswordLen,
    unsigned short ReportFileLen,
    unsigned short Reserved2Len,
    char * pSourceDbAlias,
    char * pTargetDbAlias,
    sqluint32 BufferSize,
    sqluint32 RollforwardMode,
    sqluint32 DatalinkMode,
    sqluint32 RestoreType,
    sqluint32 RestoreMode,
    sqluint32 CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    char * pTargetPath,
    sqluint32 NumBuffers,
    char * pReportFile,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaSourceList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    unsigned short RestoreInfoSize,
    void * pRestoreInfo,
    unsigned short ContainerPageListSize,
    void * pContainerPageList,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

Параметры API

SourceDbAliasLen

Входной. Двухбайтное целое число без знака - длина в байтах алиаса исходной базы данных.

TargetDbAliasLen
Входной. Двухбайтное целое число без знака - длина в байтах алиаса базы данных назначения. Задайте значение ноль, если алиас базы данных назначения не указан.

TimestampLen
Входной. Двухбайтное целое число без знака - длина в байтах отметки времени. Задайте значение ноль, если отметка времени отсутствует.

TargetPathLen
Входной. Двухбайтное целое число без знака - длина в байтах имени каталога назначения. Задайте значение ноль, если путь назначения не указан.

UserNameLen
Входной. Двухбайтное целое число без знака - длина в байтах имени пользователя. Задайте значение ноль, если имя пользователя не указано.

PasswordLen
Входной. Двухбайтное целое число без знака - длина пароля в байтах. Задайте значение ноль, если пароль не указан.

ReportFileLen
Входной. Двухбайтное целое число без знака - длина в байтах имени файла отчета. Задайте значение ноль, если имя файла отчета не указано.

Reserved2Len
Входной. Двухбайтное целое число без знака - длина в байтах зарезервированной области. Задайте значение ноль.

pSourceDbAlias
Входной. Строка, содержащая алиас базы данных для резервной копии исходной базы данных.

pTargetDbAlias
Входной. Строка, содержащая алиас базы данных назначения. Если значение этого параметра - NULL, используется значение *pSourceDbAlias*.

BufferSize
Входной. Размер буфера резервного копирования в единицах выделения памяти по 4 Кбайта (страницах). Минимальное значение - 8 единиц. Значение по умолчанию - 1024.

Заданный размер буфера должен быть равен размеру буфера для резервной копии или кратен ему.

RollforwardMode
Входной. Задает, оставлять ли базу данных в состоянии ожидания повтора транзакций в конце операции восстановления. Допустимые значения (заданы в файле `sqlutil`):

SQLUD_ROLLFWD
Оставить базу данных в состоянии ожидания повтора транзакций после ее успешного восстановления.

SQLUD_NOROLLFWD
Не оставлять базу данных в состоянии ожидания повтора транзакций после ее успешного восстановления.

Если после успешной операцией восстановления база данных находится в состоянии ожидания повтора транзакций, прежде чем базу данных можно будет использовать снова, необходимо вызвать “API повтора транзакций базы данных” на стр. 164.

DatalinkMode
Входной. Задает, следует ли привести какие-либо таблицы со столбцами DATALINK в состояние ожидания согласования связей данных и надо ли выполнить согласование связанных файлов. Допустимые значения (определены в файле `sqlutil`):

SQLUD_DATALINK
Выполнить операции согласования. В таблицах, где определен столбец DATALINK, должна быть задана опция RECOVERY YES.

SQLUD_NODATALINK
Не выполнять операции согласования. Таблицы со столбцами DATALINK переводятся в состояние ожидания согласования связей данных. В таблицах, где определен столбец DATALINK, должна быть задана опция RECOVERY YES.

RestoreType
Входной. Задает тип операции восстановления. Допустимые значения (заданы в файле `sqlutil`):

SQLUD_FULL
Восстановить все из резервной копии. Выполняется в автономном режиме. В дальнейшем это значение не будет поддерживаться. Чтобы задать операцию полного восстановления базы данных, используйте значение `SQLUD_DB`.

SQLUD_DB
Восстановить все табличные пространства в базе данных. Выполняется в автономном режиме.

SQLUD_ONLINE_TABLESPACE
Восстанавливать только резервные копии уровня табличных пространств. Выполняется в оперативном режиме. В дальнейшем это значение не будет поддерживаться. Чтобы

задать операции восстановления уровня табличных пространств, используйте значение SQLUD_TABLESPACE | SQLUD_ONLINE.

SQLUD_TABLESPACE
Восстанавливать только резервные копии уровня табличных пространств. Может выполняться в оперативном или автономном режиме.

SQLUD_HISTORY
Восстанавливать только файл хронологии восстановления.

SQLUD_INCREMENTAL
Выполнить ручную инкрементную операцию восстановления.

SQLUD_AUTOMATIC
Выполнить автоматическую инкрементную операцию восстановления. Должно задаваться с SQLUD_INCREMENTAL, то есть SQLUD_INCREMENTAL | SQLUD_AUTOMATIC.

RestoreMode
Входной. Задайте, будет ли операция восстановления выполняться автономно или оперативно. Допустимые значения (заданы в файле sqlutil):

SQLUD_OFFLINE
Выполнить операцию восстановления автономно.

SQLUD_ONLINE
Выполнить операцию восстановления оперативно.

CallerAction
Входной. Задает тип действия, которое следует предпринять. Допустимые значения (заданы в файле sqlutil):

SQLUD_RESTORE
Начать операцию восстановления.

SQLUD_TERMINATE_INCREMENTAL
Прервать инкрементную операцию восстановления до ее завершения.

SQLUD_NOINTERRUPT
Начать операцию восстановления. Задает, что операция восстановления выполняется в автоматическом режиме. В случае сценариев, обычно требующих вмешательства пользователя, либо будет предпринята попытка реализовать их без предварительного возврата вызывающей программе, либо будет выдано сообщение об ошибке. Это действие вызывающей программы используется, например, когда известно, что все

носители, необходимые для операции восстановления, смонтированы, и приглашение утилиты не требуется.

SQLUD_CONTINUE
Продолжать использовать устройство, выдавшее предупреждающее сообщение (например, продолжить после монтирования новой ленты).

SQLUD_TERMINATE
Прервать операцию восстановления, если пользователем не смог по требованию утилиты выполнить какое-либо действие.

SQLUD_DEVICE_TERMINATE
Удалить устройство из списка устройств, используемых утилитой восстановления. Когда входные данные на устройстве исчерпаны, утилита восстановления возвращает предупреждение вызывающей программе. Вызовите утилиту восстановления снова при помощи этого действия вызывающей программы. Устройство, выдавшее предупреждение, удаляется из списка используемых устройств.

SQLUD_PARM_CHECK
Проверить параметры без выполнения операции восстановления.

SQLUD_RESTORE_STORDEF
Начальный вызов. Затребовать переопределение контейнера табличного пространства.

Для *CallerAction* надо при первом вызове задать значения `SQLUD_RESTORE`, `SQLUD_NOINTERRUPT`, `SQLUD_RESTORE_STORDEF` или `SQLUD_PARM_CHECK`.

pApplicationId
Выходной. Задайте буфер длиной `SQLU_APPLID_LEN+1` (определен в файле `sqlutil`). Утилита восстановления возвращает строку, идентифицирующую агент, который обслуживает прикладную программу. Может использоваться с API системного монитора базы данных для отслеживания работы программы.

pTimestamp
Входной. Строка, представляющая отметку времени резервной копии. Это поле не обязательно, если в заданном источнике есть только одна резервная копия.

pTargetPath
Входной. Строка, содержащая относительное или полное имя каталога базы данных назначения. Используется, если во время операции восстановления требуется создать новую базу данных.

NumBuffers
Входной. Число буферов, используемых для операции восстановления.

pReportFile
Если задано имя файла, оно должно быть полным. Сообщает о файлах, которые остались не связанными в результате быстрого согласования во время операции восстановления. Эту опцию следует использовать, только если у восстанавливаемой таблицы есть столбцы типа DATALINK и связанные файлы.

pTablespaceList
Задает одно или несколько табличных пространств для восстановления. Используется при восстановлении поднабора резервных копий или табличного пространства из резервной копии уровня табличных пространств.

Применяются следующие ограничения:

- База данных должна быть восстановимой. В восстановимых базах данных параметр конфигурации *logretain* имеет значение ″RECOVERY″ или включен параметр конфигурации *userexit*, или выполнены оба эти условия.
- Восстанавливается та же база данных, что использовалась для создания резервной копии. Поэтому табличные пространства нельзя добавлять в базу данных при помощи функции восстановления табличных пространств.
- При восстановлении из обработчика пользователя в OS/2 эта функция недоступна.
- Утилита повтора транзакций обеспечивает синхронизацию табличных пространств, восстановленных в среде MPP, с любым другим узлом, содержащим те же табличные пространства.

Примечание: При восстановлении табличного пространства, которое было переименовано после создания резервной копии, нужно использовать при вызове утилиты restore новое имя табличного пространства. Если используется старое имя, табличное пространство не будет найдено.

pMediaSourceList
Входной. Исходные носители для резервной копии. Смотрите раздел “Структура данных: SQLU-MEDIA-LIST” на стр. 108. Информация, которую вызывающей программе нужно указать в этой структуре, зависит от значения в поле *media_type*. Допустимые значения для этого поля (заданы в файле `sqlutil`):

SQLU_LOCAL_MEDIA
Локальные устройства (сочетание лент, дисков или дискет). Задает список структур *sqlu_media_entry*. В операционных

системах Windows или OS/2 в записях можно использовать только полные имена каталогов, но не имена ленточных устройств.

SQLU_TSM_MEDIA
TSM. Дополнительный ввод не требуется. Используется совместно используемая библиотека TSM, поставляемая вместе с DB2. Если нужна другая версия TSM, задайте имя совместно используемой библиотеки в SQLU_OTHER_MEDIA.

SQLU_OTHER_MEDIA
Продукт другого поставщика. Задайте имя совместно используемой библиотеки в структуре *sqlu_vendor*.

SQLU_USER_EXIT
Обработчик пользователя. Дополнительный ввод не требуется (доступен только в OS/2).

pUserName
Входной. Строка, содержащая имя пользователя для соединения.

pPassword
Входной. Строка, содержащая пароль для аутентификации имени пользователя.

pReserved2
Зарезервирован для будущего использования.

VendorOptionsSize
Входной. Длина поля опций поставщика. Эта длина не может превышать 65535 байтов.

pVendorOptions
Входной. Предназначен для использования поставщиком для передачи информации от прикладной программы функциям поставщика. Эта структура данных должна быть плоской, то есть уровни косвенной ссылки не поддерживаются. Имейте в виду, что обращение байтов не выполняется, а кодовая страница для этих данных не проверяется.

Параллелизм
Входной. Степень внутриразделного параллелизма (число манипуляторов буферов).

RestoreInfoSize
Зарезервирован для будущего использования.

pRestoreInfo
Зарезервирован для будущего использования.

ContainerPageListSize
Зарезервирован для будущего использования.

pContainerPageList
Зарезервирован для будущего использования.

pReserved3
Зарезервирован для будущего использования.

pSqlca Выходной. Указатель на структуру *sqlca*. Дополнительную информацию об этой структуре смотрите в публикациях *Administrative API Reference* или *SQL Reference*.

Синтаксис API REXX

```
RESTORE DATABASE алиас-исходной-базы-данных [USING :значение] [USER
имя-пользователя USING пароль]
[TABLESPACE :имена-табличных-пространств] [ONLINE | HISTORY FILE ]
[LOAD совместно-используемая-библиотека [OPTIONS опции-поставщика] [OPEN
число-сеансов SESSIONS] |
 FROM :исходная-область | USE TSM [OPEN число-сеансов SESSIONS] | USER_EXIT]
[TAKEN AT отметка-времени] [TO каталог-назначения] [INTO
алиас-базы-данных-назначения]
[ACTION действие-пользователя] [WITH число-буферов BUFFERS] [BUFFERSIZE
размер-буфера]
[WITHOUT ROLLING FORWARD] [PARALLELISM степень-параллелизма]
```

Параметры API REXX

алиас-исходной-базы-данных
Алиас исходной базы данных, с которой была снята резервная копия.

значение
Составная переменная хоста REXX, которой возвращается информация о восстановлении базы данных. Далее XXX будет обозначать имя переменной хоста:

XXX.0 Число элементов в переменной (всегда 1)

XXX.1 ID прикладной программы, идентифицирующий обслуживающего ее агента.

имя-пользователя
Указывает имя пользователя для соединения.

пароль Пароль для аутентификации имени пользователя.

имена-табличных-пространств
Составная переменная хоста REXX, содержащая список табличных пространств для восстановления. Далее XXX будет обозначать имя переменной хоста:

XXX.0 Число табличных пространств для восстановления

XXX.1 Имя первого табличного пространства

XXX.2 Имя второго табличного пространства

XXX.3 и так далее.

HISTORY FILE
Задает восстанавливать только файл хронологии из резервной копии.

совместно-используемая-библиотека
Имя совместно используемой библиотеки (DLL в операционных системах Windows или OS/2), содержащей используемые функции ввода/вывода восстановления поставщика. Оно может быть полным именем. Если полное имя не задано, по умолчанию используется полное имя обработчика пользователя.

опции-поставщика
Информация, необходимая для функций поставщика.

число-сеансов
Число сеансов ввода/вывода, используемых с TSM или продуктом поставщика.

исходная-область
Составная переменная хоста REXX, указывающая, в каком каталоге или на каком устройстве находится резервная копия. Значение по умолчанию - текущий каталог. В операционных системах Windows или OS/2 в записях можно использовать только полные имена каталогов, но не имена ленточных устройств.

отметка-времени
Отметка времени резервной копии базы данных.

каталог-назначения
Каталог базы данных назначения.

алиас-базы-данных-назначения
Алиас базы данных назначения. Если база данных назначения не существует, она будет создана.

действие-пользователя
Задает действие, которое следует предпринять. Допустимые значения:

SQLUD_RESTORE
Начать операцию восстановления.

SQLUD_NOINTERRUPT
Начать операцию восстановления. Задает, что операция восстановления выполняется в автоматическом режиме. Для сценариев, обычно требующих вмешательства пользователя, будет либо предпринята попытка реализовать их без предварительного возврата вызывающей программе, либо будет выдано сообщение об ошибке. Это действие вызывающей программы используется, например, когда известно, что все

носители, необходимые для операции восстановления, смонтированы, и приглашения утилиты нежелательны.

SQLUD_CONTINUE
Продолжать использовать устройство, выдавшее предупреждающее сообщение (например, продолжить после монтирования новой ленты).

SQLUD_TERMINATE
Прервать операцию восстановления, если пользователем не смог по требованию утилиты выполнить какое-либо действие.

SQLUD_DEVICE_TERMINATE
Удалить устройство из списка устройств, используемых утилитой восстановления. Когда входные данные на устройстве исчерпаны, утилита восстановления возвращает предупреждение вызывающей программе. Вызовите утилиту восстановления снова при помощи этого действия вызывающей программы. Устройство, выдавшее предупреждение, удалено из списка используемых устройств.

SQLUD_PARM_CHECK
Проверить параметры без выполнения операции восстановления.

SQLUD_RESTORE_STORDEF
Начальный вызов. Затребовать переопределение контейнера табличного пространства.

число-буферов
Число используемых буферов резервного копирования.

размер-буфера
Размер буфера резервного копирования в единицах выделения памяти по 4 Кбайта. Минимальное значение - 16 единиц.

степень-параллелизма
Степень внутриразделного параллелизма (число манипуляторов буфера).

Примеры сеансов восстановления

Примеры с применением командной строки

Ниже приводится типичный сценарий перенаправленного восстановления базы данных с алиасом MYDB:

1. Введите команду RESTORE DATABASE с опцией REDIRECT.

   ```
   db2 restore db mydb replace existing redirect
   ```

   После успешного завершения шага 1 и перед выполнением шага 3 операцию восстановления можно прервать, введя команду:

   ```
   db2 restore db mydb abort
   ```

2. Введите команду SET TABLESPACE CONTAINERS для каждого табличного пространства, чьи контейнеры требуется переопределить. Например, в OS/2:

   ```
   db2 set tablespace containers for 5 using
      (file 'f:\ts3con1' 20000, file 'f:\ts3con2' 20000)
   ```

   Чтобы проверит, что контейнеры восстановленной базы данных - именно те, которые заданы в этом шаге, введите команду LIST TABLESPACE CONTAINERS.

3. После успешного завершения шагов 1 и 2 введите команду:

   ```
   db2 restore db mydb continue
   ```

   Это завершающий шаг операции перенаправленного восстановления.

4. Если шаг 3 завершается неудачно, или если операция восстановления была прервана, перенаправленное восстановление можно перезапустить с шага 1.

Следующий пример иллюстрирует стратегию еженедельного инкрементного резервного копирования восстановимой базы данных. Он включает в себя операцию еженедельного полного резервного копирования базы данных, операцию ежедневного разностного резервного копирования и операцию резервного копирования дважды в неделю:

```
(Sun) backup db kdr use tsm
(Mon) backup db kdr online incremental delta use tsm
(Tue) backup db kdr online incremental delta use tsm
(Wed) backup db kdr online incremental use tsm
(Thu) backup db kdr online incremental delta use tsm
(Fri) backup db kdr online incremental delta use tsm
(Sat) backup db kdr online incremental use tsm
```

Чтобы восстановить автоматически базу данных с использованием образов, созданных в пятницу утром, введите команду:

```
restore db kdr incremental automatic taken at (Thu)
```

Чтобы восстановить базу данных вручную с использованием образов базы данных, созданных в пятницу утром, введите команды:

```
restore db kdr incremental taken at (Thu)
restore db kdr incremental taken at (Sun)
restore db kdr incremental taken at (Wed)
restore db kdr incremental taken at (Thu)
```

Пример командного сценария DB2 и информация о том, как его использовать, приведены в разделе “Приложение F. Сценарий восстановления CLP” на стр. 457.

Примеры с применением API

Примеры программ, содержащих API DB2 и встроенные вызовы SQL и информация о том, как их использовать, приведены в разделе “Приложение E. Программы примеров восстановления” на стр. 389.

Повышение производительности восстановления

Чтобы сократить время операции восстановления:

- Увеличьте размер буфера восстановления.

  Размер буфера восстановления должен быть положительным целым числом, кратным размеру буфера резервного копирования, указанному при резервном копировании. Если указать неправильный размер буфера, будут выделены буферы наименьшего допустимого размера.

- Увеличьте число буферов.

  Указанное значение должно быть кратным числу страниц буфера резервного копирования. Минимальное число страниц - 16.

Ограничения при восстановлении

Для утилиты восстановления есть следующие ограничения:

- Утилиту восстановления можно использовать только в том случае, если ранее с помощью утилиты резервного копирования DB2 была сделана резервная копия базы данных.
- Если вы используете Центр управления DB2, вы не сможете восстановить резервные копии, созданные предыдущими версиями DB2.
- Нельзя начать операцию восстановления базы данных, пока идет процесс повтора транзакций.
- Восстанавливать табличное пространство можно только в том случае, если это табличное пространство существует, и это именно то же самое табличное пространство, то есть оно не было отброшено, а потом создано снова между операциями резервного копирования и восстановления.
- Резервную копию уровня табличного пространства нельзя восстановить в новую базу данных.
- Нельзя проводить оперативное восстановление на уровне табличных пространств, затрагивающее таблицы системного каталога.

Ограничения при восстановлении

- В OS/2 невозможны операции частичного восстановления или восстановления поднабора при восстановлении через обработчик пользователя.
- Если кодовая страница восстанавливаемой базы данных не совпадает с кодовой страницей, доступной прикладной программе, или если менеджер баз данных не поддерживает преобразование кодовой страницы базы данных в кодовую страницу, доступную прикладной программе, эта восстановленная база данных будет непригодна к использованию.

Устранение неисправностей при восстановлении

Если при попытке восстановить резервную копию в новую базу данных в другой системе было получено сообщение SQL0970N, а в исходной базе данных табличные пространства определены с помощью абсолютных путей, воспользуйтесь операцией перенаправленного восстановления, чтобы определить контейнеры табличных пространств в новой системе. Утилита восстановления пытается использовать абсолютные пути и контейнеры, которых не существует в новой системе.

Если при восстановлении базы данных произошла ошибка, вы не сможете связаться с базой данных, пока не вызовете утилиту восстановления еще раз и не завершите восстановление успешно. Если ошибка системы произошла во время восстановления табличного пространства, непригодным для использования окажется только восстанавливавшееся табличное пространство. Остальные табличные пространства в базе данных можно будет продолжать использовать.

Глава 4. Восстановление с повтором транзакций

В этом разделе описывается утилита восстановления с повтором транзакций DB2 UDB, которая используется для восстановления базы данных путем выполнения транзакций, записанных в файлы журнала восстановления этой базы данных.

Рассматриваются следующие темы:

Обзор повтора транзакций

Самая простая форма команды DB2 ROLLFORWARD DATABASE требует только указания алиаса базы данных, для которой вы хотите выполнить восстановление с повтором транзакций. Например:

```
db2 rollforward db sample stop
```

В этом примере команда возвращает:

```
                                 Состояние повтора

 Алиас входной базы данных              = sample
 Число узлов с возвращенным состоянием  = 1

 Номер узла                             = 0
 Состояние повтора транзакций           = не отложенное
```

© Copyright IBM Corp. 2001

```
 Следующий файл журнала на чтение       =
 Обработано файлов журналов             = -
 Последняя принятая транзакция          = 2001-03-11-02.39.48.000000

DB20000I  Команда ROLLFORWARD выполнена успешно.
```

Объяснение этих полей смотрите в разделе “Команда ROLLFORWARD DATABASE” на стр. 158.

Возможны следующие варианты восстановления с повтором транзакций:

1. Вызов утилиты повтора транзакций без опции STOP
2. Вызов утилиты повтора транзакций с опцией QUERY STATUS

   Если вы задаете восстановление до конца журналов, в случае, когда достигнутая точка по времени раньше, чем вы рассчитывали, при использовании опции QUERY STATUS вы можете получить сообщение, что отсутствует один или несколько файлов журнала.

   Если вы задаете восстановление до определенного момента времени, опция QUERY STATUS поможет убедиться, что операция повтора транзакций выполнена до заданного времени.
3. Вызов утилиты повтора транзакций с опцией STOP. После остановки операции нельзя повторять транзакции для дополнительных изменений.

Чтобы для базы данных можно было выполнить повтор транзакций, она должна быть успешно восстановлена (при помощи утилиты восстановления); для табличных пространств это не так. Табличное пространство может быть временно переведено в состояние ожидания повтора транзакций, но для выхода из него не требуется восстановления из резервной копии (например, при сбое питания).

При вызове утилиты повтора транзакций:

- Если база данных находится в состоянии ожидания повтора транзакций, производится повтор транзакций для базы данных. Если табличные пространства также находятся в состоянии ожидания повтора транзакций, после завершения операции повтора транзакций для базы данных необходимо снова вызвать утилиту повтора транзакций, чтобы повторить транзакции для табличных пространств.
- Если база данных *не* находится в состоянии ожидания повтора транзакций, но табличные пространства в этой базе данных *находятся* в состоянии ожидания повтора транзакций:
  - Если вы указали список табличных пространств, будет выполнен повтор транзакций только для этих табличных пространств.
  - Если вы не указали список табличных пространств, будет выполнен повтор транзакций для всех табличных пространств, находящихся в состоянии ожидания повтора транзакций.

Операция повтора транзакций для базы данных проходит в автономном режиме. База данных недоступна для использования до успешного завершения операции повтора транзакций; эта операция не может быть выполнена, если при вызове утилиты не была указана опция STOP.

Операция повтора транзакций для табличного пространства может выполняться в автономном режиме. База данных недоступна для использования до успешного завершения операции повтора транзакций. Это происходит, если достигнут конец журналов или если при вызове утилиты была указана опция STOP.

Операцию повтора транзакций можно выполнять *без отключения* для всех табличных пространств, кроме пространства SYSCATSPACE. При выполнении операции повтора транзакций для табличного пространства без отключения недоступным для использования становится только это табличное пространство, но остальные табличные пространства в базе данных *будут доступны*

При первом создании базы данных для нее разрешается только циклическая запись в журнал. Это означает, что журналы используются повторно, а не сохраняются или архивируются. При циклической записи в журнал восстановление с повтором транзакций невозможно: возможно только восстановление после отказа или восстановление версии (смотрите раздел “Что такое журналы восстановления” на стр. 33). Архивированные журналы отражают изменения в базе данных, внесенные после снятия резервной копии. Для архивирования журналов (и восстановления с повтором транзакций) надо задать для параметра конфигурации базы данных *logretain* значение RECOVERY, или для параметра конфигурации базы данных *userexit* - значение YES, или оба этих значения. По умолчанию для обоих параметров задается NO, поскольку изначально не существует образа резервной копии, который можно использовать для восстановления базы данных. При изменении значения одного или обоих параметров база данных будет переведена в состояние отложенного резервного копирования и для дальнейшего ее использования надо будет снять резервную копию базы данных в автономном режиме.

Дополнительную информацию о параметрах конфигурации базы данных, связанных с записью в журналы, смотрите в разделе “Параметры конфигурации для записи в журнал базы данных” на стр. 40.

Привилегии, полномочия и авторизация, необходимые для использования повтора транзакций

Привилегии позволяют пользователям создавать ресурсы баз данных и обращаться к ним. Уровни полномочий дают способ группировки привилегий и высокоуровневой поддержи менеджера баз данных и операций утилит. Все это вместе позволяет управлять доступом к менеджеру баз данных и его объектам

Полномочия, необходимые для использования повтора транзакций

баз данных. Пользователям доступны только объекты, для которых у них есть соответствующие права - то есть требуемые привилегии или полномочия.

Для использования утилиты повтора транзакций у вас должны быть полномочия SYSADM, SYSCTRL или SYSMAINT.

Использование повтора транзакций

Прежде, чем использовать повтор транзакций

Вы не должны быть связаны с базой данных, для которой будет выполняться восстановление с повтором транзакций: утилита повтора транзакций автоматически устанавливает соединение с указанной базой данных и разрывает его после завершения повтора тразакций.

База данных может быть как локальной, так и удаленной.

Вызов повтора транзакций

Утилита повтора транзакций может быть вызвана:

- Из командной строки.

  Пример команды ROLLFORWARD DATABASE, введенной в командной строке:

  ```
  db2 rollforward db sample to end of logs and stop
  ```

- Из записной книжки Повтор транзакций для базы данных в Центре управления. Чтобы открыть записную книжку Повтор транзакции для базы данных:

  1. В Центре управления раскрывайте дерево объектов, пока не найдете папку Базы данных.
  2. Щелкните по папке Базы данных. Все существующие базы данных будут показаны на панели содержимого в правой части окна.
  3. Щелкните правой кнопкой мыши по нужной базе данных на панели содержания и выберите во всплывающем меню Повторить транзакции. Откроется записная книжка Повтор транзакций для базы данных.

  Общую информацию о Центре управления смотрите в книге *Administration Guide*. Подробная информация приводится в электронной справке Центра управления.

- При помощи API **sqluroll**. Информацию об этом API смотрите в разделе “API повтора транзакций базы данных” на стр. 164. Общую информацию о создании прикладных программ с API управления DB2 смотрите в книге *Application Building Guide*.

В среде многораздельной базы данных утилита повтора транзакций должна быть вызвана с узла каталога базы данных.

Повтор транзакций для табличного пространства

Если в базе данных разрешено восстановление с повтором транзакций, есть возможность резервного копирования, восстановления из резервной копии и повтора транзакций для табличных пространств вместо всей базы данных. Применять стратегию восстановления отдельных табличных пространств можно с целью экономии времени: восстановление части базы данных занимает меньше времени, чем восстановление всей базы данных. Например, если у вас выходит из строя диск, содержащий только одно табличное пространство, можно восстановить это табличное пространство из резервной копии и выполнить для него повтор транзакций, не восстанавливая всю базу данных и не мешая пользователям обращаться к остальной части базы данных; этот вариант невозможен, если поврежденное табличное пространство содержит таблицы системного каталога, поскольку вы не сможете связаться с базой данных. (Табличное пространство системного каталога можно восстановить независимо, если доступна резервная копия уровня табличного пространства, содержащая табличное пространство системного каталога.) Кроме того, резервное копирование на уровне табличных пространств позволяет делать резервные копии критичных частей базы данных чаще, чем остальных, и отнимает меньше времени, чем резервное копирование всей базы данных.

После того, как табличное пространство восстановлено, оно всегда находится в состоянии ожидания повтора транзакций. Чтобы сделать табличное пространство пригодным к использованию, для него надо выполнить повтор транзакций. В большинстве случаев повтор транзакций можно выполнять до конца файлов журнала или до определенного момента времени. Однако для табличных пространств, содержащих таблицы системного каталога, нельзя выполнить повтор транзакций до определенного момента времени. Для этих табличных пространств повтор транзакций должен выполняться до конца файлов журнала, поскольку это гарантирует согласованность состояние всех табличных пространства в базе данных.

Перед повтором транзакций для табличного пространства введите команду LIST TABLESPACES SHOW DETAIL. Эта команда возвращает *самое раннее возможное время восстановления*, то есть самый ранний момент, до которого может быть выполнен повтор транзакций табличного пространства. Самое раннее время восстановления определяется последним выполнением операторов языка определения данных (data definition language - DDL) для табличного пространства или таблиц в табличном пространстве. Повтор транзакций табличного пространства должен быть выполнен, как минимум, до самого раннего времени восстановления, чтобы оно было синхронизировано с информацией в таблицах системного каталога. При восстановлении нескольких табличных пространств повтор транзакций для этих табличных пространств должен производиться до момента не раньше самого раннего времени восстановления для каждого из восстанавливаемых табличных пространств. В среде многораздельной базы данных введите команду LIST TABLESPACES

SHOW DETAIL для каждого раздела. Повтор транзакций для табличных пространств должен производиться, как минимум, до самого раннего минимального времени восстановления для всех табличных пространств на всех разделах.

Если вы выполняете повтор транзакций до определенного момента времени и таблица располагается в нескольких табличных пространствах, повтор транзакций для всех этих табличных пространств должен выполняться одновременно. Например, если данные таблицы содержатся в одном табличном пространстве, а индекс для этой таблицы - в другом, для обоих табличных пространств необходимо одновременно произвести повтор транзакций до одного и того же момента времени.

Если в таблице данные и объекты типа long находятся в отдельных табличных пространствах и эта таблица была реорганизована, оба вида табличных пространств должны быть совместно восстановлены из резервных копий, и для них, также совместно, должен быть произведен повтор транзакций. После реорганизации необходимо сделать резервную копию затронутых ей табличных пространств.

Если вы собираетесь повторить транзакции для табличного пространства до определенного момента времени, а таблица в этом табличном пространстве является:

- Базовой таблицей для таблицы сводки, находящейся в другом табличном пространстве, или
- Таблицей сводки для таблицы в другом табличном пространстве

Необходимо повторить транзакции для обоих табличных пространств до одного и того же момента времени. Если этого не сделать, по окончании операции повтора транзакций таблица сводки будет переведена в состояние ожидания проверки.

Если вы хотите повторить транзакции для табличного пространства до определенного момента времени, а таблица в табличном пространстве участвует во взаимоотношении реляционной целостности с другой таблицей, содержащейся в другом табличном пространстве, необходимо одновременно повторить транзакции для обоих табличных пространств до одного и того же момента времени. Если этого не сделать, по окончании операции повтора транзакций до определенного момента времени оба табличных пространства будут в состоянии ожидания проверки. Если одновременно повторить транзакции для обоих табличных пространств, по окончании операции повтора транзакций до определенного момента времени это ограничение останется активным.

Убедитесь в том, чтобы операция повтора транзакций для табличного пространства до определенного момента времени не привела к откату транзакции в одних табличных пространствах и к ее принятию в других. Это может произойти, если:

- Повтор транзакций до определенного момента времени выполняется для поднабора табличных пространств, в которых транзакцией были сделаны изменения, и этот момент времени предшествует времени принятия транзакции.
- У какой-либо из таблиц, содержащихся в табличном пространстве, для которого выполнен повтор транзакций до определенного момента времени, есть связанный триггер или она обновлялась при помощи триггера, действующего на другие табличные пространства, кроме пространства, для которого производился повтор транзакций.

Для предотвращения такой ситуации следует найти подходящий момент времени окончания повтора транзакций.

Можно ввести команду QUIESCE TABLESPACES FOR TABLE для создания совместимого с транзакциями момента времени для повтора транзакций для табличных пространств. Требование стабилизации (при совместном использовании, намерении изменения или в монопольном режиме) ожидает (посредством блокировки) завершения всех текущих транзакций в отношении этих табличных пространств и блокирует новые требования. Когда требование стабилизации выполнено, табличные пространства находятся в согласованном состоянии. Чтобы определить приемлемое время завершения операции повтора транзакций, можно посмотреть файл хронологии восстановления, чтобы найти точки стабилизации и проверить, какие из них находятся после самого раннего времени восстановления.

После завершения операции повтора транзакций до определенного момента времени табличное пространство переводится в состояние ожидания резервного копирования. Необходимо произвести резервное копирование табличного пространства, поскольку все изменения, сделанные в нем между заданным моментом и текущим временем, были отменены. Вы больше не можете повторять транзакции для табличного пространства до текущего времени из предыдущего образа резервной копии уровня базы данных или табличного пространства. В следующем примере показано, зачем требуется резервная копия уровня табличного пространства и как она используется. (Чтобы сделать табличное пространство доступным, можно либо сделать резервную копию всей базы данных, в которой табличное пространство находится в состоянии ожидания резервного копирования, либо набора табличных пространств, включающего в себя такое табличное пространство.)

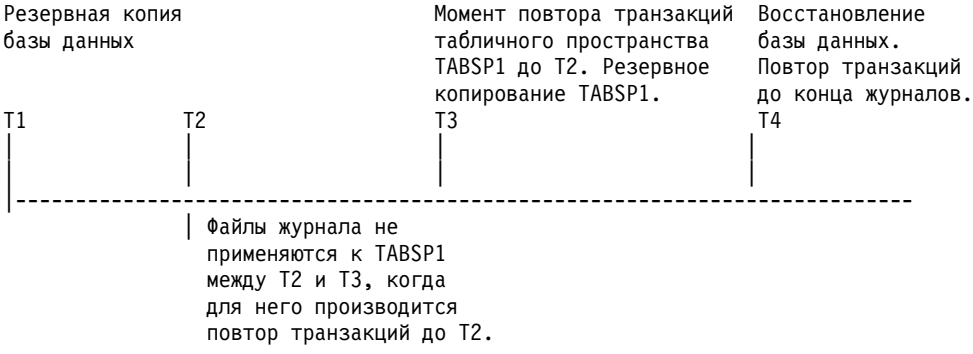


Рисунок 17. Требования к резервной копии табличного пространства

В предыдущем примере резервная копия базы данных снимается в момент T1. Затем, в момент T3, для табличного пространства TABSP1 производится повтор транзакций до определенного момента времени (T2); после времени T3 снимается резервная копия табличного пространства. Поскольку это табличное пространство находится в состоянии ожидания резервного копирования, операция резервного копирования является обязательной. Отметка времени образа резервной копии будет позже момента T3, но само табличное пространство находится в состоянии на момент T2. Записи журнала между T2 и T3 не применяются к TABSP1. В момент T4 база данных восстанавливается из резервной копии с использованием образа, созданного в момент T1, и для нее повторяются транзакции до конца файлов журнала. Табличное пространство TABSP1 переводится в состояние ожидания восстановления при достижении T3, потому что менеджер баз данных полагает, что между T3 и T4 над TABSP1 были выполнены операции без применения к этому табличному пространству изменений в журнале между T2 и T3. Если в действительности эти изменения в журнале были применены как часть операции повтора транзакций для базы данных, это предположение будет неверным. Резервная копия табличного пространства, которая должна быть сделана после повтора транзакций до определенного момента времени, допускает повтор транзакций для этого табличного пространства после предыдущего времени повтора транзакций (в нашем примере - T3).

Предположим, что вы хотите восстановить табличное пространство TABSP1 до момента T4; тогда необходимо восстановить это табличное пространство из образа резервной копии, сделанного после T3 (либо из обязательной резервной копии, либо из более поздней), а затем повторить транзакции для TABSP1 до конца файлов журнала.

В предыдущем примере самым эффективным способом восстановления базы данных до момента T4 будет выполнение необходимых шагов в следующем порядке:

1. Восстановить базу данных из резервной копии.
2. Восстановить табличное пространство из резервной копии.
3. Повторить транзакции для базы данных.
4. Повторить транзакции для табличного пространства.

Поскольку вы восстанавливаете табличное пространство до повтора транзакций для базы данных, ресурс не используется для применения записей журнала к табличному пространству при повторе транзакций для базы данных.

Если не удается найти образ резервной копии TABSP1 после момента T3 или если вы хотите восстановить TABSP1 до T3 или более раннего времени, можно:

- Повторить транзакции для табличного пространства до T3. Повторное восстановление табличного пространства не требуется, поскольку оно уже было восстановлено из образа резервной копии базы данных.
- Восстановить табличное пространство еще раз из резервной копии базы данных, сделанной в T1, а затем произвести повтор транзакций для табличного пространства до времени, предшествующего T3.
- Отбросить табличное пространство.

В среде многораздельной базы данных:

- Необходимо одновременно повторить транзакции для всех порций табличного пространства до одного и того же момента времени. Это обеспечивает совместимость табличного пространства на всех разделах базы данных.
- Если некоторые из разделов базы данных находятся в состоянии ожидания повтора транзакций, а в других разделах базы данных в этом состоянии находятся отдельные табличные пространства (но не сами разделы), сначала необходимо выполнить повтор транзакций для разделов базы данных, а потом - для табличных пространств.
- Если вы собираетесь повторить транзакции табличного пространства до конца журналов, вам не нужно восстанавливать его из резервной копии на каждом разделе базы данных; достаточно такого восстановления только на тех разделах базы данных, для которых требуется восстановление с повтором транзакций. Однако если вы собираетесь повторить транзакции для табличного пространства до определенного момента времени, необходимо восстановить это табличное пространство из резервной копии на каждом разделе базы данных.

Восстановление отброшенной таблицы

Вы можете случайно отбросить таблицу, данные которой вам еще нужны. В таком случае следует позаботиться о том, чтобы критичные для вас таблицы были восстановимыми после операции отбрасывания таблиц.

Восстановление отброшенной таблицы

Данные таблицы можно восстановить, вызвав операцию восстановления базы данных, а затем повторив транзакции до момента времени, предшествующего отбрасыванию таблицы. Для большой базы данных это может занять значительное время, и в процессе восстановления ваши данные будут недоступны.

Функция восстановления отброшенной таблицы DB2 позволяет восстанавливать данные отброшенной таблицы, используя операции восстановления из резервной копии и повтора транзакций уровня табличного пространства. Это будет быстрее восстановления уровня базы данных, а база данных останется доступной для пользователей.

Чтобы отброшенную таблицу можно было восстановить, у табличного пространства, в котором расположена эта таблица, должна быть включена опция DROPPED TABLE RECOVERY. Это может быть сделано во время создания табличного пространства или же позже, при помощи оператора ALTER TABLESPACE (смотрите справочник *SQL Reference*). Возможность задания опции DROPPED TABLE RECOVERY зависит от типа табличного пространства; она допустима только для обычного табличного пространства. Чтобы определить, есть ли у табличного пространства такая возможность, можно сделать запрос содержания столбца DROP_RECOVERY в таблице каталога SYSCAT.TABLESPACES.

Когда оператор DROP TABLE применяется для таблицы, расположенной в табличном пространстве с включенной опцией восстановления отброшенных таблиц, в файлы журнала производится дополнительная запись (идентифицирующая отброшенную таблицу). В файл хронологии восстановления помещается также запись с информацией, которую можно использовать для повторного создания таблицы.

За один раз может быть восстановлена только одна отброшенная таблица. Отброшенную таблицу можно восстановить так:

1. Найдите отброшенную таблицу, введя команду LIST HISTORY DROPPED TABLE (смотрите раздел “LIST HISTORY” на стр. 343). ID отброшенных таблиц приводятся в столбце Backup ID.
2. Восстановите образ резервной копии уровня базы данных или табличного пространства, сделанный до отбрасывания таблицы.
3. Создайте каталог для экспорта, в который будут записываться файлы, содержащие данные таблицы. Этот каталог должен быть доступен для всех разделов базы данных или существовать на каждом из разделов. Подкаталоги этого каталога для экспорта создаются автоматически каждым из разделов базы данных. Имена этих подкаталогов имеют вид `NODE`*`nnnn`*, где *nnnn* - номер раздела базы данных или узла. Файлы данных, содержащие данные отброшенной таблицы в том виде, в котором они существовали на

каждом из разделов базы данных, экспортируются в подкаталог нижнего уровня с именем data. Например, \export_directory\NODE0000\data.

4. Повторите транзакции до определенного момента времени после отбрасывания, используя команду ROLLFORWARD DATABASE с опцией RECOVER DROPPED TABLE. Другая возможность - повторить транзакции до конца файлов журнала, чтобы не потерять изменения в других таблицах табличного пространства или базы данных.
5. Повторно создайте таблицу, используя оператор CREATE TABLE из файла хронологии восстановления.
6. Импортируйте в таблицу данные, которые были экспортированы во время операции повтора транзакций.

Есть определенные ограничения на тип данных отброшенной и восстанавливаемой таблицы. Невозможно восстановить:

- Данные больших объектов или длинных полей Опция DROPPED TABLE RECOVERY не поддерживается для табличных пространств длинных полей. При попытке восстановить отброшенную таблицу, содержащую столбцы типа большой объект или LONG VARCHAR, в сгенерированном файле экспорта для них будет установлено пустое значение. Опция DROPPED TABLE RECOVERY может использоваться только для обычных табличных пространств, но не для временных табличных пространств или табличных пространств длинных полей.
- Метаданные, связанные с типами строк. (Восстанавливаются данные, но не метаданные.) Будут восстановлены данные в таблице иерархии типизированной таблицы. Данные могут содержать больше информации, чем было в отброшенной типизированной таблице.

Могут быть восстановлены имена файлов, связанных со столбцами DATALINK. После импорта данных таблица должна быть синхронизирована при помощи менеджера связей данных DB2. Возможность восстановления образов резервных копий файлов менеджером связей данных DB2 зависит от того, успели ли их удалить при чистке мусора.

Использование файла положения копии загрузки

Переменная реестра DB2LOADREC используется для идентификации файла с информацией о положении копии загрузки. Этот файл используется во время восстановления с повтором транзакций для определения положения копии загрузки. Он содержит следующую информацию:

- Тип носителя
- Число используемых устройств с носителем
- Положение копии загрузки, сгенерированной во время операции загрузки таблицы
- Имя файла копии загрузки (если применимо)

Использование файла положения копии загрузки

Если файла положения не существует или в файле не было найдено соответствующей записи, используется информация из файла журнала.

Информация в этом файле может быть переписана до того, как произойдет восстановление с повтором транзакций.

Примечания:

1. В среде многораздельной базы данных в файле `db2profile` должна быть переменная реестра DB2LOADREC.
2. В среде многораздельной базы данных файл копии загрузки должен существовать на всех серверах разделов базы данных, и имя этого файла (включая путь) везде должно быть одним и тем же.
3. Если запись в этом файле, заданном переменной реестра DB2LOADREC, недействительна, для замены такой записи будет использована информация из старого файла положения копии загрузки.

В файле положения находится следующая информация. Первые пять параметров должны быть действительными значениями, которые используются для идентификации копии загрузки. Вся эта структура повторяется для каждой из записанных копий загрузки. Например:

```
TIMestamp      19950725182542         * Отметка времени, сгенерированная во время
                                        загрузки
SCHema         PAYROLL                * Схема загруженной таблицы
TABlename      EMPLOYEES              * Имя таблицы
DATabasename   DBT                    * Имя базы данных
DB2instance    TORONTO                * DB2INSTANCE
BUFfernumber   NULL                   * Число буферов для использования при
                                        восстановлении
SESsionnumber  NULL                   * Число сеансов для использования при
                                        восстановлении
TYPeofmedia    L                      * Тип среды - L для локального устройства
                                                     A для TSM
                                                     O для устройств других
                                                       производителей
LOCationnumber 3                      * Число положений
   ENTry       /u/toronto/dbt.payroll.employes.001
   ENT         /u/toronto/dbt.payroll.employes.002
   ENT         /dev/rmt0
TIM            19950725192054
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            TORONTO
SES            NULL
BUF            NULL
TYP            A
TIM            19940325192054
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            TORONTO
```

```
SES             NULL
BUF             NULL
TYP             O
SHRlib          /@sys/lib/backup_vendor.a
```

Примечания:

1. Учитываются первые три символа в каждом из ключевых слов. Все ключевые слова должны идти в указанном порядке. Пустые строки недопустимы.
2. Отметка времени должна иметь вид *ггггммддччммсс*.
3. Все поля обязательны, за исключением BUF и SES, для которых допустимы пустые значения. При пустом значении для SES используется значение, указанное параметром конфигурации *numloadrecses*. При пустом значении для BUF по умолчанию используется SES+2.
4. Если в файле положения хотя бы одна запись неверна, в качестве источника этих значений используется предыдущий файл положения копии загрузки.
5. Тип носителя может быть локальным устройством (`L` для ленты, диска или дискет), TSM (`A`) или устройством другого производителя (`O`). При типе `L` требуется число положений, за которыми идут записи положений. Для типа `A` дополнительного ввода не требуется. Для типа `O` требуется имя совместно используемой библиотеки. Подробности об использовании в качестве носителя для резервного копирования TSM и продуктов других производителей смотрите в разделе “Приложение G. Tivoli Storage Manager” на стр. 465.
6. Параметр SHRlib указывает на библиотеку, у которой есть функция хранения данных копии загрузки.
7. Если вызвать операцию загрузки, указав опцию COPY NO или NONRECOVERABLE, и не сделать после завершения этой операции резервную копию базы данных или измененных табличных пространств, нельзя будет восстановить базу данных или табличные пространства до определенного момента времени после операции загрузки. Это значит, что будет невозможно использовать повтор транзакций для восстановления базы данных или табличных пространств до состояния, в котором они были после операции загрузки. Базу данных или табличные пространства можно будет восстановить только до момента времени, предшествующего операции загрузки.

Если вы хотите использовать конкретную копию загрузки, для определения отметки времени для этой конкретной операции загрузки можно использовать файл хронологии восстановления для базы данных. В среде многораздельной базы данных файл хронологии восстановления является локальным для каждого из разделов базы данных.

Дополнительную информацию об утилите загрузки смотрите в руководстве *Data Movement Utilities Guide and Reference*.

Синхронизация системного времени в системе многораздельной базы 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 значения, заданного для параметра конфигурации менеджера баз данных *max_time_diff* (максимальное различие времени по узлам).

Чтобы временные отметки записей журналов в системе многораздельной базы 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, но 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 раздела 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 раздела 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 раздела базы данных отметка времени в журнале базы данных устанавливается на 1999 год, и время журнала остается установленным на 7 ноября 1999 года, пока системное время не совпадет с этим временем. 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* менеджер баз данных сокращает количество случаев ошибок этого типа:

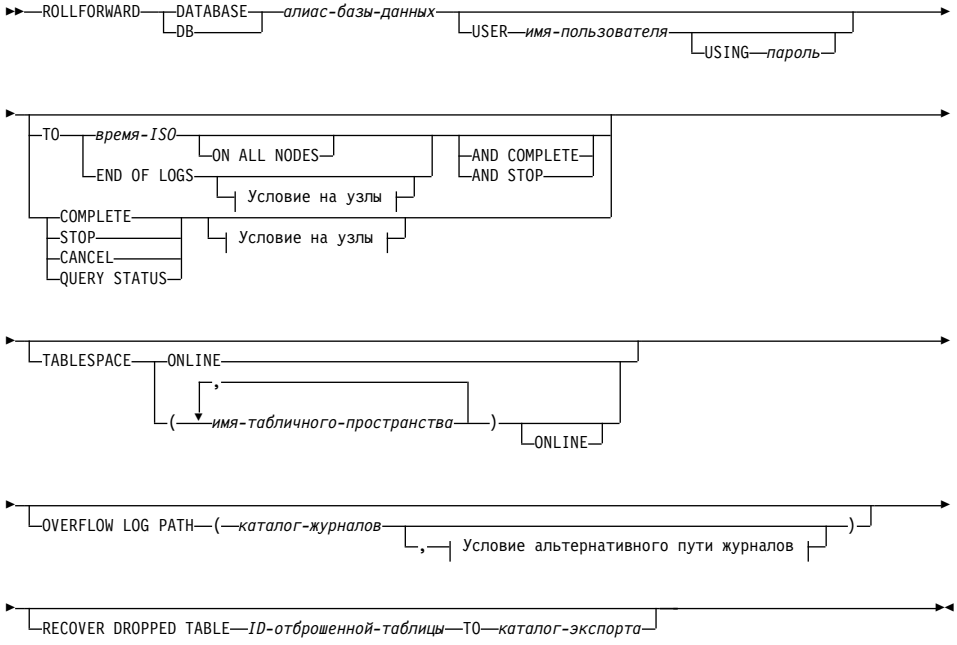
- Конфигурируемые значения этого параметра лежат в диапазоне от 1 минуты до 24 часов. Дополнительную информацию об установке *max_time_diff* смотрите в книге *Administration Guide: Performance*.
- При первом требовании соединения с узлом, не являющимся узлом каталога, сервер раздела базы данных посылает его время на узел каталога базы данных. После этого узел каталога проверяет, чтобы разница между временем на узле, требующем соединения, и его собственным временем находилась в

диапазоне, заданном параметром *max_time_diff*. Если этот диапазон превышен, требование на соединение отклоняется.

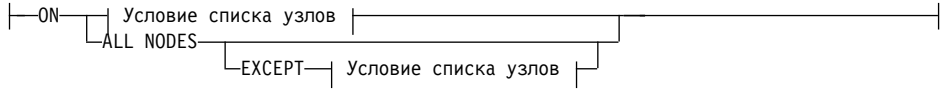
- Транзакция изменения, обращающаяся к трем и более серверам разделов базы данных, перед тем, как изменение можно будет принять, должна проверить, что часы участвующих серверов разделов базы данных синхронизированы. Если у двух или более серверов разделов базы данных отличие во времени превышает предел,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

Команда ROLLFORWARD DATABASE

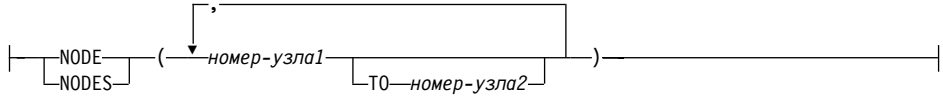
Синтаксис команды



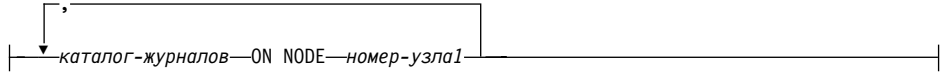
Условие на узлы:



Условие списка узлов:



Условие альтернативного пути журналов:



Параметры команды

DATABASE алиас-базы-данных
Алиас базы данных, для которой выполняется восстановление с повтором транзакций.

USER имя-пользователя
Имя пользователя, под которым выполняется восстановление с повтором транзакций для базы данных.

USING пароль
Пароль для аутентификации имени пользователя. Если пароль отсутствует, пользователю будет предложено ввести его.

TO

время-ISO
Задает момент времени - будет выполнен повтор для всех транзакций, принятых до этого момента времени (включая транзакции, принятые ровно в этот момент, а также все транзакции, принятые ранее).

Это значение задается в виде отметки времени - состоящей из семи частей символьной строки, включающей дату и время. Ее формат - *гггг-мм-дд-чч.мм.сс.нннннн* (год, месяц, день, час, минуты, секунды, микросекунды) согласованного универсального времени (UTC). Использование согласованного универсального времени помогает предотвратить появление одинаковых отметок времени для разных файлов (например, из-за изменения времени в связи с переходом на летнее время). Значение отметки времени в образе резервной копии - это локальное время для момента начала операции резервного копирования. Специальный регистр CURRENT TIMEZONE задает разницу между UTC и локальным временем на сервере прикладных программ. Эта разница представляет собой длительность времени (выраженную десятичным числом, в котором первые две цифры представляют количество часов, следующие две - минут, а последние две - секунд). Для преобразования локального времени в UTC из локального времени вычитается значение CURRENT TIMEZONE.

END OF LOGS
Задает, что должны быть применены все принятые транзакции из всех активных архивов файлов журналов, перечисленных в параметре конфигурации базы данных *logpath*.

ALL NODES
Задает, что повтор транзакций должен быть выполнен на всех узлах, заданных в файле `db2nodes.cfg`. Это значение используется по умолчанию, если не задано условие на узлы.

EXCEPT

Задает, что повтор транзакций должен быть выполнен на всех узлах, заданных в файле `db2nodes.cfg`, за исключением узлов, заданных в этом списке узлов.

ON NODE / ON NODES

Выполнить повтор транзакций для базы данных на наборе узлов.

номер-узла1

Задает номер узла в списке узлов.

номер-узла2

Задает второй номер узла, так что все узлы начиная с *номер-узла1* до *номер-узла2* (включительно) входят в список узлов.

COMPLETE / STOP

Остановить повтор транзакций для записей журнала и завершить процесс восстановления с повтором транзакций, выполнив откат всех незавершенных транзакций и отключив состояние ожидания повтора транзакций для базы данных. Это разрешает доступ к базе данных или табличным пространствам, для которых выполняется операция повтора транзакций. Эти ключевые слова эквивалентны, можно задать любое из них (но не оба вместе). Ключевое слово AND позволяет задать в одной команде несколько операций, например, `db2 rollforward db sample to end of logs and complete`.

Примечание: Когда для табличных пространств выполняется повтор транзакций до момента времени, эти табличные пространства переводятся в состояние отложенного резервного копирования.

CANCEL

Отменяет операцию восстановления с повтором транзакций. При этом база данных или одно или несколько табличных пространств на всех узлах, на которых было запущено восстановление с повтором, переводятся в состояние отложенного восстановления:

- Если операция восстановления с повтором транзакций для *базы данных* не была запущена (то есть база данных находится в состоянии отложенного повтора транзакций), эта опция переводит такую базу данных в состояние отложенного восстановления.
- Если операция восстановления с повтором транзакций для *табличных пространств* не была запущена (то есть эти табличные пространства находятся в состоянии отложенного повтора транзакций), необходимо задать список табличных пространств. Все табличные пространства из этого списка переводятся в состояние отложенного восстановления.
- Если операция повтора транзакций для табличных пространств *запущена* (то есть по крайней мере одно из табличных пространств

находится в состоянии выполнения повтора транзакций), все табличные пространства, находящиеся в состоянии выполнения повтора транзакций, переводятся в состояние отложенного восстановления. Если задается список табличных пространств, в него должны входить все табличные пространства, находящиеся в состоянии выполнения повтора транзакций. Все табличные пространства из этого списка переводятся в состояние отложенного восстановления.

- Если выполняется повтор транзакций до момента времени, все заданные в параметрах имена табличных пространств игнорируются и все табличные пространства, находящиеся в состоянии выполнения повтора транзакций, переводятся в состояние отложенного восстановления.
- Если выполняется повтор транзакций до конца журналов для списка табличных пространств, только перечисленные в этом списке табличные пространства переводятся в состояние отложенного восстановления.

Эту опцию нельзя использовать для отмены повтора транзакций *непосредственно во время выполнения этой операции*. Эту опцию можно использовать только для отмены операции повтора транзакций, которая запущена, но в настоящее время не выполняется. Операция повтора транзакций может быть запущена, но не выполняться, если:

- Она завершилась аварийно.
- Не была задана опция STOP.
- Возникла ошибка. Некоторые ошибки, например, повтор транзакций во время операции загрузки без возможности восстановления, могут перевести табличное пространство состояние отложенного восстановления.

Примечание: Используйте эту опцию с осторожностью и только в тех случаях, когда запущенная операция повтора транзакций не может быть завершена из-за того, что некоторые табличные пространства были переведены в состояние отложенного повтора транзакций или состояние отложенного восстановления. При сомнениях используйте команду LIST TABLESPACES, чтобы узнать, какие табличные пространства находятся в состоянии отложенного повтора транзакций или состоянии отложенного восстановления.

QUERY STATUS

Выводит имена файлов журналов, для которых менеджер баз данных выполнил повтор транзакций, имя следующего необходимого файла архива и отметку времени (в CUT) последней принятой транзакции

после начала выполнения повтора транзакций. В среде многораздельной базы данных эта информация о состоянии выводится для каждого узла. Возвращаемая информация содержит следующие поля:

Номер узла

Состояние повтора
Это состояние может быть: состоянием отложенного повтора транзакций для базы данных или таблицы, состоянием выполнения повтора транзакций для базы данных или таблицы, состоянием остановленного повтора транзакций для базы данных или таблицы, а также состоянием без отложенных операций.

Следующий файл журнала на чтение
Строка, содержащая имя следующего необходимого файла журнала. В среде многораздельной базы данных используйте эту информацию в тех случаях, когда утилита повтора транзакций возвращает код ошибки, указывающий на отсутствие файла журнала или на неправильную информацию в файле журнала.

Обработанные файлы журналов
Строка, содержащая имена обработанных файлов журналов, которые более не нужны для восстановления и поэтому могут быть удалены из каталога. Например, если самая ранняя непринятая транзакция начинается в файле журнала *x*, этот файл журнала не будет входить в список уже не нужных файлов журналов; этот список будет заканчиваться предыдущим файлом журнала.

Последняя принятая транзакция
Строка, содержащая значение отметки времени в формате ISO (*гггг-мм-дд-чч.мм.сс*). Эта отметка времени отмечает время последней принятой транзакции после завершения восстановления с повтором. Эта отметка времени относится к базе данных. При восстановлении с повтором транзакций для табличного пространства это отметка времени последней принятой транзакции в базе данных.

Примечание: Опция QUERY STATUS используется по умолчанию, если не заданы условия TO, STOP, COMPLETE и CANCEL. Если задано условие TO, STOP или COMPLETE, информация о состоянии выводится после успешного завершения команды. Если заданы конкретные табличные пространства, эти параметры игнорируются; запрос информации о состоянии нельзя применять только к некоторым табличным пространствам.

TABLESPACE

Это ключевое слово задает восстановление с повтором транзакций уровня табличных пространств.

имя-табличного-пространства

Обязательный параметр для восстановления с повтором транзакций до момента времени уровня табличных пространств. Для восстановления с повтором транзакций до конца журналов можно задать подмножество табличных пространств. В среде многораздельной базы данных табличные пространства из этого списка не обязательно должны существовать на всех узлах, на которых выполняется повтор транзакций. Если же табличное пространство *существует*, оно должно быть в правильном состоянии.

ONLINE

Это ключевое слово разрешает выполнение восстановления с повтором транзакций уровня табличных пространств в оперативном режиме. Это означает, что другим агентам разрешается соединяться с табличным пространством во время восстановления с повтором транзакций.

OVERFLOW LOG PATH каталог-журналов

Задает альтернативный путь журналов, в котором нужно искать архивированные журналы во время операции восстановления. Используйте этот параметр, если файлы журналов были перемещены в каталог, не совпадающий с каталогом, задаваемым параметром конфигурации базы данных *logpath*. В среде многораздельной базы данных это полное имя альтернативного пути журналов по умолчанию *для всех узлов*. Для одноразделных баз данных можно задавать относительный путь. Если утилита повтора транзакций не может найти следующий необходимый журнал, она возвращает имя этого журнала в SQLCA и операция восстановления с повтором транзакций останавливается. Если больше нет доступных журналов, используйте опцию STOP, чтобы завершить восстановление с повтором транзакций. Для незавершенных транзакций выполняется откат, чтобы база данных или табличное пространство остались в совместимом состоянии.

каталог-журналов ON NODE

В среде многораздельной базы данных эта опция позволяет задать для конкретного узла другой путь журналов, чтобы переопределить альтернативный путь журналов по умолчанию.

RECOVER DROPPED TABLE ID-отброшенной-таблицы

Указывает, что во время операции повтора транзакций нужно восстановить отброшенную таблицу. ID таблицы можно получить с помощью команды “LIST HISTORY” на стр. 343.

TO каталог-экспорта

Задает каталог, в который нужно записать файлы, содержащие данные этой таблицы. Этот каталог должен быть доступен для всех узлов.

API повтора транзакций базы данных

Синтаксис API на языке C

```
/* Файл: sqlutil.h */
/* API: Повтор транзакций для базы данных */
/* ... */
SQL_API_RC SQL_API_FN
  sqluroll (
    struct rfwd_input * pRfwdInput,
    struct rfwd_output * pRfwdOuput,
    struct sqlca * pSqlca);
/* ... */
```

Общий синтаксис API

```
/* Файл: sqlutil.h */
/* API: Повтор транзакций для базы данных */
/* ... */
SQL_API_RC SQL_API_RN
  sqlgroll (
    struct grfwd_input * grfwdin,
    struct rfwd_output * rfwdout,
    struct sqlca * sqlca);

SQL_STRUCTURE grfwd_input
{
   unsigned short DbAliasLen,
   unsigned short StopTimeLen,
   unsigned short UserNameLen,
   unsigned short PasswordLen,
   unsigned short OverflowLogPathLen,
   unsigned short ReportFileLen, /* ПРИМЕЧАНИЕ: Этот параметр более не исполь- */
                                 /* зуется для менеджера связей данных DB2.    */
   sqluint32 Version,
   char * pDbAlias,
   unsigned short CallerAction,
   char * pStopTime,
   char * pUserName,
   char * pPassword,
   char * pOverflowLogPath,
   unsigned short NumChngLgOvrflw,
   struct sqlurf_newlogpath * pChngLogOvrflw,
   unsigned short ConnectMode,
   struct sqlu_tablespace_bkrst_list * pTablespaceList,
   short AllNodeFlag,
   short NumNodes,
   SQL_PDB_NODE_TYPE * pNodeList,
   short NumNodeInfo,
   unsigned short DlMode,        /* ПРИМЕЧАНИЕ: Этот параметр более не исполь- */
                                 /* зуется для менеджера связей данных DB2.    */
   char * pReportFile,           /* ПРИМЕЧАНИЕ: Этот параметр более не исполь- */
                                 /* зуется для менеджера связей данных DB2.    */
   char * pDroppedTblID,
   char * pExportDir
}
/* ... */
```

Параметры API

pRfwdInput

Входной. Указатель на структуру *rfwd_input*. Дополнительную информацию об этой структуре смотрите в разделе “Структура данных: RFWD-INPUT” на стр. 173.

pRfwdOutput
Выходной. Указатель на структуру *rfwd_output*. Дополнительную информацию об этой структуре смотрите в разделе “Структура данных: RFWD-OUTPUT” на стр. 176.

DbAliasLen
Входной. 2-байтное целое без знака - длина алиаса базы данных в байтах.

StopTimeLen
Входной. 2-байтное целое без знака - длину параметра времени остановки в байтах. Задайте значение ноль, если время остановки не задается.

UserNameLen
Входной. Двухбайтное целое число без знака - длина в байтах имени пользователя. Задайте значение ноль, если имя пользователя не указано.

PasswordLen
Входной. Двухбайтное целое число без знака - длина пароля в байтах. Задайте значение ноль, если пароль не указан.

OverflowLogPathLen
Входной. Двухбайтное целое число без знака - длина альтернативного пути журналов в байтах. Задайте значение ноль, если путь журнала переполнения не задается.

ReportFileLen
Входной. Этот параметр сейчас не используется; он должен иметь нулевое значение.

Version
Входной. ID версии параметров повтора транзакций. Он определен как SQLUM_RFWD_VERSION.

pDbAlias
Входной. Строка, содержащая алиас базы данных. Это алиас, внесенный в системный каталог баз данных.

CallerAction
Входной. Задает выполняемое действие. Допустимые значения (заданы в `sqlutil`):

SQLUM_ROLLFWD
Повтор транзакций до момента времени, заданного значением *pPointInTime*. При выполнении восстановления с повтором для базы данных эта база данных остается в состоянии отложенного повтора транзакций. При выполнении операции повтора транзакций до момента времени уровня табличных пространств эти табличные пространства остаются в состоянии ″идет повтор транзакций″.

SQLUM_STOP
Остановить восстановление с повтором транзакций. Оставшиеся записи журналов не обрабатываются и для непринятых транзакций выполняется откат. Состояние *отложенного повтора транзакций* для базы данных или табличных пространств отключается. Синоним - SQLUM_COMPLETE.

SQLUM_ROLLFWD_STOP
Выполнить повтор транзакций до момента времени, заданного значением *pPointInTime*, и закончить восстановление с повтором. Состояние отложенного повтора транзакций для базы данных или табличных пространств отключается. Синоним - SQLUM_ROLLFWD_COMPLETE.

SQLUM_QUERY
Запросить значения для *pNextArcFileName*, *pFirstDelArcFileName*, *pLastDelArcFileName* и *pLastCommitTime*. Возвращает информацию о состоянии базы данных и номер узла.

SQLUM_PARM_CHECK
Проверить параметры, не выполняя операции повтора транзакций.

SQLUM_CANCEL
Отменить выполняемую в настоящее время операцию повтора транзакций. База данных или табличное пространство переводятся в состояние отложенного восстановления.

Примечание: Эту опцию можно использовать непосредственно во время выполнения операции повтора транзакций. Ее нельзя использовать, если эта операция приостановлена (ожидает остановки), или если во время операции повтора транзакций возникла системная ошибка. Используйте эту опцию с осторожностью.

Для операции повтора транзакций может потребоваться загрузка копии с ленты. Функция API повтора транзакций возвращает предупреждение, если для работы с устройством требуется вмешательство пользователя. Эту функцию API можно вызвать снова с одним из трех следующих действий:

SQLUM_LOADREC_CONTINUE
Продолжить использование устройства, для которого было выдано предупреждение (например, когда смонтирована новая лента).

SQLUM_LOADREC_DEVICE_TERMINATE
Прекратить использование устройства, для которого было выдано предупреждение (например, если больше нет лент).

SQLUM_LOADREC_TERMINATE
Прекратить использовать все устройства, занятых в операции восстановления с повтором.

pStopTime
Входной. Символьная строка, содержащая значение отметки времени в формате ISO. Операция восстановления базы данных будет остановлена, когда пройдет заданный момент времени. Чтобы операция восстановления с повтором продолжалась сколь угодно долго, задайте опцию SQLUM_INFINITY_TIMESTAMP. Может иметь пустое значение (NULL) для опций действия SQLUM_QUERY, SQLUM_PARM_CHECK и любой из опций загрузки образа (SQLUM_LOADREC_*xxx*).

pUserName
Входной. Строка, содержащая имя пользователя этой прикладной программы. Может иметь пустое значение (NULL).

pPassword
Входной. Строка, содержащая пароль для указанного имени пользователя (если оно указано). Может иметь пустое значение (NULL).

pOverflowLogPath
Входной. Этот параметр задает альтернативный путь журналов. Чтобы можно было использовать эту утилиту, пользователь должен в добавление к активным файлам журналов переместить в каталог *logpath* архивированные файлы журналов (смотрите раздел “sqlfxdb - Получить конфигурацию базы данных” в книге *Administrative API Reference*). Если в каталоге *logpath* недостаточно места, могут возникнуть сложности. В таких случаях используется альтернативный путь журналов. При восстановлении с повтором поиск сначала ведется в файлах журналов в каталоге *logpath*, а затем в альтернативном пути журналов. Файлы журналов, необходимые для выполнения восстановления с повтором транзакций для табличного пространства, можно поместить в каталог *logpath* или в альтернативный путь журналов. Если альтернативный путь журналов не задан в вызывающей программе, по умолчанию используется значение *logpath*. В среде многораздельных баз данных альтернативный путь журналов должен быть правильным полным путем; по умолчанию используется альтернативный путь журналов для каждого узла. В одноразделной среде альтернативный путь журналов может быть относительным, если используется локальный сервер.

NumChngLgOvrflw
Только для конфигураций с массовым параллелизмом (MPP). Число

измененных альтернативных путей журналов. Эти новые альтернативные пути журналов переопределяют пути по умолчанию только для указанного узла.

pChngLogOvrflw
Только для конфигураций с массовым параллелизмом (MPP). Указатель на структуру, содержащую полные имена измененных альтернативных путей журналов. Эти новые альтернативные пути журналов переопределяют пути по умолчанию только для указанного узла.

ConnectMode
Входной. Допустимые значения (определенные в `sqlutil`):

SQLUM_OFFLINE
Повтор транзакций в автономном режиме. Для операции восстановления с повтором базы данных должно быть задано это значение.

SQLUM_ONLINE
Повтор транзакций в оперативном режиме.

pTablespaceList
Входной. Указатель на структуру, содержащую имена табличных пространств, для которых нужно выполнить повтор транзакций до конца журналов или до момента времени. Если этот параметр не задан, выбираются те табличные пространства, для которых требуется повтор транзакций.

AllNodeFlag
Только для конфигураций с массовым параллелизмом (MPP). Входной. Указывает, должна ли операция повтора транзакций применяться ко всем узлам, определенным в `db2nodes.cfg`. Допустимые значения:

SQLURF_NODE_LIST
Применить операцию повтора транзакций к узлам, входящим в список узлов, передаваемый в параметре *pNodeList*.

SQLURF_ALL_NODES
Применить операцию повтора транзакций ко всем узлам. Параметр *pNodeList* должен быть пустым (NULL). Это значение по умолчанию.

SQLURF_ALL_EXCEPT
Применить операцию повтора транзакций ко всем узлам, кроме узлов, входящих в список узлов, передаваемый в параметре *pNodeList*.

SQLURF_CAT_NODE_ONLY
Применить операцию повтора транзакций только к узлу каталога. Параметр *pNodeList* должен быть пустым (NULL).

NumNodes
Входной. Задает число узлов в массиве *pNodeList*.

pNodeList
Входной. Указатель на массив номеров узлов, для которых нужно выполнить восстановление с повтором транзакций.

NumNodeInfo
Входной. Задает размер выходного параметра *pNodeInfo*; он должен быть достаточно велик, чтобы вместить информацию о состоянии для всех узлов, для которых выполняется повтор транзакций. В одноразделной среде этому параметру следует присвоить значение 1. Значение этого параметра должно совпадать с числом узлов, для которых вызывается эта функция API.

DlMode
Входной. Этот параметр сейчас не используется; он должен иметь нулевое значение.

pReportFile
Входной. Этот параметр сейчас не используется; он должен иметь пустое значение (NULL).

pDroppedTblID
Входной. Строка, содержащая ID отброшенной таблицы, которую нужно восстановить.

pExportDir
Входной. Каталог, в который будут экспортироваться данные отброшенной таблицы.

pSqlca Выходной. Указатель на структуру *sqlca*. Дополнительную информацию об этой структуре смотрите в публикациях *Administrative API Reference* или *SQL Reference*.

Синтаксис API REXX

```
ROLLFORWARD DATABASE алиас-базы-данных USING :значение ] [USER имя-пользователя
USING пароль]
[условие-действия-повтора-транзакций | условие-действия-загрузки-копии]
где условие-действия-повтора-транзакций означает:
    { TO момент-времени [AND STOP] |
      {
        [TO END OF LOGS [AND STOP] | STOP | CANCEL | QUERY STATUS | PARM CHECK }
        [ON {:список-узлов | ALL NODES [EXCEPT :список-узлов]}]
      }
    }
    [TABLESPACE {ONLINE |:имена-табличных-пространств [ONLINE]} ]
    [OVERFLOW LOG PATH путь-журналов-по-умолчанию [:пути-журналов]]
а условие-действия-загрузки-копии означает:
    LOAD RECOVERY { CONTINUE | DEVICE_TERMINATE | TERMINATE }
```

Параметры API REXX

алиас-базы-данных
Алиас базы данных, для которой выполняется повтор транзакций.

значение
Составная переменная хоста REXX, содержащая выходные значения. Далее XXX будет обозначать имя этой переменной хоста:

XXX.0	Число элементов в этой переменной
XXX.1	ID программы
XXX.2	Число ответов, полученных от узлов
XXX.2.1.1	Номер первого узла
XXX.2.1.2	Для первого узла: информация о состоянии
XXX.2.1.3	Для первого узла: следующий требующийся файл архива
XXX.2.1.4	Для первого узла: первый файл архива, который нужно удалить
XXX.2.1.5	Для первого узла: последний файл архива, который нужно удалить
XXX.2.1.6	Для первого узла: время последней операции принятия
XXX.2.2.1	Номер второго узла
XXX.2.2.2	Для второго узла: информация о состоянии
XXX.2.2.3	Для второго узла: следующий нужный файл архива
XXX.2.2.4	Для второго узла: первый файл архива, который нужно удалить
XXX.2.2.5	Для второго узла: последний файл архива, который нужно удалить
XXX.2.2.6	Для второго узла: время последней операции принятия
XXX.2.3.x	и так далее.

имя-пользователя
Указывает имя пользователя, под которым должна выполняться операция повтора транзакций для базы данных.

пароль Пароль, используемый для аутентификации имени пользователя.

момент-времени
Момент времени в формате ISO, *гггг-мм-дд-чч.мм.сс.нннннн* (год, месяц, день, час, минуты, секунды, микросекунды) согласованного универсального времени (UTC).

имена-табличных-пространств

Составная переменная хоста REXX, содержащая список табличных пространств, для которых нужно выполнить повтор транзакций. Далее XXX будет обозначать имя этой переменной хоста:

XXX.0 Число табличных пространств, для которых нужно выполнить повтор транзакций

XXX.1 Имя первого табличного пространства

XXX.2 Имя второго табличного пространства

XXX.x и так далее.

путь-журналов-по-умолчанию

Альтернативный путь журналов по умолчанию, в котором нужно искать архивированные журналы во время операции восстановления

пути-журналов

Составная переменная хоста REXX, содержащая список альтернативных путей журналов, в которых нужно искать архивированные журналы во время операции восстановления. Далее XXX будет обозначать имя этой переменной хоста:

XXX.0 Число измененных альтернативных путей журналов

XXX.1.1 Первый узел

XXX.1.2 Альтернативный путь журналов для первого узла

XXX.2.1 Второй узел

XXX.2.2 Альтернативный путь журналов для второго узла

XXX.3.1 Третий узел

XXX.3.2 Альтернативный путь журналов для третьего узла

XXX.x.1 и так далее.

список-узлов

Составная переменная хоста REXX, содержащая список узлов. Далее XXX будет обозначать имя этой переменной хоста:

XXX.0 Число узлов

XXX.1 Первый узел

XXX.2 Второй узел

XXX.x и так далее.

Структура данных: RFWD-INPUT

Эта структура используется для передачи информации “API повтора транзакций базы данных” на стр. 164.

Таблица 8. Поля структуры RFWD-INPUT

Имя поля	Тип данных	Описание
VERSION	sqluint32	Версия повтора транзакций.
PDBALIAS	Pointer	Алиас базы данных.
CALLERACTION	UNSIGNED SHORT	Действие.
PSTOPTIME	Pointer	Время остановки.
PUSERNAME	Pointer	Имя пользователя.
PPASSWORD	Pointer	Пароль.
POVERFLOWLOGPATH	Pointer	Альтернативный путь журналов.
NUMCHNGLGOVRFLW	UNSIGNED SHORT	Число измененных альтернативных путей журналов (только для MPP).
PCHNGLOGOVRFLW	Structure	Измененные альтернативные пути журналов (только для MPP).
CONNECTMODE	UNSIGNED SHORT	Режим соединения.
PTABLESPACELIST	Structure	Указатель на список имен табличных пространств. Информацию об этой структуре смотрите в разделе “Структура данных: SQLU-TABLESPACE-BKRST-LIST” на стр. 112.
ALLNODEFLAG	SHORT	Флаг применения ко всем узлам.
NUMNODES	SHORT	Размер списка узлов.
PNODELIST	Pointer	Список номеров узлов.
NUMNODEINFO	SHORT	Размер *pNodeInfo* в “Структура данных: RFWD-OUTPUT” на стр. 176.
DLMODE	UNSIGNED SHORT	Этот параметр сейчас не используется.
PREPORTFILE	Pointer	Этот параметр сейчас не используется.
PDROPPEDTBLID	Pointer	Строка, содержащая ID отброшенной таблицы, для которой предпринимается попытка восстановления.
PEXPORTDIR	Pointer	Каталог, куда будут экспортироваться данные отброшенной таблицы.
NODENUM	SQL_PDB_NODE_TYPE	Номер узла.
PATHLEN	UNSIGNED SHORT	Длина нового пути журналов.
LOGPATH	CHAR(255)	Новый альтернативный путь журналов.

Структура данных: RFWD-INPUT

Синтаксис языка

Структура на языке C

```
/* Файл: sqlutil.h */
/* Структура: RFWD-INPUT */
/* ... */
SQL_STRUCTURE rfwd_input
{
  sqluint32              version;
  char                   *pDbAlias;
  unsigned short         CallerAction;
  char                   *pStopTime;
  char                   *pUserName;
  char                   *pPassword;
  char                   *pOverflowLogPath;
  unsigned short         NumChngLgOvrflw;
  struct sqlurf_newlogpath *pChngLogOvrflw;
  unsigned short         ConnectMode;
  struct sqlu_tablespace_bkrst_list *pTablespaceList;
  short                  AllNodeFlag;
  short                  NumNodes;
  SQL_PDB_NODE_TYPE      *pNodeList;
  short                  NumNodeInfo;
  unsigned short         DlMode;         /* Этот параметр сейчас */
                                         /*  не используется.    */
  char                   *pReportFile;   /* Этот параметр сейчас */
                                         /*  не используется.    */
  char                   *pDroppedTblID;
  char                   *pExportDir;
};
/* ... */

/* Файл: sqlutil.h */
/* Структура: SQLURF-NEWLOGPATH */
/* ... */
SQL_STRUCTURE sqlurf_newlogpath
{
  SQL_PDB_NODE_TYPE      nodenum;
  unsigned short         pathlen;
  char                   logpath[SQL_LOGPATH_SZ+SQL_LOGFILE_NAME_SZ+1];
};
/* ... */
```

Структура на языке COBOL

```
* Файл: sqlutil.cbl
01 SQL-RFWD-INPUT.
    05 SQL-VERSION            PIC 9(9) COMP-5.
    05 SQL-DBALIAS            USAGE IS POINTER.
    05 SQL-CALLERACTION       PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-STOPTIME           USAGE IS POINTER.
    05 SQL-USERNAME           USAGE IS POINTER.
    05 SQL-PASSWORD           USAGE IS POINTER.
    05 SQL-OVERFLOWLOGPATH    USAGE IS POINTER.
    05 SQL-NUMCHANGE          PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-P-CHNG-LOG-OVRFLW  USAGE IS POINTER.
    05 SQL-CONNECTMODE        PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-P-TABLESPACE-LIST  USAGE IS POINTER.
    05 SQL-ALLNODEFLAG        PIC S9(4) COMP-5.
    05 SQL-NUMNODES           PIC S9(4) COMP-5.
    05 SQL-NODELIST           USAGE IS POINTER.
    05 SQL-NUMNODEINFO        PIC S9(4) COMP-5.
    05 SQL-DLMODE             PIC 9(4) COMP-5.  * Этот параметр сейчас
                                                *   не используется.
    05 SQL-REPORTFILE         USAGE IS POINTER. * Этот параметр сейчас
                                                *   не используется.
    05 SQL-DROPPEDTBLID       USAGE IS POINTER.
    05 SQL-EXPORTDIR       USAGE IS POINTER.
*

* Файл: sqlutil.cbl
01 SQLURF-NEWLOGPATH.
    05 SQL-NODENUM            PIC S9(4) COMP-5.
    05 SQL-PATHLEN            PIC 9(4) COMP-5.
    05 SQL-LOGPATH            PIC X(254).
    05 FILLER                 PIC X.
    05 FILLER                 PIC X(1).
*
```

Структура данных: RFWD-OUTPUT

Данная структура используется для передачи информации от “API повтора транзакций базы данных” на стр. 164 .

Таблица 9. Поля структуры RFWD-OUTPUT

Имя поля	Тип данных	Описание
PAPPLICATIONID	Pointer	Адрес буфера длиной SQLU_APPLID_LEN+1 (задается в sqlutil) для хранения идентификатора программы, возвращаемого API. Этот идентификатор может быть использован системным монитором API базы данных для мониторинга программы. Если в этом нет необходимости, задайте указатель NULL. В среде многораздельных баз данных возвращается только идентификатор программы для узла каталога.
PNUMREPLIES	Pointer	Число полученных от узлов ответов. Каждый отвечающий узел заполняет структуру *sqlurf_info* в *pNodeInfo*. В одноразделной среде этот параметр имеет значение 1.
PNODEINFO	Structure	Информация ответа узла. Определенный пользователем массив из *NumNodeInfo* структур *sqlurf_info*.

Таблица 10. Поля структуры SQLURF-INFO

Имя поля	Тип данных	Описание
NODENUM	SQL_PDB_NODE_TYPE	Номер узла.
STATE	LONG	Информация о состоянии.
NEXTARCLOG	UNSIGNED CHAR(13)	12-байтный буфер для хранения возвращенного имени следующего требуемого архивированного файла журнала. Если действие вызывающего - не SQLUM_QUERY, величина, возвращаемая в этом поле, указывает, что при обращении к файлу произошла ошибка. Возможные причины: • Файл не был найден ни в каталоге журналов базы данных, ни по пути, указанному параметром пути переполнения журналов. • Программа обработчика пользователя не смогла возвратить архивированный файл.

Таблица 10. Поля структуры SQLURF-INFO (продолжение)

Имя поля	Тип данных	Описание
FIRSTARCDEL	UNSIGNED CHAR(13)	12-байтный буфер для хранения возвращенного имени первого архивированного файла журнала, который больше не потребуется для восстановления. Этот файл, как и все другие файлы вплоть до *lastarcdel*, можно переместить, чтобы освободить место на диске. Например, если в *firstarcdel* и *lastarcdel* возвращены значения `S0000001.LOG` и `S0000005.LOG`, можно переместить следующие файлы журнала: • `S0000001.LOG` • `S0000002.LOG` • `S0000003.LOG` • `S0000004.LOG` • `S0000005.LOG`
LASTARCDEL	UNSIGNED CHAR(13)	12-байтный буфер для хранения возвращенного имени последнего архивированного файла журнала, который можно удалить из каталога журналов базы данных.
LASTCOMMIT	UNSIGNED CHAR(27)	26-символьная строка, содержащая отметку времени в формате ISO. Это значение содержит отметку времени последней принятой транзакции после прекращения операции восстановления с повтором транзакций.

Допустимые значения для *STATE* (определенные в `sqlutil`):

SQLURFQ_NOT_AVAILABLE
Не удается соединиться с узлом.

SQLURFQ_NOT_RFW_PENDING
База данных не находится в состоянии отложенного повтора транзакций.

SQLURFQ_DB_RFW_PENDING
База данных находится в состоянии отложенного повтора транзакций.

SQLURFQ_TBL_RFW_PENDING
Табличное пространство находится в состоянии отложенного повтора транзакций.

SQLURFQ_DB_RFW_IN_PROGRESS
База данных находится в состоянии ″идет повтор транзакций″.

SQLURFQ_TBL_RFW_IN_PROGRESS
Табличное пространство находится в состоянии ″идет повтор транзакций″.

SQLURFQ_DB_RFW_STOPPING
Операция восстановления базы данных с повтором транзакций была прервана в результате получения требования STOP.

SQLURFQ_TBL_RFW_STOPPING
Операция восстановления табличного пространства с повтором транзакций была прервана в результате получения требования STOP.

Синтаксис языка

Структура на языке C

```
/* Файл: sqlutil.h */
/* Структура: RFWD-OUTPUT */
/* ... */
SQL_STRUCTURE rfwd_output
{
  char            *pApplicationId;
  long            *pNumReplies;
  struct sqlurf_info *pNodeInfo;
};
/* ... */

/* Файл: sqlutil.h */
/* Структура: SQLURF-INFO */
/* ... */
SQL_STRUCTURE sqlurf_info
{
  SQL_PDB_NODE_TYPE nodenum;
  long            state;
  unsigned char   nextarclog[SQLUM_ARCHIVE_FILE_LEN+1];
  unsigned char   firstarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
  unsigned char   lastarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
  unsigned char   lastcommit[SQLUM_TIMESTAMP_LEN+1];
};
/* ... */
```

Структура на языке COBOL

```
* Файл: sqlutil.cbl
01 SQL-RFWD-OUTPUT.
    05 SQL-APPLID             USAGE IS POINTER.
    05 SQL-NUMREPLIES         USAGE IS POINTER.
    05 SQL-P-NODE-INFO        USAGE IS POINTER.
*
```

Структура данных: RFWD-OUTPUT

```
* Файл: sqlutil.cbl
01 SQLURF-INFO.
    05 SQL-NODENUM           PIC S9(4) COMP-5.
    05 FILLER                PIC X(2).
    05 SQL-STATE             PIC S9(9) COMP-5.
    05 SQL-NEXTARCLOG        PIC X(12).
    05 FILLER                PIC X.
    05 SQL-FIRSTARCDEL       PIC X(12).
    05 FILLER                PIC X.
    05 SQL-LASTARCDEL        PIC X(12).
    05 FILLER                PIC X.
    05 SQL-LASTCOMMIT        PIC X(26).
    05 FILLER                PIC X.
    05 FILLER                PIC X(2).
*
```

Примеры сеансов с повтором транзакций

Примеры CLP

Пример 1

Команда ROLLFORWARD DATABASE допускает одновременное указание нескольких операций, которые разделяются ключевым словом AND. Например, чтобы повторить транзакции до конца журналов и завершить работу, можно объединить отдельные команды:

```
db2 rollforward db sample to end of logs
db2 rollforward db sample complete
```

следующим образом:

```
db2 rollforward db sample to end of logs and complete
```

Эти два выражения эквивалентны, однако все же рекомендуется разбивать такие операции на два шага. Важно убедиться, что операция повтора транзакций прошла правильно, перед тем, как останавливать работу с возможной потерей журналов. Особенно это важно, если в процессе восстановления с повтором транзакций обнаружится дефектный журнал, который будет интерпретирован как “последний журнал”. В таких случаях для продолжения операции повтора транзакций с последующими журналами может быть использована неповрежденная резервная копия этого журнала.

Пример 2

Повторить транзакции до конца журналов (было восстановлено два табличных пространства):

```
db2 rollforward db sample to end of logs
db2 rollforward db sample to end of logs and stop
```

Эти два оператора эквивалентны. Для восстановления табличного пространства с повтором транзакций до конца журналов не требуется ни AND STOP, ни AND COMPLETE. Не требуются также имена табличных пространств. Если они не указаны, будут включены все табличные пространства, для которых требуется восстановление с повтором транзакций. Имена необходимо указывать только в том случае, когда повтор транзакций выполняется для части этих табличных пространств.

Пример 3

После восстановления трех табличных пространств из резервной копии повторить для одного из них транзакции до конца файлов журнала, а для двух остальных - до определенного момента времени; обе операции выполнить оперативно:

```
db2 rollforward db sample to end of logs tablespace(TBS1) online

db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS2, TBS3) online
```

Обратите внимание на то, что две операции повтора транзакций нельзя запускать одновременно. Вторую команду можно вызвать только после успешного завершения первой операции повтора транзакций.

Пример 4

После восстановления базы данных из резервной копии повторить транзакции до определенного момента времени, используя OVERFLOW LOG PATH для указания каталога, в котором обработчик пользователя сохраняет архивированные журналы:

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   overflow log path (/logs)
```

Пример 5 (MPP)

Есть три узла: 0, 1 и 2. Табличное пространство TBS1 определено на всех узлах, а табличное пространство TBS2 - на узлах 0 и 2. После восстановления базы данных из резервной копии на узле 1, а TBS1 на узлах 0 и 2, повторить транзакции для базы данных на узле 1:

```
db2 rollforward db sample to end of logs and stop
```

При этом будет возвращено предупреждение SQL1271 (“База данных восстановлена, но одно или несколько табличных пространств на узле (узлах) 0 и 2 отключены.”).

```
db2 rollforward db sample to end of logs
```

Это приведет к повтору транзакций для TBS1 на узлах 0 и 2. В данном случае условие TABLESPACE(TBS1) - необязательное.

Пример 6 (MPP)

После восстановления табличного пространства TBS1 из резервной копии только на узлах 0 и 2 повторить транзакции для TBS1 на узлах 0 и 2:

```
db2 rollforward db sample to end of logs
```

Узел 1 игнорируется.

```
db2 rollforward db sample to end of logs tablespace(TBS1)
```

Эта операция завершится неудачно, поскольку TBS1 не готово для восстановления с повтором транзакций на узле 1. Будет получено сообщение SQL4906N.

```
db2 rollforward db sample to end of logs on nodes (0, 2) tablespace(TBS1)
```

Эта операция завершится успешно.

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

Эта операция завершится неудачно, поскольку TBS1 не готово для восстановления с повтором транзакций на узле 1, а повтор транзакций для всех частей должен проводиться совместно.

Примечание: При повторе транзакций для табличного пространства до определенного момента времени условие для узлов не воспринимается. Операция повтора транзакций должна выполняться на всех узлах, на которых расположено табличное пространство.

После восстановления TBS1 на узле 1:

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

Эта операция завершится успешно.

Пример 7 (MPP)

После восстановления табличного пространства из резервной копии на всех узлах повторить транзакции до PIT2, но не указывать AND STOP. Операция повтора транзакций еще не закончена. Отменить и повторить транзакции до PIT1:

```
db2 rollforward db sample to pit2 tablespace(TBS1)
db2 rollforward db sample cancel tablespace(TBS1)

** восстановить TBS1 на всех узлах **

db2 rollforward db sample to pit1 tablespace(TBS1)
db2 rollforward db sample stop tablespace(TBS1)
```

Пример 8 (MPP)

Восстановить с повтором транзакций табличное пространство, которое расположено на восьми узлах (3 - 10), перечисленных в файле db2nodes.cfg:

```
db2 rollforward database dwtest to end of logs tablespace (tssprodt)
```

Эта операция успешно выполняется до конца журналов (не до определенного момента времени). Узлы, на которых расположено табличное пространство, указывать не требуется. Утилита по умолчанию использует файл db2nodes.cfg.

Пример 9 (MPP)

Восстановить с повтором транзакций шесть небольших табличных пространств, расположенных на одноузловой группе узлов (на узле 6):

```
db2 rollforward database dwtest to end of logs on node (6)
   tablespace(tsstore, tssbuyer, tsstime, tsswhse, tsslscat, tssvendor)
```

Эта операция успешно выполняется до конца журналов (не до определенного момента времени).

Пример командного сценария DB2 и информация о его использовании приведены в разделе “Приложение F. Сценарий восстановления CLP” на стр. 457.

Примеры API

Примеры программ, содержащих API DB2 и встроенные вызовы SQL, а также информация об их использовании приведены в разделе “Приложение E. Программы примеров восстановления” на стр. 389.

Ограничения на повтор транзакций

На утилиту повтора транзакций накладываются следующие ограничения:

- Одновременно можно вызвать только одну операцию повтора транзакций. Если требуется восстановить несколько табличных пространств, все их можно указать в одной и той же операции.
- Если после последней операции резервного копирования вы переименовали табличное пространство, при повторе транзакций для него надо использовать новое имя. Предыдущее имя табличного пространства не будет опознаваться.
- Операцию повтора транзакций нельзя отменить во время выполнения. Можно отменить только завершенную операцию повтора транзакций, для которой не была указана опция STOP, или операцию повтора транзакций, в которой до ее завершения произошла ошибка.
- Нельзя *продолжать* операцию повтора транзакций табличного пространства до определенного момента времени, указав отметку времени более раннюю, чем предыдущая. Если момент времени не указан, используется предыдущий. Можно инициировать операцию повтора транзакций до определенного момента времени, просто указав STOP, но это разрешено только в том случае, когда все восстанавливаемые табличные пространства были восстановлены из одного и того же образа резервной копии, сделанного в автономном режиме. В этом случае не требуется обработки журналов. Если вы начинаете другую операцию повтора транзакций с другим списком табличных пространств до того, как текущая операция повтора транзакций была завершена или отменена, возвращается сообщение об ошибке (SQL4908). Вызовите команду LIST TABLESPACES на всех узлах, чтобы определить, для каких табличных пространств в данное время выполняется повтор транзакций (состояние выполнения повтора транзакций), а для какие табличные пространства готовы к повтору транзакций (состояние ожидания повтора транзакций). Есть три возможности:

- Закончить текущую операцию повтора транзакций для всех табличных пространств.
- Закончить текущую операцию повтора транзакций для поднабора табличных пространств. (Это может быть невозможно, если операция повтора транзакций должна закончится до определенного момента времени, для чего требуется участие всех узлов.)
- Отменить текущую операцию повтора транзакций.

Устранение ошибок при повторе транзакций

Не восстанавливайте табличные пространства, не отказавшись от текущей операции повтора транзакций; в противном случае вы можете получить набор табличных пространств, в котором некоторые табличные пространства находятся в состоянии выполнения повтора транзакций, а другие - в состоянии ожидания повтора транзакций. Текущая операция повтора транзакций может работать только с табличными пространствами, находящимися в состоянии выполнения повтора транзакций.

Если утилита повтора транзакций встречает невосстановимую операцию (например, загрузку без копирования), связанное табличное пространство переводится в состояние ожидания восстановления из резервной копии. Чтобы вывести табличное пространство из состояние ожидания восстановления из резервной копии, необходимо выполнить восстановление из наиболее позднего образа резервной копии уровня базы данных или табличного пространства.

Сообщение SQL1271 возвращается (даже если еще не было выдано требование AND STOP) как предупреждение о том, что есть табличное пространство в состоянии ожидания повтора транзакций или ожидания восстановления из резервной копии. Во время операции повтора транзакций базы данных это означает, что утилита повтора транзакций использовала одно из табличных пространств в автономном состоянии. Во время операции повтора транзакций *табличных пространств* это может означать, что одно из них использовано утилитой повтора транзакций в автономном состоянии или что есть другое (отсутствующее в списке) табличное пространство, для которого все еще требуется повтор транзакций.

Сообщение SQL1272 возвращается, если все табличные пространства, для которых выполнялся повтор транзакций, были использованы утилитой повтора транзакций в автономном состоянии (либо указанные в списке, либо из-за того, что они находились в состоянии ожидания повтора транзакций). Это может означать, что в них содержится таблица, подвергшаяся невосстановимой операции загрузки, или таблица с опцией обработки NOT LOGGED INITIALLY. Если возвращено это сообщение об ошибке, операция повтора транзакций табличных пространств должна быть остановлена (то есть не должна продолжаться).

Часть 2. Системы высокой доступности

© Copyright IBM Corp. 2001

Глава 5. Высокая доступность и восстановление при отказах. Введение

Успех электронной коммерции зависит от непрерывной доступности систем обработки транзакций, которые, в свою очередь, управляются системами управления базами данных, такими как DB2, которые должны быть доступны 24 часа в сутки и 7 дней в неделю (“24 x 7”).

Высокая доступность

Высокая доступность (High availability, HA) - термин, применяемый для описания систем, работающих и доступных клиентам в любой момент времени. Для этого:

- Транзакции должны обрабатываться эффективно, то есть без значительного понижения производительности (или даже потери доступности) во время максимальной нагрузки. В среде многораздельных баз данных DB2 для эффективной обработки транзакций может использовать преимущества как внутрираздельного, так и межраздельного параллелизма. *Внутрираздельный параллелизм* можно использовать в среде SMP для одновременной обработки различных составляющих сложного оператора SQL. С другой стороны, *межраздельный параллелизм* в среде многораздельных баз данных, означает одновременную обработку запроса на всех участвующих узлах; каждый узел обрабатывает свой набор строк таблицы. Дополнительную информацию о параллелизме смотрите в книге *Administration Guide* .
- Должна быть возможность быстро восстановить систему, если произошла аппаратная или программная ошибка или случилась авария. Для этого может оказаться полезным использовать журнальная файловую систему. *Журнальная файловая система* - это система, автоматически записывающая изменения в своей структуре в журнал. Это гарантирует сохранность всех изменений в структуре файловой системы при отказах системы, если сам носитель информации не был утерян или испорчен. После отказа системы над файловой системой производятся изменения, записанные в этих журналах, в порядке их записи в журнале. Для журнальной файловой системы характерна высокая скорость восстановления. Такая система предпочтительна для сред с большими файловыми системами и в случаях, когда целостность данных особенно важна.

  Способность системы быстро восстанавливаться существенно зависит от наличия надежной резервной копии и алгоритма ее восстановления. Дополнительную информацию об алгоритмах восстановления смотрите в разделе “Глава 1. Разработка правильной стратегии резервного копирования и восстановления” на стр. 3 .

© Copyright IBM Corp. 2001

- Программное обеспечение, обеспечивающее работу базы данных предприятия, должно быть пригодно для непрерывной работы и обработки транзакций. Для непрерывности работы менеджера баз данных надо, чтобы в случае отказа выполнение его функций взял на себя другой менеджер баз данных. Это называется восстановлением при отказах. Возможность *восстановления при отказах* позволяет при аппаратных отказах автоматически передавать рабочую нагрузку с одной системы на другую.

Для обеспечения восстановления при отказах можно хранить копию базы данных на другом компьютере, который будет постоянно повторять транзакции из файлов журналов. *Отправка журналов* - это процесс копирования всех файлов журналов на резервный компьютер с устройства архивации или через программу обработчика пользователя, выполняемую для исходной базы данных. При таком подходе исходная база данных восстанавливается на резервном компьютере либо с помощью утилиты восстановления DB2, либо с помощью функции отделения зеркальной копии. Для быстрой инициализации новой базы данных можно использовать новую возможность приостановки ввода-вывода (смотрите раздел “Высокая доступность с помощью оперативного отделения зеркальной копии и поддержки приостановленного ввода/вывода” на стр. 190). Вторичная база данных на резервном компьютере постоянно повторяет транзакции из файлов журналов. Если в первичной базе данных происходит отказ, все оставшиеся файлы журналов копируются на резервный компьютер. После повтора всех транзакций до конца журналов и прекращения операции все клиенты подсоединяются к вторичной базе данных на резервном компьютере.

Восстановление при отказах можно также обеспечить с помощью программ для конкретных платформ, которые можно добавить в систему. Например:

- HACMP/ES (High Availability Cluster Multi-Processing, Enhanced Scalability - мультипроцессорная кластерная обработка с высокой доступностью и улучшенной масштабируемостью) для AIX.

  Подробную информацию о HACMP/ES смотрите в разделе “Глава 6. Высокая доступность в AIX” на стр. 193, или в документе под названием “IBM DB2 Universal Database Enterprise Edition для AIX и HACMP/ES”, который можно найти на сайте “DB2 UDB and DB2 Connect Online Support” (http://www.ibm.com/software/data/pubs/papers/).

- Microsoft Cluster Server для Windows NT или Windows 2000.

  Подробную информацию о MSCS смотрите в разделе “Глава 7. Высокая доступность в операционной системе Windows” на стр. 239 .

- Серверы Sun Cluster или VERITAS Cluster для операционной среды Solaris.

  Информацию о Sun Cluster 2.x смотрите в разделе “Глава 8. Высокая доступность в операционной среде Solaris” на стр. 273 ; информацию о Sun Cluster 3.0 смотрите в документе под названием “DB2 and High Availability on Sun Cluster 3.0” на сайте “DB2 UDB and DB2 Connect Online Support”

(http://www.ibm.com/software/data/pubs/papers/). Информацию о VERITAS Cluster Server смотрите в документе под названием “DB2 and High Availability on VERITAS Cluster Server”, который также можно найти на сайте “DB2 UDB and DB2 Connect Online Support”.

- Multi-Computer/ServiceGuard для Hewlett-Packard.

  Подробную информацию о HP MC/ServiceGuard смотрите в документе под названием “IBM DB2 EE v.7.1 Implementation and Certification With Hewlett-Packard’s MC/ServiceGuard High Availability Software”, который можно найти на сайте (http://www.ibm.com/software/data/pubs/papers/).

Алгоритмы восстановления при отказах обычно основаны на использовании кластеров из систем. *Кластер* - это группа связанных систем, работающих совместно, как одна система. Каждый процессор рассматривается как узел в кластере. Кластеризация позволяет серверам подстраховывать друг друга: при отказе рабочую нагрузку сервера, на котором произошел отказ, принимает на себя другой сервер кластера.

Подмена IP-адреса - это возможность переносить IP-адрес сервера с одного компьютера на другой при отключении сервера; для пользователей эти два компьютера в разные моменты времени представляются одним сервером.

Программное обеспечение для восстановления при отказах для проверки связи между системами использует *мониторинг работоспособности* или *пакеты подтверждения активности*. Мониторинг работоспособности выполняется системными службами, которые обеспечивают постоянную связь между узлами кластера. Если пропадают сигналы работоспособности, начинается восстановление на резервную систему. Конечные пользователи обычно не замечают, что в работе системы был сбой.

Два наиболее часто используемых алгоритма восстановления при отказах - это *горячее резервирование* и *взаимная подмена*; в зависимости от производителя эти конфигурации могут называться по-другому.

Горячее резервирование
: В этом режиме одна система используется для запуска экземпляра DB2, а вторая система “ожидает” или находится в режиме резервирования; она готова в случае сбоя операционной системы или аппаратного обеспечения, затрагивающего первую систему, принять на себя обслуживание экземпляра. Общая производительность системы не изменяется, так как пока в резервной системе нет потребности, она не несет нагрузки.

Взаимная подмена
: В этом режиме каждая система является одновременно и резервной копией другой системы. Общая производительность системы может снизиться, так как для восстановления после отказа система должна

выполнять дополнительную работу: собственную работу плюс работу, которая выполнялась отказавшей системой.

Алгоритмы восстановления при отказах могут применятся для восстановления экземпляра, раздела или нескольких логических узлов.

Высокая доступность с помощью оперативного отделения зеркальной копии и поддержки приостановленного ввода/вывода

Приостановленный ввод/вывод обеспечивает постоянную доступность системы, полностью реализуя оперативное отделение зеркальной копии, то есть отделение зеркальной копии без закрытия базы данных. *Зеркальная копия* - это “мгновенная” копия базы данных, для ее получения можно поддерживать точную копию дисков с данными и отделить эту зеркальную копию, когда потребуется. *Зеркальное копирование дисков* - это процесс параллельной записи всех данных на два отдельных жестких диска, один из них будет зеркальной копией другого. *Отделение* зеркальной копии - это процесс снятия резервной копии с зеркальной.

Если вы не хотите снимать резервную копию с большой базы данных с помощью утилиты резервного копирования DB2, можно сделать копию с зеркальной копии с помощью приостановленного ввода/вывода и функции разделения зеркальной копии. Этот подход:

- Устраняет издержки от операции резервного копирования на основном компьютере
- Представляет собой быстрый способ клонирования систем
- Представляет собой быструю реализацию восстановления с помощью горячего резервирования. В этом случае нет предварительных операций для восстановления, и, в отличие от процедуры повтора транзакций, которая может оказаться слишком медленной или приводить к ошибкам, повторная инициализация происходит очень быстро.

Команда **db2inidb** инициализирует отделенную зеркальную копию, в результате ее можно использовать:

- Для создания клона базы данных

  Клон базы данных, предназначенный только для чтения, можно использовать, например, для создания отчетов.
- Как резервную базу данных
- Как резервную копию

Эту команду можно выполнить только для отделенной зеркальной копии, а для отделенной зеркальной копии надо выполнить **db2inidb**, прежде чем ее можно будет использовать (смотрите раздел “db2inidb - инициализировать зеркальную копию базы данных” на стр. 338).

В среде многораздельных баз данных надо выполнить команду **db2inidb** для всех разделов, прежде чем можно будет использовать отделенную копию какого-либо из разделов. Инструмент можно применять ко всем разделам одновременно.

Создание клона базы данных

Клон базы данных можно использовать как автономную "резервную копию" основной (действующей) базы данных. Однако с клонированной базы данных нельзя сделать резервную копию, восстановить эту копию на исходную систему и повторить транзакции из файлов журналов, созданных на исходной системе.

Чтобы клонировать базу данных, выполните следующие действия:

1. Приостановите операции ввода/вывода в исходной базе данных:
   ```
   db2 set write suspend for database
   ```
2. Воспользуйтесь соответствующей командой уровня операционной системы, чтобы отделить зеркальную копию от исходной базы данных.
3. Возобновите операции ввода/вывода в исходной базе данных:
   ```
   db2 set write resume for database
   ```
4. Подключитесь к зеркальной копии базы данных с другого компьютера.
5. Запустите экземпляр базы данных:
   ```
   db2start
   ```
6. Инициализируйте зеркальную копию базы данных как клон первичной базы данных:
   ```
   db2inidb алиас_базы_данных as snapshot
   ```

   Примечание: Эта команда произведет откат транзакций, которые выполнялись в момент отделения.

Использование отделенной зеркальной копии в качестве резервной базы данных

Так как для зеркальной (резервной) базы данных непрерывно выполняется повтор транзакций с использованием журналов, новые журналы, которые создаются в исходной базе данных, постоянно считываются из исходной системы. Чтобы воспользоваться отделенной зеркальной копией как резервной базой данных, выполните следующие действия:

1. Приостановите операции ввода/вывода в исходной базе данных:
   ```
   db2 set write suspend for database
   ```
2. Воспользуйтесь соответствующей командой уровня операционной системы, чтобы отделить зеркальную копию от исходной базы данных.
3. Возобновите операции ввода/вывода в исходной базе данных:
   ```
   db2 set write resume for database
   ```
4. Присоедините зеркальную копию базы данных к другому экземпляру.

5. Переведите зеркальную копию базы данных в состояние отложенного повтора транзакций:

   ```
   db2inidb алиас_базы_данных as standby
   ```

   Если у вас есть табличные пространства, управляемые базой данных (DMS), можно сделать резервную копию всей базы данных, чтобы уменьшить издержки от снятия резервной копии с рабочей базы данных.
6. Настройте программу обработчика пользователя, чтобы она получала из исходной базы данных последние файлы журналов.
7. Повторите для базы данных транзакции до конца журналов.
8. Продолжайте получать файлы журналов и повторять транзакции до конца журналов, пока исходная база данных будет работать.

Использование отделенной зеркальной копии в качестве резервной копии

Чтобы воспользоваться отделенной зеркальной копией как “резервной копией”, выполните следующие действия:

1. Приостановите операции ввода/вывода в исходной базе данных:

   ```
   db2 set write suspend for database
   ```
2. Воспользуйтесь соответствующей командой уровня операционной системы, чтобы отделить зеркальную копию от исходной базы данных.
3. Возобновите операции ввода/вывода в исходной базе данных:

   ```
   db2 set write resume for database
   ```
4. Воспользуйтесь командами операционной системы, чтобы скопировать данные и журналы зеркальной копии поверх первичной системы.
5. Запустите экземпляр базы данных:

   ```
   db2start
   ```
6. Инициализируйте зеркальную копию базы данных как “резервную копию”, которую можно использовать для копирования данных на отделенных дисках обратно на диски исходной системы. (Не возвращайте обратно файловую систему, которая содержит файлы журналов, так как журналы будут нужны для процесса повтора транзакций.)

   ```
   db2inidb алиас_базы_данных as mirror
   ```
7. Повторите для базы данных (в исходной системе) транзакции до конца журналов.

Глава 6. Высокая доступность в AIX

Улучшенная масштабируемость (ES) - это возможность HACMP для AIX. Эта функция обеспечивает такое же восстановление после отказа, как HACMP, и у нее та же структура событий (смотрите книгу *HACMP for AIX, V4.2.2, Enhanced Scalability Installation and Administration Guide*). Другие возможности ES:

- Большие размеры кластеров HACMP.
- 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. 213.
- 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.

© Copyright IBM Corp. 2001

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 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 спаренные узлы имели доступ к группам тома 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 11. 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. 18 на стр. 197 и рис. 19 на стр. 198 показан пример конфигурации подсистемы ввода/вывода DB2 SSA и планирование, необходимое, чтобы

обеспечить высокопроизводительный доступ к внешним дискам и отсутствие конфликтов при обращении ко всем группам томов.

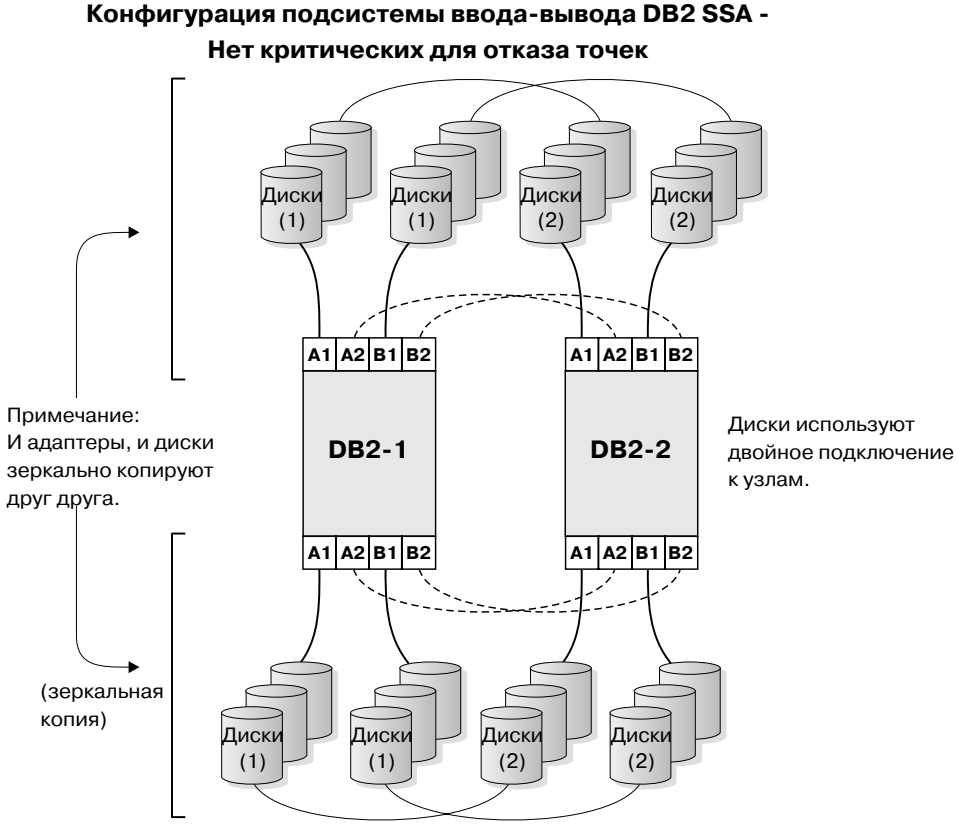


Рисунок 18. Конфигурация, устойчивая к отказу в одной любой точке

Конфигурация подсистемы ввода-вывода DB2 SSA - Группы томов и логические тома

база данных db2 testdata на файловой системе/имя экземпляра powertp

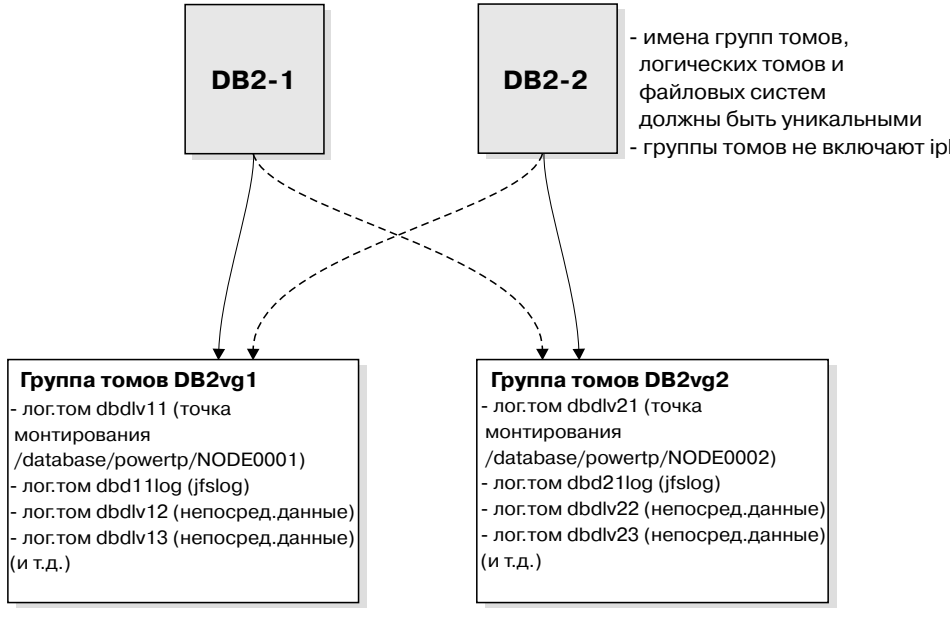


Рисунок 19. Задание группы томов и логического тома

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 отработки отказа и восстановления HACMP ES на узлах горячего резервирования или взаимной подмены. Кроме того, размеры пула буферов DB2 могут настраиваться во время отработки 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 <экземпляр> <номер_раздела> <вторичный_номер> start <коммутатор>
/usr/bin/rc.db2pe <экземпляр> <номер_раздела> <вторичный_номер> stop <коммутатор>

   где:

   <экземпляр> - имя экземпляра.
   <номер_раздела> - номер раздела базы данных.
   <вторичный ном.> в  конфигурациях взаимной подмены -
      номер "компаньона" раздела базы данных; в конфигурациях горячего
      резервирования совпадает с <ном.раздела>.
   <коммутатор> обычно пробел; пробел указывает, что
      сеть коммутатора SP используется для поля имя_хоста
      в файле db2nodes.cfg (весь трафик для DB2 направляется через коммутатор SP);
      иначе задается имя хоста для используемого 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.

Далее 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 предполагается конфигурация взаимной подмены между физическими узлами 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. 217.
- 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.

В случае конфигураций взаимной подмены DB2 HACMP (рис. 20 на стр. 205, рис. 21 на стр. 206 и рис. 22 на стр. 207):

- 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.

Конфигурация взаимной подмены DB2 HACMP с восстановлением NFS - Нормальная работа

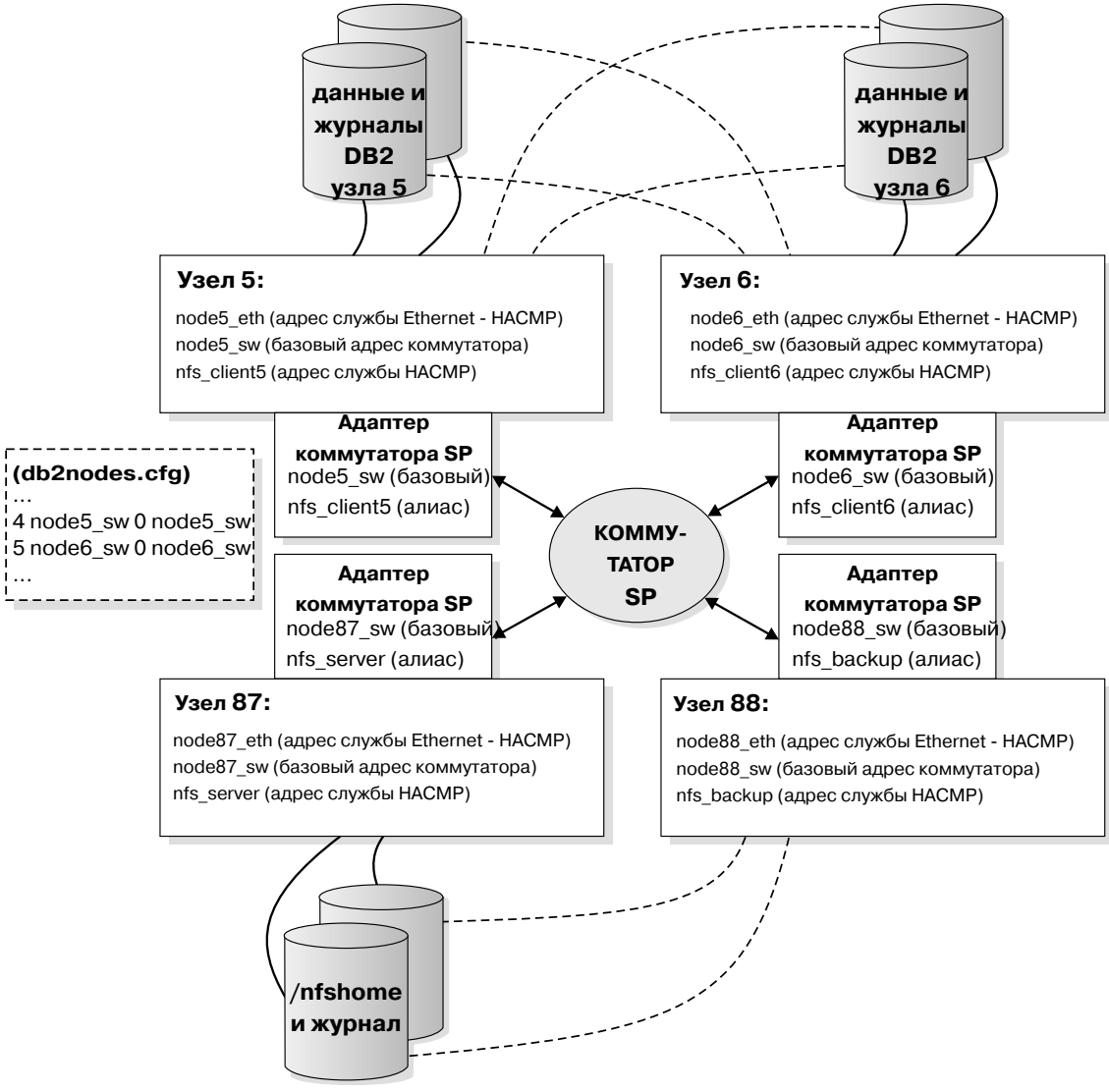


Рисунок 20. Восстановление после отказов при взаимной подмене с использованием NFS - нормальная работа

Конфигурация взаимной подмены DB2 HACMP с восстановлением NFS - Восстановление NFS

- алиас коммутатора SP nfs_server, IP-адрес и nfs, смонтированная в точке /nfshome, перемещены с узла 87 на 88.
- код arp коммутатора SP имеет возможность при таком перемещении изменить все кэши arp коммутатора.

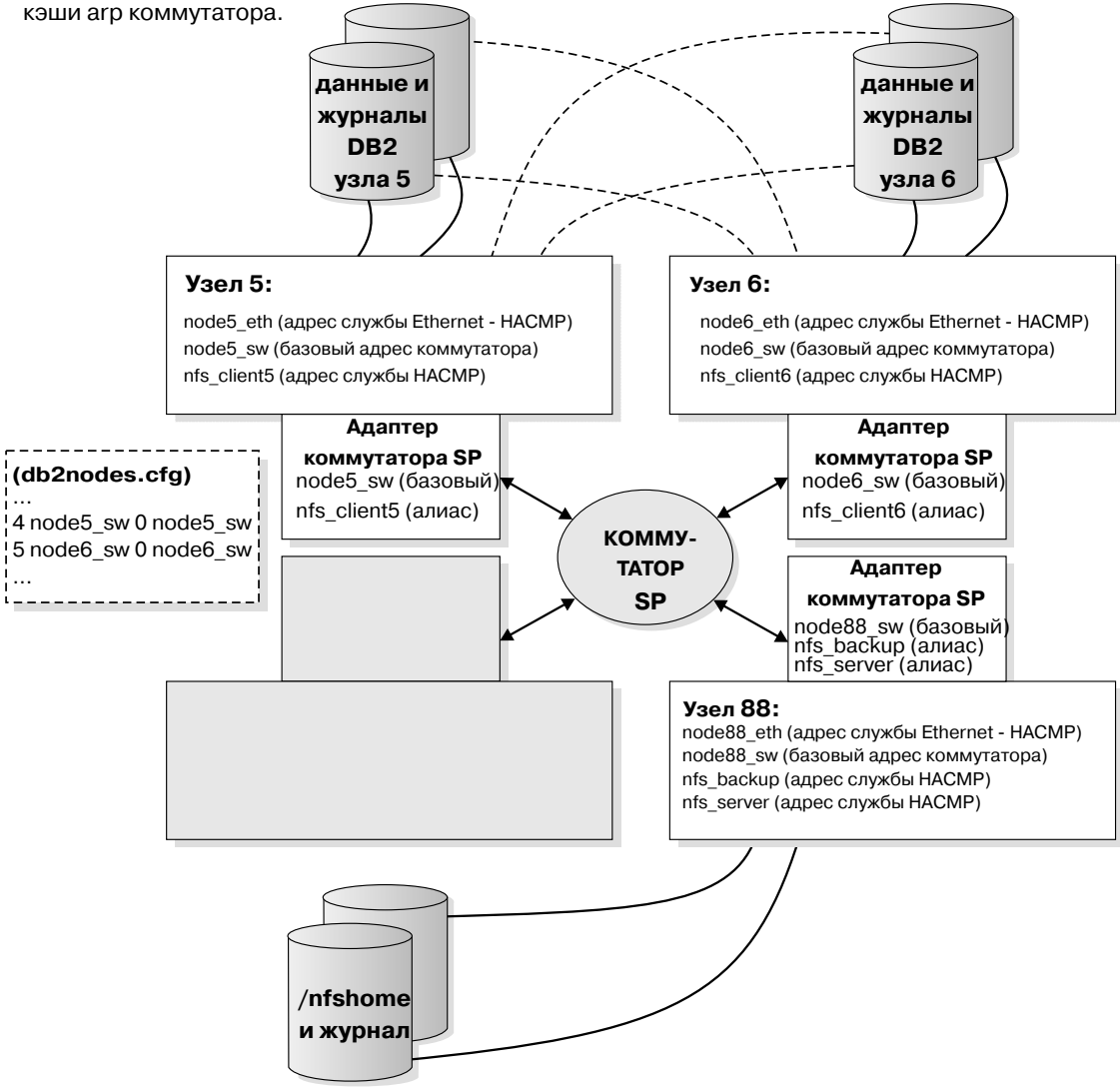


Рисунок 21. Восстановление после отказов при взаимной подмене с использованием NFS - отказ NFS

Конфигурация взаимной подмены DB2 HACMP с восстановлением NFS - отказ DB2

- подмена IP-адреса коммутатора позволяет другим серверам (типа ADSM) восстановить связь.
- На узле 5 работает 2 логических узла DB2.

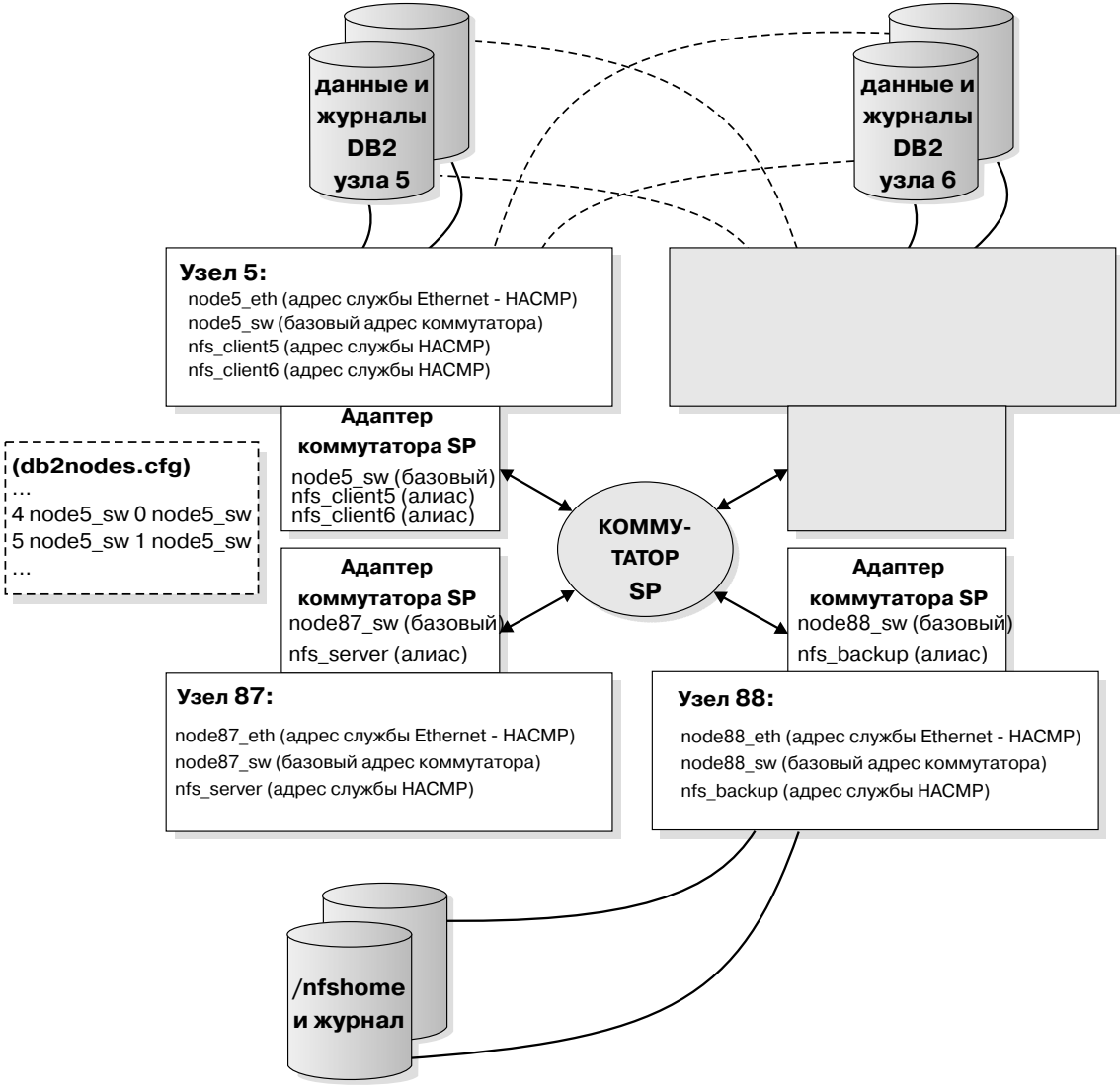


Рисунок 22. Восстановление после отказов при взаимной подмене с использованием NFS - отказ DB2

Для конфигураций горячего резервирования DB2 HACMP (рис. 23 на стр. 209 и рис. 24 на стр. 210):

- 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.

Горячее резервирование DB2 HACMP с восстановлением NFS - нормальная работа

Примечание: Узел горячего резервирования может быть резервным для нескольких узлов в зависимости от подключения дисков.

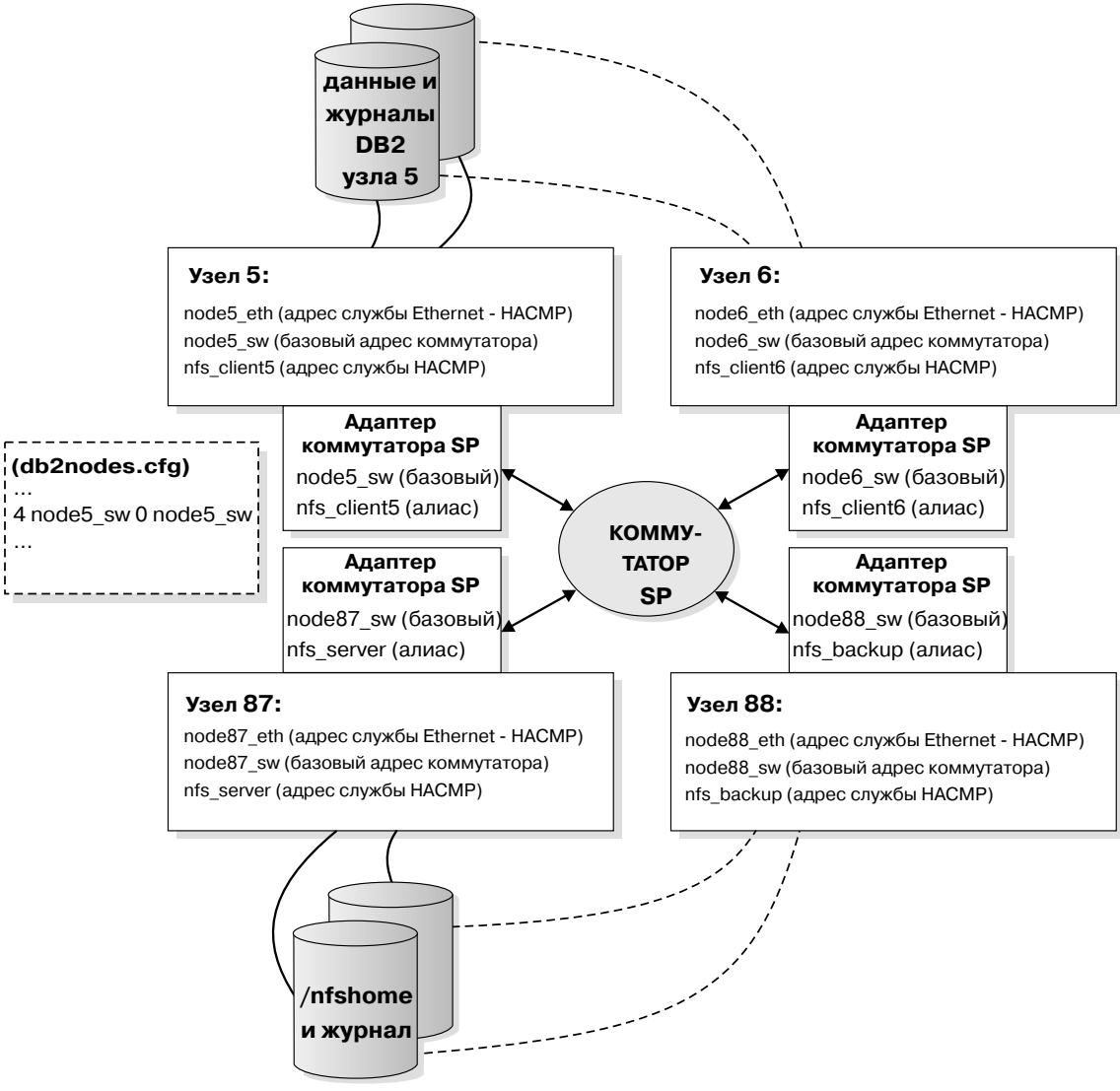


Рисунок 23. Восстановление после отказов при горячем резервировании с использованием NFS - нормальная работа

Горячее резервирование DB2 HACMP с восстановлением NFS - отказ DB2

Примечание: Узел горячего резервирования может быть резервным для нескольких узлов в зависимости от подключения дисков.

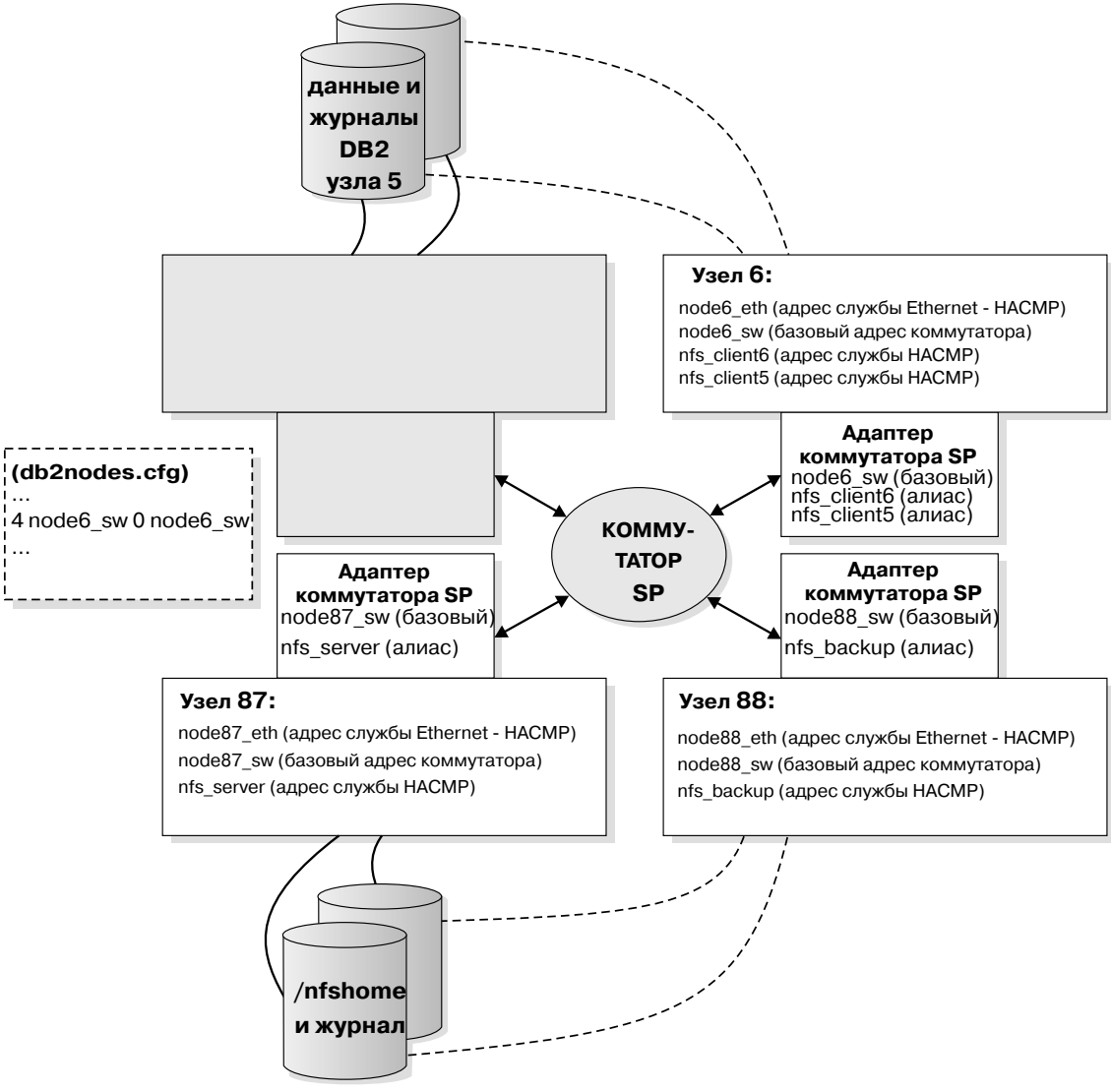


Рисунок 24. Восстановление после отказов при горячем резервировании с использованием NFS - отказ DB2

Для конфигураций взаимной подмены DB2 HACMP без NFS (рис. 25 на стр. 211 и рис. 26 на стр. 212):

- 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.

Взаимная подмена DB2 HACMP без восстановления NFS - нормальная работа

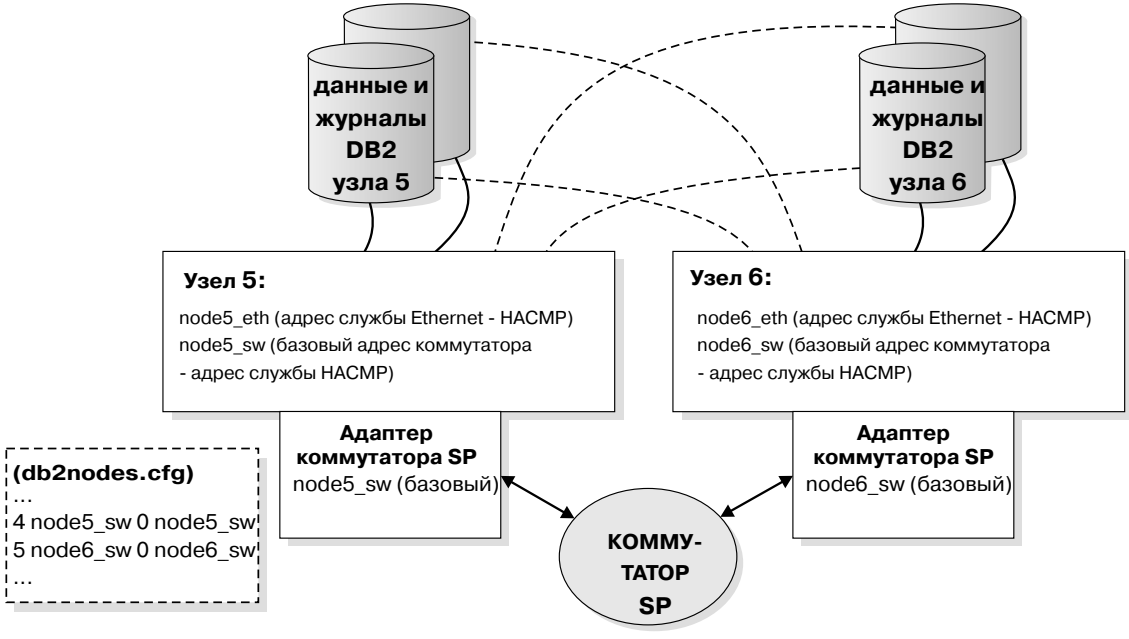


Рисунок 25. Восстановление после отказов при взаимной подмене без NFS - нормальная работа

Взаимная подмена DB2 HACMP без восстановления NFS - отказ DB2

- На узле 5 работают 2 логических узла DB2.

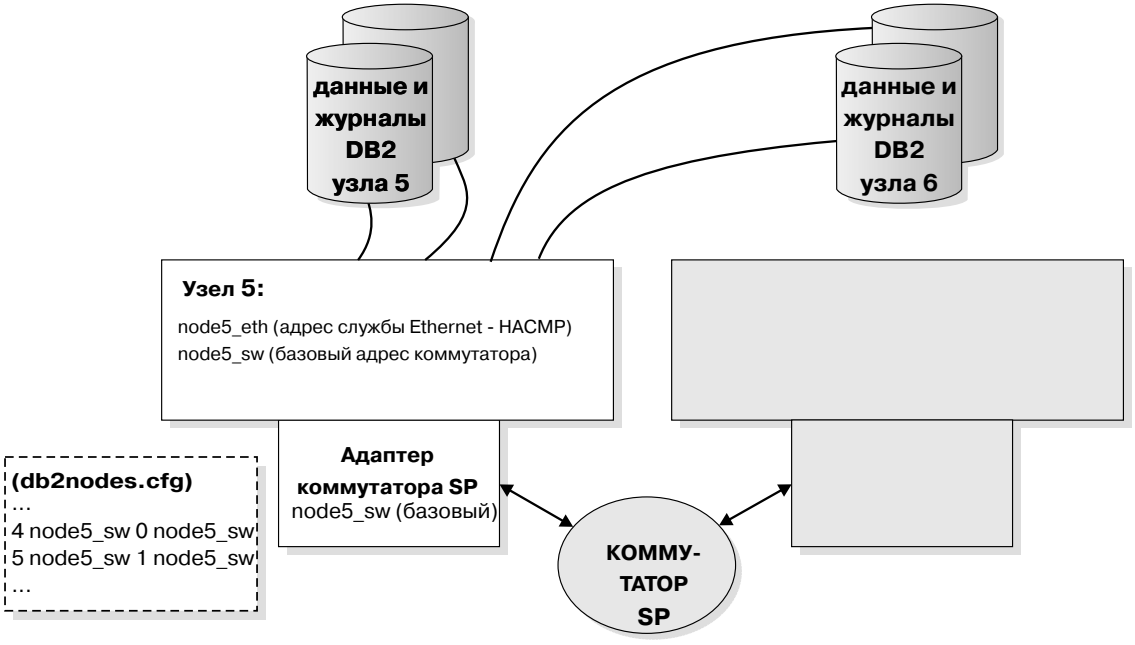


Рисунок 26. Восстановление после отказов при взаимной подмене без NFS - отказ DB2

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, переставив 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 нижний 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 каждым значением 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 из 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 pexec; 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. Проверяется, 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
```

Мониторинг кластеров HACMP

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 достаточно пространства 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 доступа 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 конфигурацию 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` проверьте, 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` проверьте, 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 подготовки конфигурации внешнего диска SSA (они проведены 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 одной любой 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 установки DB2 с внешними дисками

- Двойное подключение обеспечивает высокую доступность.

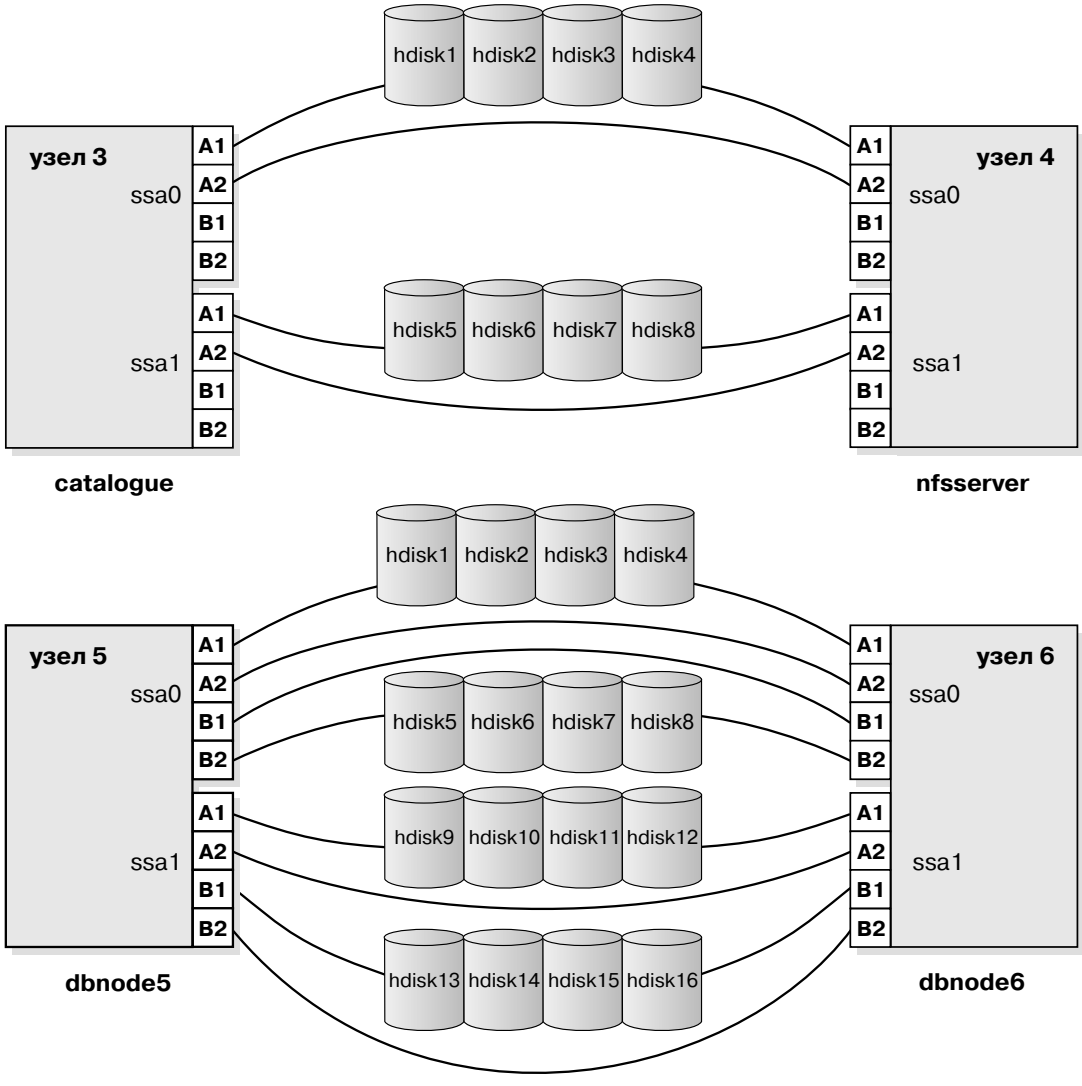


Рисунок 27. Пример настройки внешних дисков базы данных DB2 на 4 узлах

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 12. Группы томов HACMP, логические тома и файловые системы

Узел SP	Имя группы томов	Размер PP (Мб)	Имя логического тома	Число PP	Число копий	Список hdisk	Точка монтирования файловой системы	Логический том журнала файловой системы	Описание и резервная копия узла	Пользователь - владелец логического устройства /dev
3	havg3	8	hlv300	10	2	hdisk1 hdisk5	/newdata /pwq /NODE0003	hlog301	Точка монтирования 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	Пространство 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 монтирования 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	Непосредственное временное пространство Dbnode5; узел 6	pwq **

Таблица 12. Группы томов HACMP, логические тома и файловые системы (продолжение)

Узел SP	Имя группы томов	Размер PP (Мб)	Имя логического тома	Число PP	Число копий	Список hdisk	Точка монтирования файловой системы	Логический том журнала файловой системы	Описание и резервная копия узла	Пользователь - владелец логического устройства /dev
5	havg5	8	hlv502	100	2	hdisk2 hdisk10	нет	нет	Непосредственное табличное пространство Dbnode5; узел 6	pwq **
5	havg5	8	halv503	100	2	hdisk3 hdisk11	нет	нет	Непосредственное табличное пространство Dbnode5; узел 6	pwq **
5	havg5	8	halv504	100	2	hdisk3 hdisk11	нет	нет	Непосредственное табличное пространство Dbnode5; узел 6	pwq **
5	havg5	8	halv505	100	2	hdisk4 hdisk12	/dbdata5	hlog501	Системное табличное пространство Dbnode6; узел 6	root *
6	havg6	8	hlv600	10	2	hdisk5 hdisk13	/newdata/ pwq/ NODE0006	hlog601	Точка монтирования 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	Непосредственное временное пространство Dbnode6; узел 5	pwq **

Таблица 12. Группы томов HACMP, логические тома и файловые системы (продолжение)

Узел SP	Имя группы томов	Размер PP (Мб)	Имя логического тома	Число PP	Число копий	Список hdisk	Точка монтирования файловой системы	Логический том журнала файловой системы	Описание и резервная копия узла	Пользователь - владелец логического устройства /dev
6	havg6	8	hlv602	100	2	hdisk6 hdisk14	нет	нет	Табличное пространство Dbnode6; узел 5	pwq **
6	havg6	8	hlv603	100	2	hdisk7 hdisk15	нет	нет	Табличное пространство Dbnode6; узел 5	pwq **
6	havg6	8	hlv604	100	2	hdisk7 hdisk15	нет	нет	Табличное пространство Dbnode6; узел 5	pwq **
6	havg6	8	hlv605	100	2	hdisk8 hdisk16	/dbdata6	hlog601	Системное табличное пространство 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 13. Группы томов HACMP, логические тома и файловые системы - пустая таблица

Узел SP	Имя группы томов	Размер PP (Мб)	Имя логического тома	Число PP	Число копий	Список hdisk	Точка монтирования файловой системы	Логический том журнала файловой системы	Описание и резервная копия узла	Пользователь - владелец логического устройства /dev

Таблица 13. Группы томов HACMP, логические тома и файловые системы - пустая таблица (продолжение)

Узел SP	Имя группы томов	Размер PP (Мб)	Имя логического тома	Число PP	Число копий	Список hdisk	Точка монтирования файловой системы	Логический том журнала файловой системы	Описание и резервная копия узла	Пользователь - владелец логического устройства /dev

Таблица 13. Группы томов HACMP, логические тома и файловые системы - пустая таблица (продолжение)

Узел SP	Имя группы томов	Размер PP (Мб)	Имя логического тома	Число PP	Число копий	Список hdisk	Точка монтирования файловой системы	Логический том журнала файловой системы	Описание и резервная копия узла	Пользователь - владелец логического устройства /dev

Таблица 13. Группы томов HACMP, логические тома и файловые системы - пустая таблица (продолжение)

Узел SP	Имя группы томов	Размер PP (Мб)	Имя логического тома	Число PP	Число копий	Список hdisk	Точка монтирования файловой системы	Логический том журнала файловой системы	Описание и резервная копия узла	Пользователь - владелец логического устройства /dev

Таблица 14. 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

Таблица 14. Планирование сервера HACMP NFS (продолжение)

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)
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 15. Планирование сервера HACMP NFS - пустая таблица

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)

Таблица 15. Планирование сервера HACMP NFS - пустая таблица (продолжение)

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)

Таблица 15. Планирование сервера HACMP NFS - пустая таблица (продолжение)

Узел SP	Внешняя файловая система	Резерв-ный 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)

Глава 7. Высокая доступность в операционной системе Windows

Систему баз данных можно сконфигурировать так, чтобы при отказе одного из компьютера 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 “Особенности восстановления работы” на стр. 251.)

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. 251.

© Copyright IBM Corp. 2001

После завершения настройки экземпляра для поддержки восстановления после отказов полученная конфигурация будет подобна конфигурации на рис. 28.

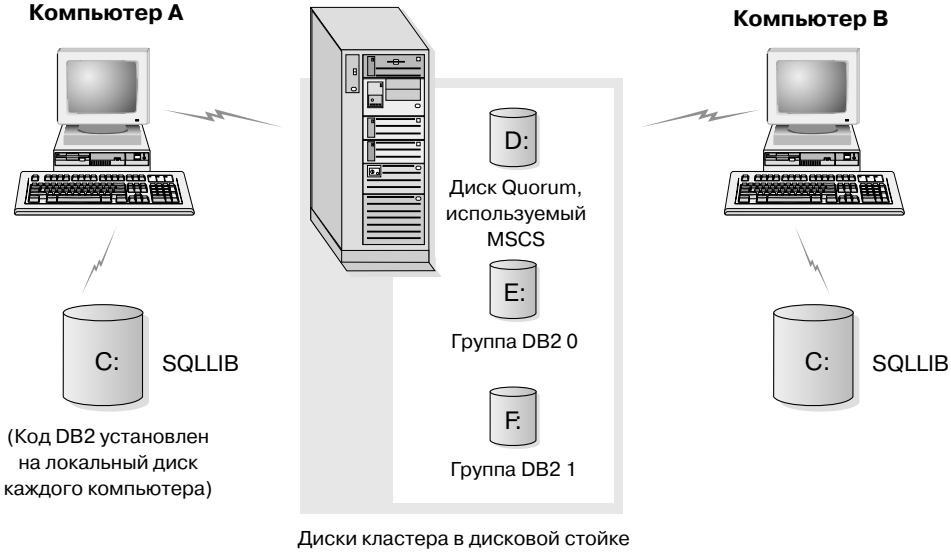


Рисунок 28. Пример конфигурации MSCS

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. 29.

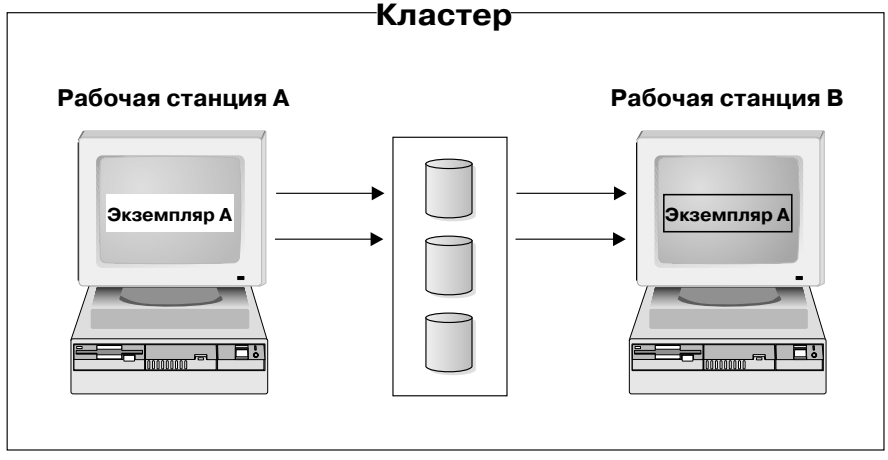


Рисунок 29. Конфигурация горячего резервирования

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. 30 на стр. 242.

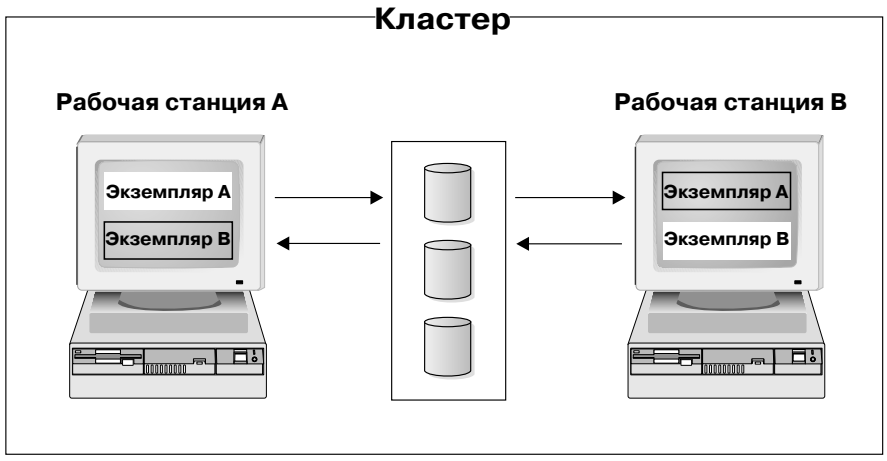


Рисунок 30. Конфигурация взаимной подмены

Использование утилиты DB2MSCS

Утилита DB2MSCS используется для создания инфраструктуры DB2 для восстановления после отказов в среде Windows NT при помощи Microsoft Cluster Service (MSCS). Эту утилиту можно использовать для включения восстановления после отказов в средах одноразделных и многораздельных баз данных.

Для успешного запуска утилиты DB2MSCS Cluster Service должна иметь возможность найти библиотеку DLL ресурсов, `db2wolf.dll`, которая находится в каталоге `%ProgramFiles%\SQLLIB\bin`. Программа установки DB2 UDB Версии 7 включает каталог `%ProgramFiles%\SQLLIB\bin` в переменную среды PATH системы. Однако при работе в среде Windows 2000 перезапуска компьютера после установки не требуется. Если вы хотите запустить утилиту DB2MSCS, надо перезапустить компьютер, чтобы изменить переменную среды PATH для Cluster Service.

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. 243.

2. Проверяет параметры, прочитанные из входного файла.
3. Регистрирует тип источника DB2.
4. Создает группу (или группы) MSCS, содержащие 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 этой 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 эту 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 многоразделных 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. 254.

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 многораздельной базы 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=...
```

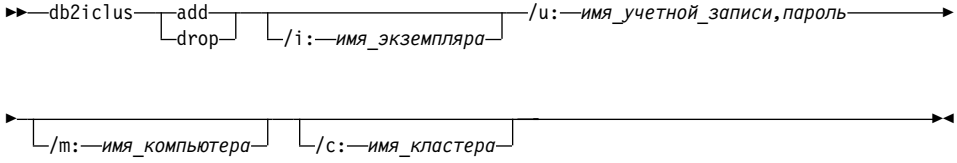
```
...
# Группа 4
GROUP_NAME=DB2 Group 4
DB2_NODE=3
IP_NAME=...

...
```

Полный пример смотрите в разделе “Пример - настройка системы многораздельной базы данных с четырьмя узлами для конфигурации взаимной подмены” на стр. 257.

Обслуживание 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 этой 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.

Особенности восстановления работы

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 базы данных можно задать букву диска, на котором должна быть создана эта база данных.

Регистрация отображения дисков базы 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:
```

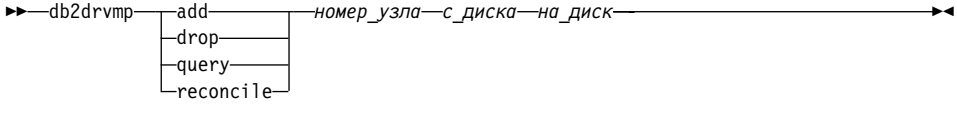
Чтобы эта команда была выполнена успешно, диск E: должен быть доступен на обоих компьютерах. В конфигурации взаимной подмены каждый сервер раздела базы данных может быть активен на разных компьютерах, а диск кластера E: доступен только на одном компьютере. В этой ситуации команда CREATE DATABASE всегда 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 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
```

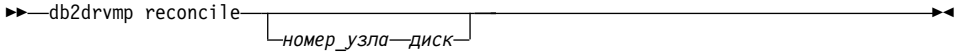
Примечание: Задание или изменение отображения дисков базы данных не вступает в силу немедленно. Чтобы активировать отображение дисков базы данных, надо при помощи инструмента администратора кластеров сделать этот ресурс DB2 пассивным и затем вновь активировать его.

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 этой 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 для этого примера - 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 базы 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 Версия 7 на компьютере `db2test1`. Установите это программное обеспечение в каталог `C:\SQLLIB` на локальном диске.
4. Установите DB2 Universal Database Enterprise Edition Версия 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 для этого примера - 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 базы 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 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

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 базы данных MSCS не перезапустит этот ресурс DB2 на резервном компьютере кластера.

Для экземпляра DB2 могут применяться три операции:

Подключено
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 все программы от системы DB2 и остановит все серверы баз 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` должен быть доступным 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 от 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 пользовательского доступа.

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 поддержки пользователей и групп в Windows NT смотрите в разделе ″Аутентификация пользователя в DB2 for Windows NT″ книги *Administration Guide: Implementation*.

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 не доступен, если не активна 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 8. Высокая доступность в операционной среде Solaris

Высокую доступность в операционной среде Solaris можно обеспечить, еслиDB2 работает с Sun Cluster 2.x (SC2.x), Sun Cluster 3.0 (SC3.0) или Veritas Cluster Server (VCS). Информацию о Sun Cluster 3.0 смотрите в ″белой книге″ под заголовком “DB2 and High Availability on Sun Cluster 3.0” на Web-сайте “DB2 UDB and DB2 Connect Online Support” (http://www.ibm.com/software/data/pubs/papers/). Информацию о VERITAS Cluster Server смотрите в ″белой книге″ под заголовком “DB2 and High Availability on VERITAS Cluster Server”, которая также доступна на Web-сайте “DB2 UDB and DB2 Connect Online Support”.

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. 31).

Рисунок 31. 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

© Copyright IBM Corp. 2001

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. 32.
Интерфейсы собственной сети служат для обмена сообщениями

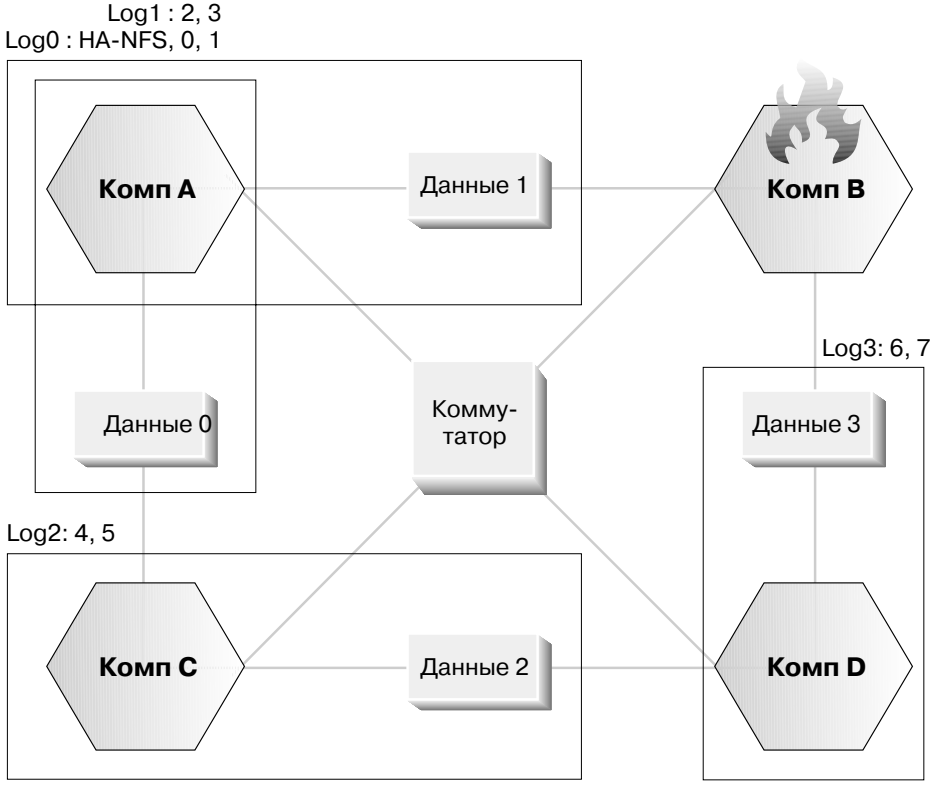


Рисунок 32. Восстановление после сбоев

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	ввод/вывод
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 Мбайт - 64 Гбайт	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 бесплатно 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
#device to mount             device  to fsck               mount
#                                                          point

/dev/vx/dsk/data0/data1-stat /dev/vx/rdsk/data0/data1-stat /log0
/dev/vx/dsk/data0/vol01      /dev/vx/rdsk/data0/vol01      /log0/home
/dev/vx/dsk/data0/vol02      /dev/vx/rdsk/data0/vol02      /log0/data

scadmin@thrash(221)# cat vfstab.log1
#device to mount             device  to fsck               mount
#                                                          point
/dev/vx/dsk/data1/data1-stat /dev/vx/rdsk/data1/data1-stat /log1
/dev/vx/dsk/data1/vol01      /dev/vx/rdsk/data1/vol01      /log1/home
/dev/vx/dsk/data1/vol02      /dev/vx/rdsk/data1/vol02      /log1/data
/dev/vx/dsk/data1/vol03      /dev/vx/rdsk/data1/vol03      /log1/data1



FS   fsck mount   options
```

```
type pass at boot

ufs  2    no      -
ufs  2    no      -
ufs  2    no      -


FS   fsck mount   options
type pass at boot

ufs  2    no      -
ufs  2    no      -
ufs  2    no      -
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 в точности 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 базы 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 они были 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. При ошибке **fsck** файловая система не будет смонтирована и в журнал записывается 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
total 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. 293).

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. 33 на стр. 290). 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.

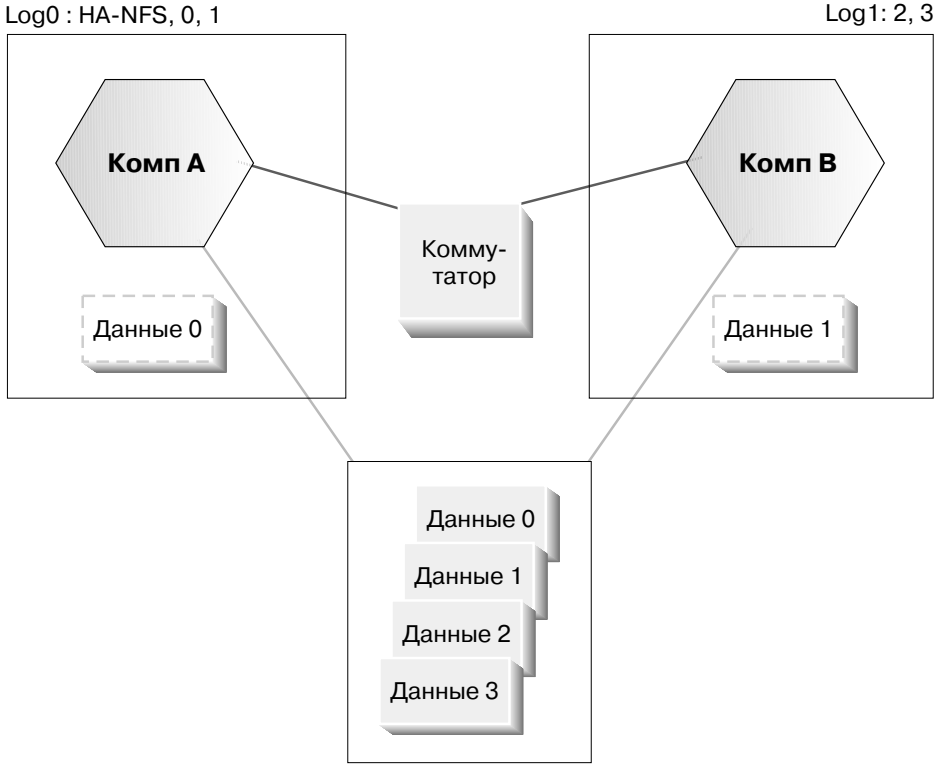


Рисунок 33. По одному логическому хосту на каждом компьютере

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 базы данных

Параметры конфигурации менеджера базы данных можно изменить после отказа и восстановления и до запуска DB2 с помощью сценария `pre_db2start` (смотрите раздел “Пользовательские сценарии” на стр. 295). 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 отказа

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 отказа и других формах восстановления базы данных смотрите в разделе “Глава 1. Разработка правильной стратегии резервного копирования и восстановления” на стр. 3.

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.

Предварительные требования для DB2 Connect в Sun Cluster 2.2

DB2 Connect поддерживается в Sun Cluster 2.2, если:

- Для связи с хостом используется протокол TCP/IP (не SNA)
- Двухфазное принятие не используется. Это ограничение ослабляется, если пользователь сконфигурирует журнал SPM на совместно используемом диске (это можно сделать при помощи параметра конфигурации менеджера баз данных *spm_log_path*) и если подменяющий компьютер имеет такую же конфигурацию TCP/IP (те же имя хоста, IP-адрес и т.д.).

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 управления экземпляры, 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 доступны методы управления, 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 конфигурацией 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 и логическими хостами 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	Никаких действий
STOP_NET	Никаких действий	Остановить экземпляр DB2 или разделы базы данных
FM_START	Запустить монитор отказов для экземпляра	Никаких действий
FM_STOP	Никаких действий	Остановить мониторинг отказов для экземпляра

Методы управления, выполняющие действия по запуску, применяются только к размещенным в данное время логическими хостами, а методы управления, которые выполняют действия по остановке, применяются только к неразмещенным в данное время логическими хостами.

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 базы 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 рассматриваться до того, как кластер начнет работать в производственном режиме. Например, если на каком-либо компьютеру 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. 303).

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. 34 на стр. 299.

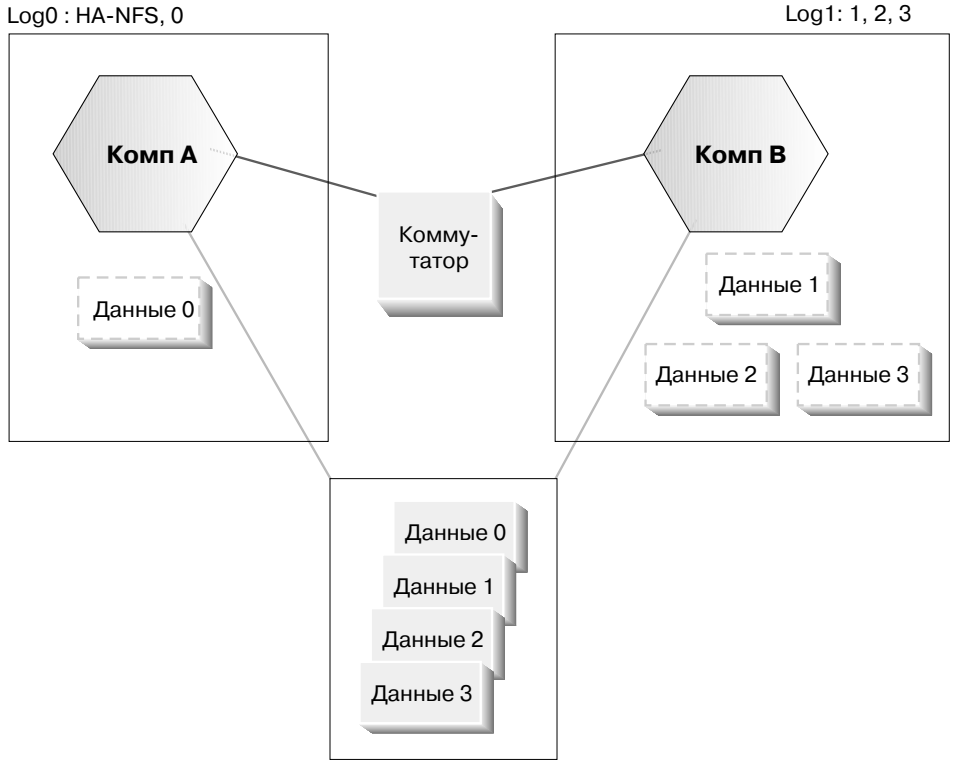


Рисунок 34. Кластер из двух компьютеров с четырьмя разделами базы данных

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 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 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 переопределяется при помощи переменной SCRIPT_USER, в которой можно задавать ID пользователя, который может 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.

Параметры

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. 309.

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. 287.

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 журнализацией может резко сократить это время. Поскольку в среде высокой

доступности как правило, используются файловые системы с журнализацией, 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 базы 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 16. Устранение неисправностей высокой доступности в Sun Cluster 2.2

Симптом	Вероятная причина	Действие
Невозможно 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]

Таблица 16. Устранение неисправностей высокой доступности в Sun Cluster 2.2 (продолжение)

Симптом	Вероятная причина	Действие
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.
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 16. 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.
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.

Таблица 16. Устранение неисправностей высокой доступности в Sun Cluster 2.2 (продолжение)

Симптом	Вероятная причина	Действие
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. 303).

Таблица 16. 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 16. 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 16. 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 3. Приложения

© Copyright IBM Corp. 2001

Приложение A. Как читать синтаксические диаграммы

Синтаксическая диаграмма показывает, как надо задать команда, чтобы операционная система могла правильно ее интерпретировать.

Синтаксическую диаграмму читают слева направо и сверху вниз, следуя вдоль горизонтальной линии (главного пути). Если строка заканчивается стрелкой, синтаксис команды продолжается на следующей строке, которая начинается с того же символа стрелки. Вертикальная черта обозначает конец синтаксиса команды.

При вводе команд из синтаксической диаграммы важно не пропускать знаки препинания, например,кавычки и знаки равенства.

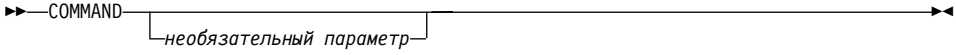
Параметры делятся на ключевые слова и переменные:

- Ключевые слова являются константами; она показаны в верхнем регистре, однако в командной строке ключевые слова можно вводить с использованием символов верхнего, нижнего или обоих регистров. Пример ключевого слова - имя команды.
- Переменные - это именами или величинами, задаваемыми пользователем; они показаны в нижнем регистре; однако, если только на этот счет нет конкретных требований, в командной строке переменные могут вводиться с использованием символов верхнего, нижнего или обоих регистров. Пример переменной - имя файла.

Параметр может быть сочетанием ключевого слова и переменной.

Обязательные параметры показаны на главном пути:

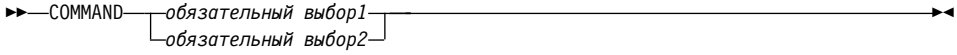
Необязательные параметры приведены под главным путем:



© Copyright IBM Corp. 2001

Значения параметров по умолчанию приведены над путем:

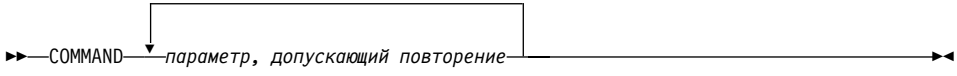
Вертикальный ряд параметров, где первый параметр показан на главном пути, указывает на то, что должен быть выбран один из параметров:



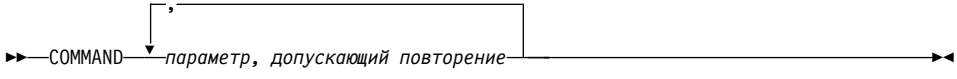
Вертикальный ряд параметров, где первый параметр показан под главным путем, указывает на то, что можно выбрать один из параметров:

Стрелка влево над путем означает, что элементы могут повторяться в соответствии с нижеприведенными соглашениями:

- Если стрелка непрерывна, элемент может повторяться в списке, где элементы разделены пробелами:

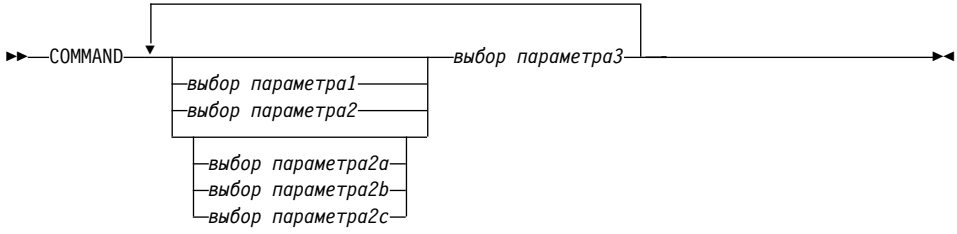


- Если стрелка содержит запятую, элемент может повторяться в списке, где элементы разделены запятыми:

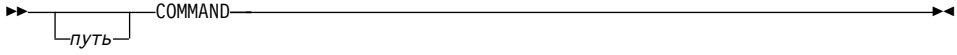


Элементы из вертикальных рядов могут повторяться в соответствии с описанными выше соглашениями об обязательных и необязательных параметрах.

Некоторые синтаксические диаграммы содержат вертикальные ряды параметров, которые, в свою очередь, содержат другие вертикальные ряды параметров. Элементы из вертикальных рядов могут повторяться только в соответствии с описанными выше соглашениями. Это означает, что если над внутренним рядом нет стрелки повтора, а над внешним она есть, только один параметр из внутреннего ряда может быть выбран и комбинирован с каким-либо параметром из внешнего ряда, и уже эту комбинацию можно повторить. Например, следующая диаграмма показывает, что можно комбинировать параметр *выбор2a* с параметром *выбор2*, а затем повторить эту комбинацию снова (*выбор2* плюс *выбор2a*):



Перед некоторыми командами используется необязательный параметр пути:



Если этот параметр не задан, система пытается выполнить команду из текущего каталога. Если система не находит команду, она продолжает поиск команды во всех каталогах, перечисленных в `.profile`.

У некоторых команд есть функционально эквивалентные синтаксические варианты:

Как читать синтаксические диаграммы

Приложение B. Предупреждения, сообщения об ошибках и завершении

Сообщения, генерируемые различными утилитами, резервного копирования и восстановления включены в общий список сообщений SQL. Такие сообщения генерируются менеджером баз данных при обнаружении ошибочных ситуаций или ситуаций предупреждений. У каждого сообщения есть свой идентификатор, состоящий из префикса (SQL) и четырех- или пятизначного номера сообщения. Сообщения делятся на три типа: извещения, предупреждения и критические сообщения. Сообщения, идентификаторы которых оканчиваются буквой `N` - это сообщения об ошибках. Сообщения, идентификаторы которых оканчиваются буквой `W` - предупреждения или информационные сообщения. Сообщения, идентификаторы которых оканчиваются буквой `C`, извещают о критических ошибках системы.

Номер сообщения используется также как код (*SQLCODE*).Этот код передается прикладной программе в виде положительного или отрицательного числа в зависимости от типа сообщения (N, W или C). Сообщениям типа N и C соответствуют отрицательные числа, а сообщениям типа W - положительные. Система DB2 возвращает прикладной программе код SQLCODE, а программа может по этому коду восстановить соответствующее ему сообщение. Система DB2 также возвращает *SQLSTATE* - значение условий, описывающих ошибки, которые могут произойти при выполнении оператора SQL. Некоторые значения SQLCODE связаны с определенными значениями SQLSTATE.

Подробную информацию обо всех сообщениях DB2 смотрите в книге *Справочник по сообщениям*. Вы можете использовать информацию этой книги для нахождения причин ошибки и их устранения путем выполнение соответствующих действий. Эта информация помогает также понять, где генерируются и куда записываются сообщения.

Сообщения SQL и тексты сообщений, соответствующие значениям SQLSTATE, можно посмотреть также из командной строки операционной системы. Чтобы получить справку для сообщения об ошибке, введите в командной строке операционной системы:

```
db2 ? SQLnnnnn
```

где *nnnnn* - номер сообщения. В системах на основе UNIX рекомендуется использовать двойные кавычки; это поможет избежать проблем в случае, если в каталоге есть файлы с именем из одного символа:

```
db2 "? SQLnnnnn"
```

© Copyright IBM Corp. 2001

Идентификатор сообщения, принимаемый как параметр команды **db2**, регистронезависим, код серьезности сообщения не обязателен. Тем самым, следующие команды дают один и тот же результат:

```
db2 ? SQL0000N
db2 ? sql0000
db2 ? SQL0000n
```

Если выводимый текст не помещается на вашем экране, используйте следующую команду (в системах на основе UNIX и в прочих системах, где поддерживается команда ″more″):

```
db2 ? SQLnnnnn | more
```

Можно также перенаправить вывод в файл, а затем просмотреть этот файл.

Справку можно также вызвать из режима интерактивного ввода. Чтобы войте в этот режим, введите в командной строке операционной системы:

```
db2
```

Чтобы получить справку по сообщению DB2 в этом режиме, введите в ответ на приглашение (db2 =>):

```
? SQLnnnnn
```

Текст сообщений, связанных с кодами SQLSTATE, можно получить так:

```
db2 ? nnnnn
 или
db2 ? nn
```

где *nnnnn* - пятизначный SQLSTATE (алфавитно-цифровой), а *nn* - двузначный код класса SQLSTATE (первые две цифры значения SQLSTATE).

Приложение C. Дополнительные команды DB2

© Copyright IBM Corp. 2001

db2adutl - Работа с архивными образами TSM

Позволяет пользователям запрашивать, извлекать, проверять и удалять образы резервных копий, файлов журнала и копий загрузки, сохраненные при помощи Tivoli Storage Manager (прежнее название - ADSM).

В системах на основе UNIX эта утилита находится в каталоге INSTHOME/sqllib/misc. В операционных системах Windows и OS/2 она находится в каталоге \sqllib\misc.

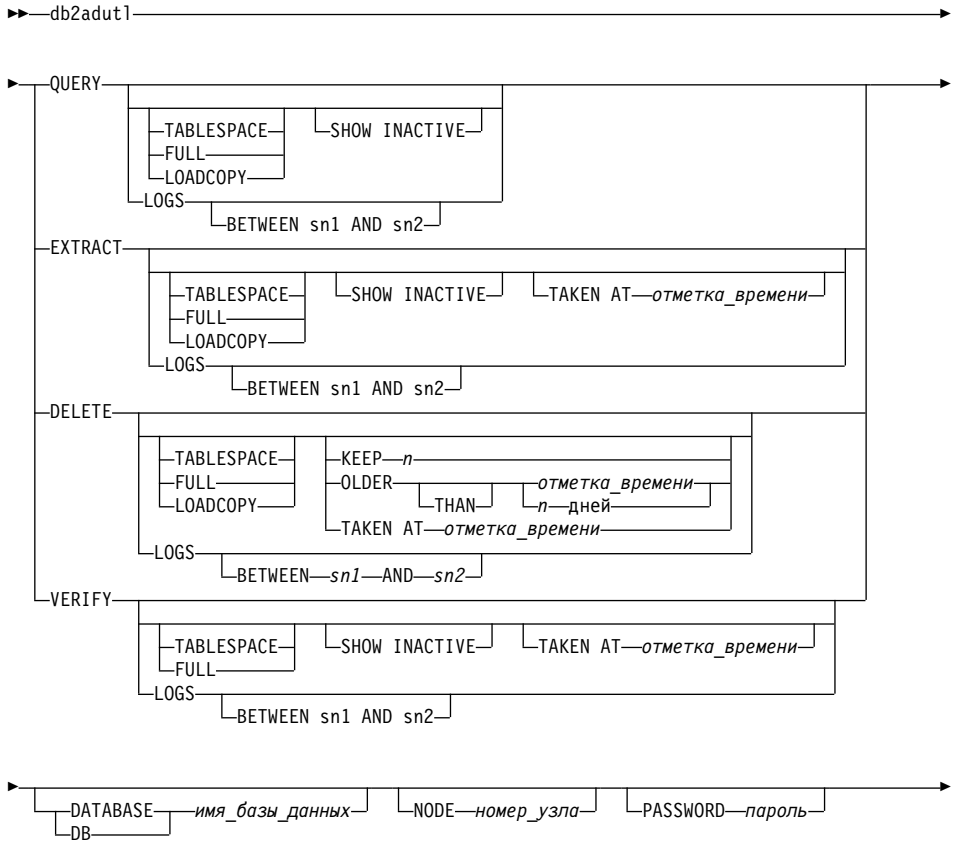
Полномочия

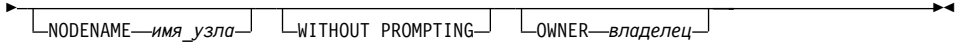
Нет

Необходимое соединение

Нет

Синтаксис команды





Параметры команды

QUERY
Запрашивает у сервера TSM объекты DB2.

EXTRACT
Копирует объекты DB2 с сервера TSM в текущий каталог локального компьютера.

DELETE
Либо деактивирует объекты резервных копий, либо удаляет архивы журналов с сервера TSM.

VERIFY
Выполняет проверку согласованности для резервной копии, находящейся на сервере.

Примечание: Этот параметр вызывает передачу по сети всего образа резервной копии.

TABLESPACE
Включает только образы резервных копий табличных пространств.

FULL Включает только полные образы резервных копий баз данных.

LOADCOPY
Включает только образы копий загрузки.

LOGS Включает только образы архивов журнала

BETWEEN *sn1* **AND** *sn2*
Задает использование файлов журнала с порядковыми номерами от sn1 до sn2.

SHOW INACTIVE
Включает объекты резервных копий, которые были деактивированы.

TAKEN AT *отметка_времени*
Задает образ резервной копии, исходя из ее отметки времени.

KEEP *n*
Деактивирует все объекты указанного типа, кроме *n* самых последних по отметкам времени.

OLDER THAN *отметка_времени* **или** *n* **дней**
Указывает, что будут деактивированы объекты с отметкой времени раньше, чем *отметка_времени* или старше *n* дней.

DATABASE *имя_базы_данных*
Применять операцию только к тем объектам, которые связаны с указанным именем базы данных.

NODE *номер_узла*
Применять операцию только к тем объектам, которые созданы узлом с указанным номером.

PASSWORD *пароль*
Указывает пароль клиента TSM для этого узла, если он требуется. Если указана конкретная база данных, а пароль не предоставлен, TSM передается значение, указанное для параметра конфигурации базы данных *tsm_password*; в противном случае пароль не используется.

NODENAME *номер_узла*
Учитывает только те образы, которые связаны с заданным именем узла TSM.

WITHOUT PROMPTING
Перед удалением объектов не запрашивается подтверждение пользователя.

OWNER *владелец*
Применять операцию только к тем объектам, которые созданы указанным владельцем.

Примеры

Ниже приводится пример вывода команды: `db2 backup database rawsampl use tsm`

```
Резервное копирование завершилось успешно. Отметка времени копии : 19970929130942

  db2adutl query

  Query for database RAWSAMPL

  Retrieving full database backup information.
     full database backup image: 1, Time: 19970929130942,
                                    Oldest log: S0000053.LOG, Sessions used: 1
     full database backup image: 2, Time: 19970929142241,
                                    Oldest log: S0000054.LOG, Sessions used: 1

  Retrieving table space backup information.
     table space backup image: 1, Time: 19970929094003,
                                  Oldest log: S0000051.LOG, Sessions used: 1
     table space backup image: 2, Time: 19970929093043,
                                  Oldest log: S0000050.LOG, Sessions used: 1
     table space backup image: 3, Time: 19970929105905,
                                  Oldest log: S0000052.LOG, Sessions used: 1

  Retrieving log archive information.
     Log file: S0000050.LOG
     Log file: S0000051.LOG
```

```
   Log file: S0000052.LOG
   Log file: S0000053.LOG
   Log file: S0000054.LOG
   Log file: S0000055.LOG
```

Ниже приводится пример вывода команды: `db2adutl delete full rawsampl use tsm`

```
  Query for database RAWSAMPL

  Retrieving full database backup information.  Please wait.

    full database backup image: RAWSAMPL.0.db26000.0.19970929130942.001

    Do you want to deactivate this backup image (Y/N)? y

      Are you sure (Y/N)? y

  db2adutl query

  Query for database RAWSAMPL

  Retrieving full database backup information.
     full database backup image: 2, Time: 19950929142241,
                               Oldest log: S0000054.LOG, Sessions used: 1

  Retrieving table space backup information.
     table space backup image: 1, Time: 19950929094003,
                             Oldest log: S0000051.LOG, Sessions used: 1
     table space backup image: 2, Time: 19950929093043,
                             Oldest log: S0000050.LOG, Sessions used: 1
     table space backup image: 3, Time: 19950929105905,
                             Oldest log: S0000052.LOG, Sessions used: 1

  Retrieving log archive information.
     Log file: S0000050.LOG
     Log file: S0000051.LOG
     Log file: S0000052.LOG
     Log file: S0000053.LOG
     Log file: S0000054.LOG
     Log file: S0000055.LOG
```

db2ckbkp - Проверка резервной копии

Эту утилиту можно использовать для проверки целостности образа резервной копии и для определения возможности восстановления этого образа. Ее можно также использовать для вывода метаданных, хранящихся в заголовке резервной копии.

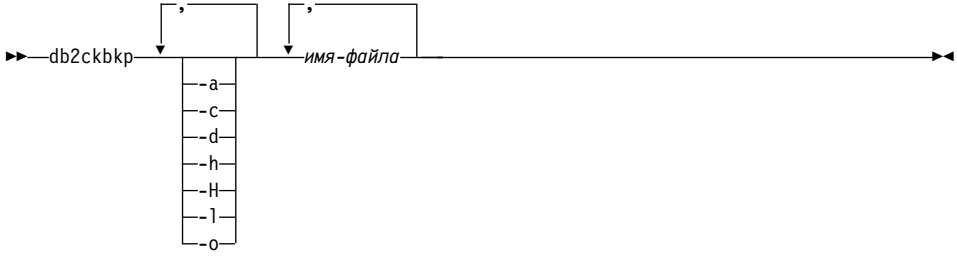
Полномочия

Доступ к этой утилите может получить любой пользователь, но чтобы запустить ее для конкретной резервной копии, надо иметь разрешение на ее чтение.

Необходимое соединение

Нет

Синтаксис команды



Параметры команды

-a Выводит всю доступную информацию.

-c Выводит результаты проверки контрольных битов и контрольных сумм.

-d Выводит информацию из заголовков страниц данных табличного пространства DMS.

-h Выводит информацию заголовка носителя, включая имя или путь образа, ожидаемого утилитой восстановления.

-H Выводит только информацию заголовка носителя.

Примечания:

1. Эта опция не проверяет образ. Если эта опция не задана, выполняется проверка всего образа.
2. Эта опция недопустима в сочетании с любой другой опцией.

-l Выводит данные заголовков файлов журнала.

-o Выводит подробную информацию из заголовков объектов.

имя-файла

Имя файла образа резервной копии. За один вызов можно проверить один или несколько файлов.

Примечания:

1. Если в полной резервной копии содержится ряд объектов, проверка будет успешной только в случае, если **db2ckbkp** используется для проверки всех объектов в одно и то же время.
2. При проверке нескольких частей образа первый объект образа резервной копии (.001) должен быть задан первым.

Примеры

```
db2ckbkp SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.*
[1] Buffers processed:  ##
[2] Buffers processed:  ##
[3] Buffers processed:  ##
Image Verification Complete - successful.

db2ckbkp -h SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001


=====================
MEDIA HEADER REACHED:
=====================
        Server Database Name           -- SAMPLE2
        Server Database Alias          -- SAMPLE2
        Client Database Alias          -- SAMPLE2
        Timestamp                      -- 19990818122909
        Node                           -- 0
        Instance                       -- krodger
        Sequence Number                -- 1
        Release ID                     -- 900
        Database Seed                  -- 65E0B395
        DB Comment's Codepage (Volume) -- 0
        DB Comment (Volume)            --
        DB Comment's Codepage (System) -- 0
        DB Comment (System)            --
        Authentication Value           -- 255
        Backup Mode                    -- 0
        Backup Type                    -- 0
        Backup Gran.                   -- 0
        Status Flags                   -- 11
        System Cats inc                -- 1
        Catalog Node Number            -- 0
        DB Codeset                     -- ISO8859-1
        DB Territory                   --
        Backup Buffer Size             -- 4194304
        Number of Sessions             -- 1
        Platform                       -- 0

The proper image file name would be:
SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001
```

```
[1] Buffers processed:  ####
Image Verification Complete - successful.
```

Замечания по использованию

Если образ резервной копии создавался с использованием нескольких сеансов, **db2ckbkp** может проверить все файлы одновременно. Пользователь должен задать сеанс с последовательным номером `001` первым заданным файлом.

Эта утилита может также проверять образы резервных копий на ленте (за исключением образов, созданных с переменным размером блоков). Для этого лента подготавливается так же, как и для операции восстановления, а затем вызывается данная утилита и задается имя ленточного устройства. Например, в системах на основе UNIX:

```
db2ckbkp -h /dev/rmt0
```

а в Windows NT:

```
db2ckbkp -d \\.\tape1
```

Если образ резервной копии находится в TSM, смотрите “db2adutl - Работа с архивными образами TSM” на стр. 326.

db2ckrst - Проверка последовательности резервных копий для инкрементного восстановления

Просматривает хронологию базы данных и создает список отметок времени для резервных копий, необходимых для инкрементного восстановления. Создает также упрощенный синтаксис восстановления для инкрементного восстановления вручную.

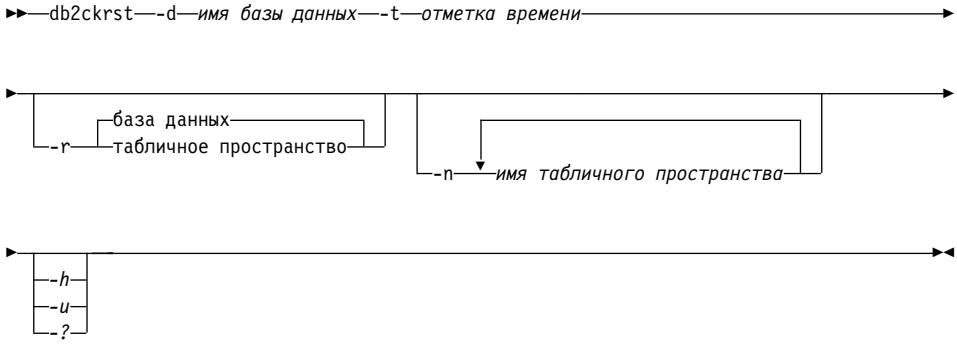
Полномочия

Нет

Необходимое соединение

Нет

Синтаксис команды



Параметры команды

-d имя-базы-данных имя-файла
Задает алиас для восстанавливаемой базы данных.

-t отметка-времени
Задает отметку времени для резервной копии, подлежащей инкрементному восстановлению.

-r Задает тип восстановления, которое нужно выполнить. По умолчанию восстанавливается база данных.

Примечание: Если выбрано восстановление табличного пространства, но не указано имен табличных пространств, утилита использует для восстановления имена табличных пространств, перечисленные в записи хронологии заданной резервной копии.

-n имя-табличного-пространства

Задает имя одного или нескольких табличных пространств для восстановления.

Примечание: Если выбрано восстановление базы данных, но задан список имен табличных пространств, утилита будет выполнять восстановление указанных табличных пространств.

-h/-u/-?

Выводит справку. Если указана эта опция, прочие опции игнорируются и выводится только информация справки.

Примеры

```
db2ckrst -d mr -t 20001015193455 -r database
db2ckrst -d mr -t 20001015193455 -r tablespace
db2ckrst -d mr -t 20001015193455 -r tablespace -n tbsp1 tbsp2

> db2 backup db mr

Backup successful. The timestamp for this backup image is : 20001016001426

> db2 backup db mr incremental

Backup successful. The timestamp for this backup image is : 20001016001445

> db2ckrst -d mr -t 20001016001445

Suggested restore order of images using timestamp 20001016001445 for database mr.
==================================================================
  db2 restore db mr incremental taken at 20001016001445
  db2 restore db mr incremental taken at 20001016001426
  db2 restore db mr incremental taken at 20001016001445
==================================================================

> db2ckrst -d mr -t 20001016001445 -r tablespace -n userspace1
Suggested restore order of images using timestamp 20001016001445 for database mr.
==================================================================
  db2 restore db mr tablespace ( USERSPACE1 ) incremental taken at 20001016001445
  db2 restore db mr tablespace ( USERSPACE1 ) incremental taken at 20001016001426
  db2 restore db mr tablespace ( USERSPACE1 ) incremental taken at 20001016001445
==================================================================
```

Замечания по использованию

Для использования этой утилиты необходимо наличие хронологии базы данных. Если хронологии базы данных нет, следует перед использованием утилиты задать опцию HISTORY FILE в команде RESTORE DATABASE.

Если используется опция FORCE команды PRUNE HISTORY, есть вероятность удаления записей, требуемых для восстановления из самой поздней полной

резервной копии базы данных. По умолчанию команда PRUNE HISTORY не удаляет нужные записи. Не рекомендуется использовать опцию FORCE команды PRUNE HISTORY.

Мы советуем хранить полную запись операций восстановления и использовать эту утилиту как руководство.

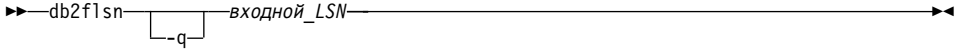
db2flsn - Найти последовательный номера журнала

Возвращает имя файла, содержащего запись журнала, идентифицированную заданным LSN (log sequence number - последовательный номер журнала).

Полномочия

Нет

Синтаксис команды



Параметры команды

-q Задает, что будет выводиться только имя файла журнала. Не будут выводиться ни сообщения об ошибках, ни предупреждающие сообщения, а состояние можно будет определить только по коду возврата. Допустимые коды ошибок:

- -100 Недопустимый ввод
- -101 Не удалось открыть файл LFH
- -102 Ошибка чтения файла LFH
- -103 Недопустимый файл LFH
- -104 Невосстановимая база данных
- -105 Слишком большой LSN
- -500 Логическая ошибка.

Другие допустимые коды возврата:

- 0 Успешное выполнение
- 99 Предупреждение: результат выдается на основе размера последнего известного файла журнала.

входной_LSN

12-байтная строка, представляющая внутреннее (6-байтное) шестнадцатеричное значение с ведущими нулями.

Примеры

```
db2flsn 000000BF0030
   Given LSN is contained in log file S0000002.LOG

db2flsn -q 000000BF0030
   S0000002.LOG

db2flsn 000000BE0030
   Warning: the result is based on the last known log file size.
   The last known log file size is 23 4K pages starting from log extent 2.
```

```
   Given LSN is contained in log file S0000001.LOG

db2flsn -q 000000BE0030
   S0000001.LOG
```

Замечания по использованию

Файл управления заголовками журнала `sqlogctl.lfh` должен находиться в текущем каталоге. Поскольку этот файл находится в каталоге базы данных, можно запустить этот инструмент оттуда, или же скопировать файл управления в каталог, из которого будет запускаться этот инструмент.

Этот инструмент используется параметр *logfilsiz* конфигурации базы данных. DB2 записывает три самых последних значения для этого параметра и первый файл журнала, который создается с каждым значением параметра *logfilsiz*; это обеспечивает правильную работу инструмента, когда значение параметра *logfilsiz* изменяется. Если заданный LSN предшествует наиболее раннему записанному значению параметра *logfilsiz*, инструмент использует именно это значение и возвращает предупреждающее сообщение. Данный инструмент может использоваться с менеджерами баз данных, предшествовавшими UDB Версии 5.2; в этом случае предупреждающее сообщение возвращается даже с правильным результатом (полученным, если значение параметра *logfilsiz* остается неизменным).

Этот инструмент используется только с восстановимыми базами данных. База данных считается восстановимой, если при ее конфигурировании для *logretain* задано значение `RECOVERY` или для *userexit* задано значение `ON`.

db2inidb - инициализировать зеркальную копию базы данных

В среде с отделенной зеркальной копией эта команда используется для инициализации зеркальной копии базы данных с различными целями.

Полномочия

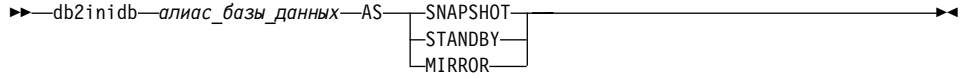
Одни из следующих:

- *sysadm*
- *sysctrl*
- *sysmaint*

Необходимое соединение

Нет

Синтаксис команды



Параметры команды

алиас_базы_данных
Указывает алиас инициализируемой базы данных.

SNAPSHOT
Задает, что база данных зеркальной копии будет инициализирована как клон первичной базы данных. Эта база данных предназначена только для чтения.

STANDBY
Задает, что база данных будет переведена в состояние отложенного восстановления повтором. Новые файлы журнала первичной базы данных считываются и применяются к этой резервной базе данных. Резервную базу данных можно будет использовать вместо первичной в случае аварии.

MIRROR
Задает, что зеркальная копия базы данных предназначена для использования в качестве образа резервной копии, который можно использовать для восстановления первичной базы данных.

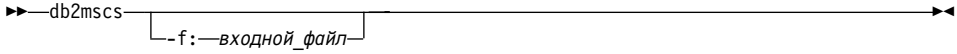
db2mscs - Конфигурирование утилиты Failover Windows NT

Создает инфраструктуру для поддержки восстановления при отказах DB2 в Windows NT/2000 при помощи Microsoft Cluster Server (MSCS). Эту утилиту можно использовать для восстановления при отказах как в одноразделной, так и в многораздельной среде.

Полномочия

Пользователь должен быть зарегистрирован под учетной записью, принадлежащей к группе администратором на каждом компьютере кластера MSCS.

Синтаксис команды



Параметры команды

-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.

ARCHIVE LOG

Закрывает и усекает активный файл журнала восстановимой базы данных. Если включен обработчик пользователя, посылает требование на архивирование.

Область действия

В среде MPP эта команда закрывает и усекает активные журналы на всех узлах; однако можно указать и поднабор узлов.

Полномочия

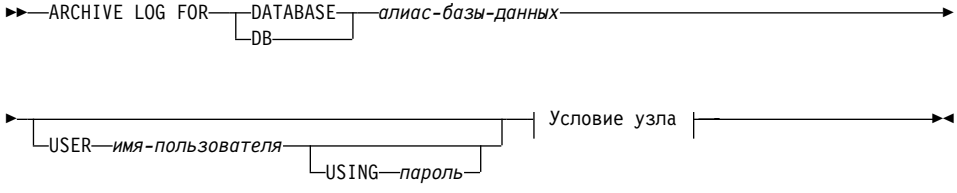
Одни из следующих:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

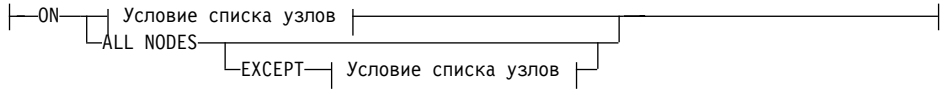
Необходимое соединение

Эта команда автоматически устанавливает соединение с указанной базой данных. Если это соединение уже существует, возвращается ошибка.

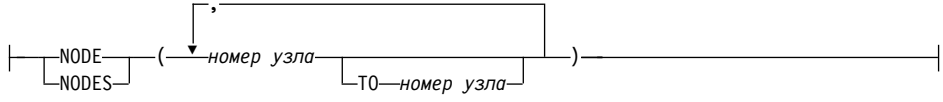
Синтаксис команды



Условие узла:



Условие списка узлов:



Параметры команды

DATABASE алиас-базы-данных
Задает алиас базы данных, активный журнал которого нужно сархивировать.

USER имя-пользователя
Задает имя пользователя для попытки соединения.

USING пароль
Задает пароль для имени пользователя.

ON ALL NODES
Указывает, что команду следует выполнить для всех узлов в файле `db2nodes.cfg`.Это значение используется по умолчанию, если не задано условие узла.

EXCEPT
Указывает, что команду следует выполнить для всех узлов в файле `db2nodes.cfg`, кроме приведенных в списке узлов.

ON NODE/ON NODES
Указывает, что следует архивировать журналы для заданной базы данных на наборе узлов.

номер узла
Задает номер узла в списке узлов.

TO номер узла
Используется при задании диапазона узлов, для которого следует архивировать журналы. В список включаются все узлы от первого указанного номера узла до второго указанного номера узла включительно.

Замечания по использованию

Эту команду можно использовать для сбора полного комплекта файлов журнала до известного момента. Затем файлы журнала можно использовать для обновления резервной базы данных.

Если другие прикладные программы выполняют транзакции в то время, что выдана эта команда, небольшое снижение производительности будет замечено при сохранении на диск буфера журнала; другим транзакциям, пытающимся записать в буфер записи журнала, придется ждать завершения сохранения буфера.

Эта команда приводит к потере базой данных части ее пространства LSN, что учащает исчерпание допустимых LSN.

INITIALIZE TAPE

DB2 for Windows NT/2000 поддерживает операции резервного копирования и восстановления для потоковых ленточных устройств. Эта команда используется для инициализации ленты.

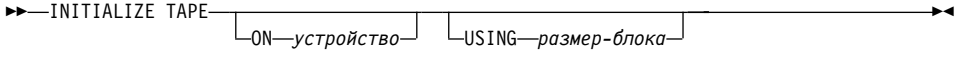
Полномочия

Нет

Необходимое соединение

Нет

Синтаксис команды



Параметры команды

ON устройство

Задает допустимое имя ленточного устройства. Значение по умолчанию - `\\.\TAPE0`.

USING размер-блока

Задает для устройства размер блока в байтах. Устройство инициализируется для использования указанного размера блока, если его значение лежит в диапазоне поддерживаемых этим устройством размеров блоков.

Примечание: Размер буфера, заданный в “Команда BACKUP DATABASE” на стр. 95, и в “команда RESTORE DATABASE” на стр. 122 должен быть кратным указанному здесь размеру блока.

Если значение для этого параметра не указано, устройство инициализируется с использованием размера блока по умолчанию. Если указано нулевое значение, устройство инициализируется для использования переменного размера блока; если устройство не поддерживает режим переменного размера блока, возвращается ошибка.

Смотрите также

“REWIND TAPE” на стр. 348

“SET TAPE POSITION” на стр. 349.

LIST HISTORY

Создает список записей в файле хронологии. Файл хронологии содержит записи событий восстановления и управления. События восстановления - это создание полных резервных копий базы данных, копий уровня табличных пространств, инкрементных копий, операции восстановления и повтора транзакций. Кроме того, записываются события создания, изменения и переименования табличного пространства, сбора статистики, реорганизации таблицы, отбрасывания таблицы и загрузки.

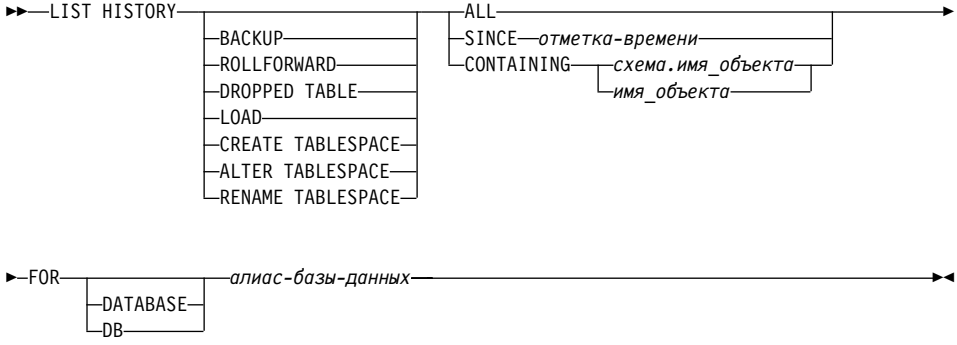
Полномочия

Нет

Необходимое соединение

Экземпляр. Явное подключение не требуется. Если база данных указана как удаленная, при выполнении команды производится подключение экземпляра к удаленному узлу.

Синтаксис команды



Параметры команды

HISTORY
Выводит список всех событий, записанных в файле хронологии на данный момент.

BACKUP
Выводит список всех операций резервного копирования и восстановления.

ROLLFORWARD
Выводит список операций повтора транзакций.

DROPPED TABLE
Выводит записи об отброшенных таблицах.

LOAD Выводит записи об операциях загрузки.

CREATE TABLESPACE
Выводит список операций создания и отбрасывания табличных пространств.

RENAME TABLESPACE
Выводит список операций переименования табличных пространств.

ALTER TABLESPACE
Выводит список операций изменения табличных пространств.

ALL Выводит список всех записей заданного типа в файле хронологии.

SINCE отметка-времени
Можно задать полную отметку времени (в формате *ггггммддччннсс*), или начальный фрагмент (минимум *гггг*). Выводится список всех записей с отметками времени, равными или большими заданной.

CONTAINING схема.имя_объекта
Специфицированное имя однозначно определяет таблицу.

CONTAINING имя_объекта
Неспецифицированное имя однозначно определяет табличное пространство.

FOR DATABASE алиас-базы-данных
Используется для идентификации базы данных, для которой выводится информация файла хронологии восстановления.

Примеры

```
db2 list history since 19980201 for sample
db2 list history backup containing userspace1 for sample
db2 list history dropped table all for db sample
```

Замечания по использованию

В отчете, генерируемом этой командой, используются следующие обозначения:

Операция

```
A - Создать табличное пространство
B - Резервное копирование
C - Загрузить копию
D - Отброшенная таблица
F - Повтор
G - Реорганизовать таблицу
L - Загрузка
N - Переименовать табличное пространство
O - Отбросить табличное пространство
Q - Стабилизировать
R - Восстановить
S - Запустить статистику
T - Изменить таблично пространство
U - Выгрузить
```

Тип

```
Типы резервной копии:

   F - Автономная
   N - Оперативная
   I - Инкрементная автономная
   O - Инкрементная оперативная
   D - Разностная автономная
   E - Разностная оперативная

Типы восстановления:

   E - До конца журналов
   P - До момента времени

Типы загрузки:

   I - Вставка
   R - Замена

Типы изменения табличного пространства:

   C - Добавить контейнеры
   R - Равномерно распределить содержимое по контейнерам

Типы стабилизации:

   S - Стабилизация с совместным доступом
   U - Стабилизация с изменениями
   X - Монопольная стабилизация
   Z - Стабилизация со сбросом
```

Если повтор транзакций выполняются до конца журналов, ID резервной копии указывает предельное возможное время, то есть 99991231235959.

PRUNE HISTORY/LOGFILE

Используется для удаления записей из файла хронологии восстановлений и для удаления файлов журналов из пути активных файлов журналов. Удаление записей из файла хронологии восстановлений может потребоваться, если файл станет чрезмерно большим при длительном сроке хранения. Удаление файлов журналов из пути активных журналов может потребоваться, если журналы архивируются вручную (а не с использованием программы обработчика пользователя).

Полномочия

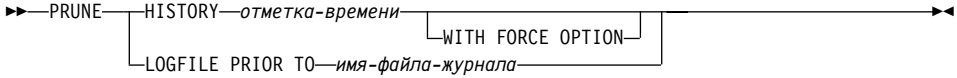
Одни из следующих:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Необходимое соединение

База данных

Синтаксис команды



Параметры команды

HISTORY отметка-времени

Идентифицирует диапазон удаляемых записей в файле хронологии восстановлений. Для отметки времени можно задать полный формат (*гггг*ммддччммсс) или только начальный префикс (минимум *гггг*). Из файла хронологии восстановлений удаляются все записи с отметками времени до указываемой включительно.

WITH FORCE OPTION

Задает, что записи будут сокращаться в соответствии с заданной отметкой времени, даже если из файла удаляются некоторые записи из самого последнего набора восстановления. Набор восстановления - это самая последняя полная резервная копия базы данных, в состав которой входят все восстановления данного образа резервной копии. Если этот параметр не задается, все записи из этого образа резервной копии будут оставлены в хронологии.

LOGFILE PRIOR TO имя-файла-журнала

Задает строку для имени файла журнала, например, `S0000100.LOG`. Будут удалены все файлы журнала, предшествующие заданному файлу

журнала (но не сам заданный файл). Параметр LOGRETAIN конфигурации базы данных должен иметь значение RECOVERY или CAPTURE.

Примеры

Чтобы удалить из файла хронологии восстановлений записи для всех восстановлений, загрузок, резервных копий табличных пространств и полных резервных копий базы данных до 1 декабря 1994 года включительно, введите команду:

```
db2 prune history 199412
```

Примечание: 199412 интерпретируется как 19941201000000.

Замечания по использованию

Сокращение записей резервной копии из файла хронологии приводит к удалению связанных резервных копий файлов на серверах менеджеров связей данных DB2.

REWIND TAPE

DB2 for Windows NT/2000 поддерживает операции резервного копирования и восстановления для потоковых ленточных устройств. Эта команда используется для перемотки ленты.

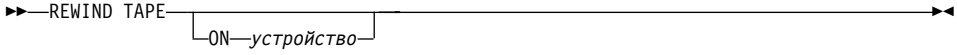
Полномочия

Нет

Необходимое соединение

Нет

Синтаксис команды



Параметры команды

ON устройство
Задает допустимое имя ленточного устройства. Значение по умолчанию - `\\.\TAPE0`.

Смотрите также

“INITIALIZE TAPE” на стр. 342

“SET TAPE POSITION” на стр. 349.

SET TAPE POSITION

DB2 for Windows NT/2000 поддерживает операции резервного копирования и восстановления для потоковых ленточных устройств. Эта команда используется для позиционирования ленты.

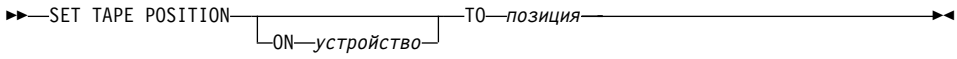
Полномочия

Нет

Необходимое соединение

Нет

Синтаксис команды



Параметры команды

ON устройство

Задает допустимое имя ленточного устройства. Значение по умолчанию - `\\.\TAPE0`.

TO позиция

Указывает метку для позиционирования ленты. DB2 for Windows NT/2000 записывает на ленту метки после каждой резервной копии. Значение 1 указывает первую позицию, 2 - вторую позицию и так далее. Например, если магнитная лента позиционирована на метке 1, восстанавливаться будет архив 2.

Смотрите также

“INITIALIZE TAPE” на стр. 342

“REWIND TAPE” на стр. 348.

UPDATE HISTORY FILE

Изменяет положение, тип устройства или комментарий в записи файла хронологии.

Полномочия

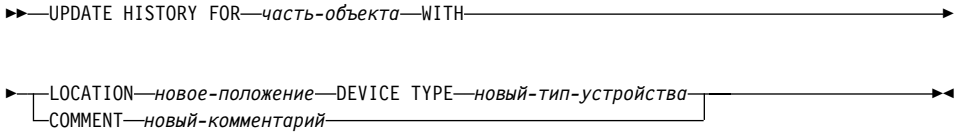
Одни из следующих:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Необходимое соединение

База данных

Синтаксис команды



Параметры команды

FOR часть-объекта
Задает идентификатор для образа копии. Это отметка времени с необязательным последовательным номером от 001 до 999.

LOCATION новое-положение
Задает новое физическое положение образа резервной копии. Интерпретация этого параметра зависит от типа устройства.

DEVICE TYPE новый-тип-устройства
Задает новый тип устройства для хранения образа резервной копии. Допустимые типы устройств:

D Диск

K Дискета

T Лента

A TSM

U Обработчик пользователя

O Другой

COMMENT новый-комментарий
Задает новый комментарий, описывающий запись.

Примеры

Чтобы изменить запись файла хронологии для полной резервной копии, снятой 13 апреля 1997 года в 10.00, введите:

```
db2 update history for 19970413100000001 with
   location /backup/dbbackup.1 device type d
```

Замечания по использованию

Файл хронологии используется администраторами базы данных для хранения записей. Он используется внутри DB2 при автоматическом восстановлении из инкрементных резервных копий.

Смотрите также

“PRUNE HISTORY/LOGFILE” на стр. 346.

UPDATE HISTORY FILE

Приложение D. Дополнительные API и связанные с ними структуры данных

© Copyright IBM Corp. 2001

db2ArchiveLog - API архивирования активного журнала

Закрывает и усекает активный файл журнала восстановимой базы данных. Если включен обработчик пользователя, посылает требование на архивирование.

Область действия

В среде MPP этот API закрывает и усекает активные журналы на всех узлах.

Полномочия

Одни из следующих:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Необходимое соединение

Этот API автоматически устанавливает соединение с указанной базой данных. Если это соединение уже существует, возвращается ошибка.

Файл включения API

db2ApiDf.h

Синтаксис API на языке C

```
/* Файл: db2ApiDf.h */
/* API:  архивировать активный журнал */
/* ... */
SQL_API_RC SQL_API_FN
   db2ArchiveLog (
      db2Uint32 version,
      void * pDB2ArchiveLogStruct,
      struct sqlca * pSqlca);

typedef struct
{
   char * piDatabaseAlias;
   char * piUserName;
   char * piPassword;
   db2Uint16           iAllNodeFlag;
   db2Uint16           iNumNodes;
   SQL_PDB_NODE_TYPE * piNodeList;
   db2Uint32 iOptions;
} db2ArchiveLogStruct;
/* ... */
```

Общий синтаксис API

```
/* Файл: db2ApiDf.h */
/* API:  архивировать активный журнал */
/* ... */
SQL_API_RC SQL_API_FN
   db2gArchiveLog (
      db2Uint32 version,
      void * pDB2gArchiveLogStruct,
      struct sqlca * pSqlca);

typedef struct
{
   db2Uint32 iAliasLen;
   db2Uint32 iUserNameLen;
   db2Uint32 iPasswordLen;
   char * piDatabaseAlias;
   char * piUserName;
   char * piPassword;
   db2Uint16           iAllNodeFlag;
   db2Uint16           iNumNodes;
   SQL_PDB_NODE_TYPE * piNodeList;
   db2Uint32 iOptions;
} db2gArchiveLogStruct;
/* ... */
```

Параметры API

version
Входной. Задает уровень версии и выпуска для переменной, передаваемой во втором параметре, *pDB2ArchiveLogStruct*.

pDB2ArchiveLogStruct
Входной. Указатель на структуру *db2ArchiveLogStruct*.

pSqlca Выходной. Указатель на структуру *sqlca*. Дополнительную информацию об этой структуре смотрите в публикациях *Administrative API Reference* или *SQL Reference*.

iAliasLen
Входной. 4-байтное целое без знака, равное длине (в байтах) алиаса базы данных.

iUserNameLen
Входной. 4-байтное целое без знака, задающее длину в байтах имени пользователя. Задайте равным нулю, если имя пользователя не используется.

iPasswordLen
Входной. 4-байтное целое без знака, задающее длину пароля в байтах. Задайте равным нулю, если пароль не используется.

piDatabaseAlias
Входной. Строка, содержащая алиас (как он внесен в системный каталог баз данных) базы данных, для которой надо архивировать активный журнал.

piUserName
Входной. Строка, содержащая имя пользователя для попытки соединения.

piPassword
Входной. Строка, содержащая пароль для попытки соединения.

iAllNodeFlag
Входной. Только для MPP. Флаг, указывающий, надо ли применять операцию ко всем узлам, перечисленным в файле `db2nodes.cfg`.Допустимые значения:

DB2ARCHIVELOG_ALL_NODES
Применяется на всех узлах (piNodeList должен быть пуст). Это значение по умолчанию.

DB2ARCHIVELOG_NODE_LIST
Применяется на всех узлах из переданного в параметре *piNodeList* списка.

DB2ARCHIVELOG_ALL_EXCEPT
Применяется на всех узлах *кроме* узлов из переданного в параметре *piNodeList* списка.

iNumNodes
Входной. Только для MPP. Задает число узлов в списке *piNodeList*.

piNodeList
Входной. Только для MPP. Указатель на массив номеров узлов, на которых выполняется операция архивирования журнала.

iOptions
Входной. Зарезервирован для будущего использования.

Замечания по использованию

Этот API можно использовать для сбора полного комплекта файлов журнала до известного момента. Затем файлы журнала можно использовать для обновления резервной базы данных.

Если другие прикладные программы выполняют транзакции в то время, что вызван этот интерфейс, небольшое снижение производительности будет замечено при сохранении на диск буфера журнала; другим транзакциям, пытающимся записать в буфер записи журнала, придется ждать завершения сохранения буфера.

Этот API заставляет базу данных потерять часть ее пространства LSN и тем участить исчерпание допустимых LSN.

db2HistoryCloseScan - API Закрыть просмотр файла хронологии восстановлений

Завершает просмотр файла хронологии восстановления и освобождает все ресурсы DB2, необходимые для этого просмотра. Этому API должен предшествовать удачный вызов “db2HistoryOpenScan - API Открыть просмотр файла истории восстановления” на стр. 364.

Полномочия

Нет

Необходимое соединение

Экземпляр. Перед вызовом этого API нет необходимости вызывать **sqleatin**.

Файл включения API

db2ApiDf.h

Синтаксис API на языке C

```
/* Файл: db2ApiDf.h */
/* API: Закрыть просмотр файла хронологии восстановлений */
/* ... */
SQL_API_RC  SQL_API_FN
  db2HistoryCloseScan (
    db2Uint32 version,
    void * piHandle,
    struct sqlca * pSqlca);
/* ... */
```

Общий синтаксис API

```
/* Файл: db2ApiDf.h */
/* API: Закрыть просмотр файла хронологии восстановлений */
/* ... */
SQL_API_RC  SQL_API_FN
  db2GenHistoryCloseScan (
    db2Uint32 version,
    void * piHandle,
    struct sqlca * pSqlca);
/* ... */
```

Параметры API

version

Входной. Задает уровень версии и выпуска для второго параметра, *piHandle*.

piHandle

Входной. Задает указатель на хэндл для доступа просмотра, возвращенный “db2HistoryOpenScan - API Открыть просмотр файла истории восстановления” на стр. 364.

pSqlca Выходной. Указатель на структуру *sqlca*. Дополнительную информацию об этой структуре смотрите в публикациях *Administrative API Reference* или *SQL Reference*.

Синтаксис API REXX

```
CLOSE RECOVERY HISTORY FILE :id-просмотра
```

Параметры API REXX

id-просмотра
Переменная хоста, содержащая идентификатор просмотра, возвращенный операцией OPEN RECOVERY HISTORY FILE SCAN.

Замечания по использованию

Подробное описание использования API файла хронологии восстановлений смотрите в “db2HistoryOpenScan - API Открыть просмотр файла истории восстановления” на стр. 364.

Смотрите также

“db2HistoryGetEntry - API Получить следующую запись файла хронологии восстановлений” на стр. 360

“db2HistoryOpenScan - API Открыть просмотр файла истории восстановления” на стр. 364

“API db2Prune” на стр. 374

“db2HistoryUpdate - API Изменить файл хронологии восстановлений” на стр. 370.

db2HistoryGetEntry - API Получить следующую запись файла хронологии восстановлений

Получает следующую запись из файла хронологии восстановлений. Перед этим API должен быть выполнен успешный вызов “db2HistoryOpenScan - API Открыть просмотр файла истории восстановления” на стр. 364.

Полномочия

Нет

Необходимое соединение

Экземпляр. Перед вызовом этого API не обязательно вызывать **sqleatin**.

Файл включения API

db2ApiDf.h

Синтаксис API на языке C

```
/* Файл: db2ApiDf.h */
/* API: Получить следующую запись файла хронологии восстановлений */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryGetEntry (
    db2Uint32 version,
    void * pDB2HistoryGetEntryStruct,
    struct sqlca * pSqlca);

typedef struct
{
  db2Uint16 iHandle,
  db2Uint16 iCallerAction,
  struct db2HistData * pioHistData
} db2HistoryGetEntryStruct;
/* ... */
```

Общий синтаксис API

```
/* Файл: db2ApiDf.h */
/* API: Получить следующую запись файла хронологии восстановлений */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryGetEntry (
    db2Uint32 version,
    void * pDB2GenHistoryGetEntryStruct,
    struct sqlca * pSqlca);

typedef struct
{
  db2Uint16 iHandle,
  db2Uint16 iCallerAction,
  struct db2HistData * pioHistData
} db2GenHistoryGetEntryStruct;
/* ... */
```

Параметры API

version
Входной. Задает уровень версии и выпуска для структуры, передаваемой во втором параметре - *pDB2HistoryGetEntryStruct*.

pDB2HistoryGetEntryStruct
Входной. Указатель на структуру *db2HistoryGetEntryStruct*.

pSqlca Выходной. Указатель на структуру *sqlca*. Дополнительную информацию об этой структуре смотрите в публикациях *Administrative API Reference* или *SQL Reference*.

iHandle
Входной. Содержит хэндл доступа для просмотра, возвращенный “db2HistoryOpenScan - API Открыть просмотр файла истории восстановления” на стр. 364.

iCallerAction
Входной. Указывает тип действия, которое будет выполнено. Допустимые значения (определены в `db2ApiDf`):

DB2HISTORY_GET_ENTRY
Получить следующую запись, но без командных данных.

DB2HISTORY_GET_DDL
Получить только командные данные от предыдущей выборки.

DB2HISTORY_GET_ALL
Получить следующую запись, включая все данные.

pioHistData

Входной. Указатель на структуру *db2HistData*. Подробную информацию об этой структуре смотрите в “Структура данных: db2HistData” на стр. 381.

Синтаксис API REXX

```
GET RECOVERY HISTORY FILE ENTRY :id-просмотра [USING :значение]
```

Параметры API REXX

id-просмотра

Переменная хоста, содержащая идентификатор просмотра, который возвращен вызовом OPEN RECOVERY HISTORY FILE SCAN.

значение

Составная переменная хоста REXX, в которой возвращается информация из записи файла хронологии восстановлений. Далее XXX будет обозначать имя этой переменной хоста:

XXX.0	Число элементов первого уровня в переменной (всегда 15)
XXX.1	Число элементов табличных пространств
XXX.2	Число элементов используемых табличных пространств
XXX.3	OPERATION (тип выполняемой операции)
XXX.4	OBJECT (уровень операции)
XXX.5	OBJECT_PART (отметка времени и порядковый номер)
XXX.6	OPTYPE (спецификатор операции)
XXX.7	DEVICE_TYPE (тип используемого устройства)
XXX.8	FIRST_LOG (ID самого раннего журнала)
XXX.9	LAST_LOG (ID текущего журнала)
XXX.10	BACKUP_ID (идентификатор резервной копии)
XXX.11	SCHEMA (спецификатор для имени таблицы)
XXX.12	TABLE_NAME (имя загруженной таблицы)
XXX.13.0	NUM_OF_TABLESPACES (количество табличных пространств, участвующих в резервном копировании или восстановлении)
XXX.13.1	Имя первого скопированного/восстановленного табличного пространства.

XXX.13.2	Имя второго скопированного/восстановленного табличного пространства.
XXX.13.3	и так далее
XXX.14	LOCATION (где хранится резервная копия или копия)
XXX.15	COMMENT (текст описания данной записи).

Замечания по использованию

Возвращаемые записи будут выбираться с использованием значений, указанных при вызове “db2HistoryOpenScan - API Открыть просмотр файла истории восстановления” на стр. 364.

Подробное описание использования различных API файла хронологии восстановлений смотрите в разделе “db2HistoryOpenScan - API Открыть просмотр файла истории восстановления” на стр. 364.

Смотрите также

“db2HistoryCloseScan - API Закрыть просмотр файла хронологии восстановлений” на стр. 358

“db2HistoryOpenScan - API Открыть просмотр файла истории восстановления” на стр. 364

“API db2Prune” на стр. 374

“db2HistoryUpdate - API Изменить файл хронологии восстановлений” на стр. 370.

db2HistoryOpenScan - API Открыть просмотр файла истории восстановления

Запуск просмотра файла хронологии восстановления.

Полномочия

Нет

Необходимое соединение

Экземпляр. Перед вызовом этого API нет необходимости вызывать **sqleatin**. Если база данных внесена в каталог как удаленная, производится подключение к удаленному узлу.

Файл включения API

db2ApiDf.h

Синтаксис API на языке C

```
/* Файл: db2ApiDf.h */
/* API: Открыть просмотр файла истории восстановления */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryOpenScan (
    db2Uint32 version,
    void * pDB2HistoryOpenStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piDatabaseAlias,
  char * piTimestamp,
  char * piObjectName,
  db2Uint32 oNumRows,
  db2Uint16 iCallerAction,
  db2Uint16 oHandle
} db2HistoryOpenStruct;
/* ... */
```

Общий синтаксис API

```
/* Файл: db2ApiDf.h */
/* API: Открыть просмотр файла истории восстановления */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryOpenScan (
    db2Uint32 version,
    void * pDB2GenHistoryOpenStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piDatabaseAlias,
  char * piTimestamp,
  char * piObjectName,
  db2Uint32 oNumRows,
  db2Uint16 iCallerAction,
  db2Uint16 oHandle
} db2GenHistoryOpenStruct;
/* ... */
```

Параметры API

version
Входной. Задает уровень версии и выпуска для структуры, передаваемой во втором параметре, *pDB2HistoryOpenStruct*.

pDB2HistoryOpenStruct
Входной. Указатель на структуру *db2HistoryOpenStruct*.

pSqlca Выходной. Указатель на структуру *sqlca*. Дополнительную информацию об этой структуре смотрите в публикациях *Administrative API Reference* или *SQL Reference*.

piDatabaseAlias
Входной. Указатель на строку символов, содержащую алиас базы данных.

piTimestamp
Входной. Указатель на строку символов, задающую отметку времени, которая будет использоваться для выбора записей. Выбираются записи, у которых отметка времени равна этому значению или больше него. Установка для этого параметра пустого или нулевого значения отключает фильтрацию записей с использованием отметки времени.

piObjectName
Входной. Указатель на строку символов, задающую имя объекта, которое будет использоваться для выбора записей. Объект может быть таблицей или табличным пространством. Для таблицы необходимо

задать полное имя. Установка для этого параметра пустого или нулевого значения отключает фильтрацию записей с использованием имени объекта.

oNumRows

Выходной. После выхода из API этот параметр содержит число совпадающих записей файла хронологии восстановления.

iCallerAction

Входной. Указывает тип действия, которое будет выполнено. Допустимые значения (определены в `db2ApiDf`):

DB2HISTORY_LIST_HISTORY

Выводит список всех событий, в данное время зарегистрированные в файле хронологии.

DB2HISTORY_LIST_BACKUP

Выводит список операций резервного копирования и восстановления из резервной копии.

DB2HISTORY_LIST_ROLLFORWARD

Выводит список операций повтора транзакций.

DB2HISTORY_LIST_DROPPED_TABLE

Выводит список записей об отбрасывании таблиц. Связанное с записью поле DDL не возвращается. Чтобы получить для записи информацию DDL, необходимо вызвать из программы “db2HistoryGetEntry - API Получить следующую запись файла хронологии восстановлений” на стр. 360 с действием `DB2HISTORY_GET_DDL` немедленно после считывания этой записи.

DB2HISTORY_LIST_LOAD

Выводит список операций загрузки.

DB2HISTORY_LIST_CRT_TABLESPACE

Выводит список операций создания и отбрасывания табличных пространств.

DB2HISTORY_LIST_REN_TABLESPACE

Выводит список операций переименования табличных пространств.

DB2HISTORY_LIST_ALT_TABLESPACE

Выводит список операций изменения табличных пространств. Связанное с записью поле DDL не возвращается. Чтобы получить для записи информацию DDL, необходимо вызвать из программы “db2HistoryGetEntry - API Получить следующую запись файла хронологии восстановлений” на стр. 360 с действием `DB2HISTORY_GET_DDL` немедленно после считывания этой записи.

DB2HISTORY_LIST_RUNSTATS
Выводит список операций RUNSTATS. Это значение в настоящее время не поддерживаются.

DB2HISTORY_LIST_REORG
Выводит список операций REORGANIZE TABLE. Это значение в настоящее время не поддерживаются.

oHandle
Выходной. После возврата из API этот параметр содержит хэндл для доступа к просмотру. Он впоследствии используется в “db2HistoryGetEntry - API Получить следующую запись файла хронологии восстановлений” на стр. 360 и “db2HistoryCloseScan - API Закрыть просмотр файла хронологии восстановлений” на стр. 358.

Синтаксис API REXX

```
OPEN [BACKUP] RECOVERY HISTORY FILE FOR алиас_базы_данных
[OBJECT имя_объекта] [TIMESTAMP :отметка_времени]
USING :значение
```

Параметры API REXX

алиас_базы_данных
Алиас базы данных, файл хронологии которой будет просмотрен.

имя_объекта
Указывает имя объекта, которое будет использоваться для выбора записей. Объект может быть таблицей или табличным пространством. Для таблицы необходимо задать полное имя. Установка для этого параметра пустого значения отключает фильтрацию записей с использованием *имя_объекта*.

отметка_времени
Указывает отметку времени, которая будет использоваться для выбора записей. Выбираются записи, у которых отметка времени равна этому значению или больше него. Установка для этого параметра пустого значения отключает фильтрацию записей с использованием *отметка_времени*.

значение
Составная переменная хоста REXX, в которую возвращается информация из файла хронологии восстановления. Далее XXX будет обозначать имя этой переменной хоста.

XXX.0 Число элементов в переменной (всегда 2)

XXX.1 Идентификатор (хэндл) для последующего доступа к просмотру

XXX.2 Число совпадающих элементов файла хронологии восстановления.

Замечания по использованию

Сочетание из отметки времени, имени объекта и действия вызывающей программы, которое может быть использовано для фильтрации записей. Возвращаются только записи, прошедшие все указанные фильтры.

Фильтрующее действие имени объекта зависит от указанного значения:

- Указание таблицы возвратит записи для операций загрузки, поскольку это единственная информация для таблиц в файле хронологии.
- Указание табличного пространства возвратит записи для операций резервного копирования, восстановления из резервной копии и загрузки для этого табличного пространства.

Примечание: Чтобы возвращались записи для таблиц, они должны быть указаны как *схема.имя_таблицы*. Если указать *имя_таблицы*, будут возвращены только записи для табличных пространств.

Для процесса максимально допускается восемь просмотров файла хронологии.

Чтобы вывести все записи в файле хронологии, типичная прикладная программа выполнит следующие действия:

1. Вызывает **db2HistoryOpenScan**, который возвратит *oNumRows*.
2. Размещает структуру *db2HistData* с пространством для *n* полей *oTablespace*, где *n* - определенное число.
3. Устанавливает для поля *iDB2NumTablespace* структуры *db2HistData* значение *n*.
4. Выполняет в цикле следующие действия:
   - Вызывает **db2HistoryGetEntry** для чтения из файла хронологии.
   - Если **db2HistoryGetEntry** возвращает SQLCODE `SQL_RC_OK`, поле *sqld* структуры *db2HistData* используется для определения числа возвращенных записей табличных пространств.
   - Если **db2HistoryGetEntry** возвращает SQLCODE `SQLUH_SQLUHINFO_VARS_WARNING`, для всех табличных пространств, которые пытается возвратить DB2, это значит, что было выделено недостаточно места; освободите и повторно разместите структуру *db2HistData* с достаточным пространством для записей табличных пространств *oDB2UsedTablespace*, а также установите *oDB2UsedTablespace* для *iDB2NumTablespace*.
   - Если **db2HistoryGetEntry** возвращает SQLCODE `SQLE_RC_NOMORE`, все записи файла хронологии восстановления прочитаны.
   - Любой другой SQLCODE означает ошибку.
5. Когда вся информация прочитана, вызывает “db2HistoryCloseScan - API Закрыть просмотр файла хронологии восстановлений” на стр. 358 для освобождения ресурсов, выделенных при вызове **db2HistoryOpenScan**.

Для облегчения определения, сколько памяти требуется для структуры *db2HistData* с пространством для *n* полей *oTablespace*, используется макрокоманда SQLUHINFOSIZE(*n*), определенная в `sqlutil`.

Смотрите также

“db2HistoryCloseScan - API Закрыть просмотр файла хронологии восстановлений” на стр. 358

“db2HistoryGetEntry - API Получить следующую запись файла хронологии восстановлений” на стр. 360

“API db2Prune” на стр. 374

“db2HistoryUpdate - API Изменить файл хронологии восстановлений” на стр. 370.

db2HistoryUpdate - API Изменить файл хронологии восстановлений

Изменяет положение, тип устройства или комментарий в записи файла хронологии.

Полномочия

Одни из следующих:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Необходимое соединение

База данных. Чтобы изменить записи в файле хронологии базы данных, отличной от базы данных по умолчанию, надо перед вызовом этого API установить соединение с базой данных.

Файл включения API

db2ApiDf.h

Синтаксис API на языке C

```
/* Файл: db2ApiDf.h */
/* API: Изменить файл хронологии восстановлений */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryUpdate (
    db2Uint32 version,
    void * pDB2HistoryUpdateStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piNewLocation,
  char * piNewDeviceType,
  char * piNewComment,
  db2Uint32 iEID
} db2HistoryUpdateStruct;
/* ... */
```

Общий синтаксис API

```
/* Файл: db2ApiDf.h */
/* API: Изменить файл хронологии восстановлений */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryUpdate (
    db2Uint32 version,
    void * pDB2GenHistoryUpdateStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piNewLocation,
  char * piNewDeviceType,
  char * piNewComment,
  db2Uint32 iEID
} db2GenHistoryUpdateStruct;
/* ... */
```

Параметры API

version
Входной. Задает уровень версии и выпуска для структуры, передаваемой во втором параметре - *pDB2HistoryUpdateStruct*.

pDB2HistoryUpdateStruct
Входной. Указатель на структуру *db2HistoryUpdateStruct*.

pSqlca Выходной. Указатель на структуру *sqlca*. Дополнительную информацию об этой структуре смотрите в публикациях *Administrative API Reference* или *SQL Reference*.

piNewLocation
Входной. Указатель на строку, задающую новое положение для образов резервной копии, копий восстановления или загрузки. Если задать для этого параметра значение NULL или указатель на ноль, положение останется прежним.

piNewDeviceType
Входной. Указатель на строку, задающую новый тип устройства для хранения образов резервной копии, копий восстановления или загрузки. Если задать для этого параметра значение NULL или указатель на ноль, тип останется прежним.

piNewComment
Входной. Указатель на строку, задающую новый комментарий для описания записи. Если задать для этого параметра значение NULL или указатель на ноль, комментарий останется прежним.

iEID Входной. Уникальный идентификатор, с помощью которого можно изменять определенную запись в файле хронологии.

db2HistoryUpdate - API Изменить файл хронологии восстановлений

Синтаксис API REXX

```
UPDATE RECOVERY HISTORY USING :значение
```

Параметры API REXX

значение

Составная переменная хоста REXX с информацией, относящуюся к новому положению для записи в файле хронологии восстановлений. Далее XXX обозначает имя этой переменной хоста:

XXX.0 Количество элементов в этой переменной (должно быть от 1 до 4)

XXX.1 OBJECT_PART (отметка времени с порядковым номером от 001 до 999)

XXX.2 Новое положение для образа резервной копии или копии (необязательный параметр)

XXX.3 Новое устройство для хранения образа резервной копии или копии (необязательный параметр)

XXX.4 Новый комментарий (необязательный параметр).

Замечания по использованию

Это функция изменяет информацию; прежняя информация стирается и не может быть восстановлена. Эти изменения не регистрируются в журналах.

Файл хронологии используется только для записи операций. Он не используется непосредственно функциями восстановления или повтора транзакций. При операции восстановления можно указать положение резервной копии, и файл хронологии полезен для отслеживания этого положения. Эту информацию можно потом передать утилите резервного копирования (смотрите раздел “API резервного копирования базы данных” на стр. 99). Аналогично, если образ копии загрузки был перенесен, утилите повтора транзакций надо передать информацию о новом положении и типе среды хранения. Более подробную информацию смотрите в разделе “API повтора транзакций базы данных” на стр. 164.

Смотрите также

“db2HistoryCloseScan - API Закрыть просмотр файла хронологии восстановлений” на стр. 358

“db2HistoryGetEntry - API Получить следующую запись файла хронологии восстановлений” на стр. 360

“db2HistoryOpenScan - API Открыть просмотр файла истории восстановления” на стр. 364

“API db2Prune” на стр. 374.

API db2Prune

Удаляет записи из файла хронологии восстановлений или файлы журналов из каталога активных журналов.

Полномочия

Одни из следующих:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Необходимое соединение

База данных. Чтобы удалить записи из файла хронологии восстановлений базы данных, отличной от базы данных по умолчанию, надо перед вызовом этого API установить соединение с базой данных.

Файл включения API

db2ApiDf.h

Синтаксис API на языке C

```
/* Файл: db2ApiDf.h */
/* API: Сократить файл хронологии восстановлений */
/* ... */
SQL_API_RC SQL_API_FN
  db2Prune (
    db2Uint32 version,
    void * pDB2PruneStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piString,
  db2Uint32 iEID,
  db2Uint32 iCallerAction,
  db2Uint32 iOptions
} db2PruneStruct;
/* ... */
```

Общий синтаксис API

```
/* Файл: db2ApiDf.h */
/* API: Сократить файл хронологии восстановлений */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenPrune (
    db2Uint32 version,
    void * pDB2GenPruneStruct,
    struct sqlca * pSqlca);

typedef struct
{
  db2Uint32 iStringLen;
  char * piString,
  db2Uint32 iEID,
  db2Uint32 iCallerAction,
  db2Uint32 iOptions
} db2GenPruneStruct;
/* ... */
```

Параметры API

version
Входной. Задает уровень версии и выпуска для структуры, передаваемой во втором параметре - *pDB2PruneStruct*.

pDB2PruneStruct
Входной. Указатель на структуру *db2PruneStruct*.

pSqlca Выходной. Указатель на структуру *sqlca*. Дополнительную информацию об этой структуре смотрите в публикациях *Administrative API Reference* или *SQL Reference*.

iStringLen
Входной. Задает длину *piString* в байтах.

piString
Входной. Указатель на строку, задающую отметку времени или последовательный номер журнала (LSN). Отметка времени или часть отметки времени (минимум *гггг*, то есть год) используется, чтобы выбрать записи для удаления. Все записи с отметкой времени, равной или меньшей указанной, будут удалены. Надо указать допустимую отметку времени; для параметра NULL не определено поведение по умолчанию.

С помощью этого параметра можно также передать LSN, чтобы удалить неактивные журналы.

iEID Входной. Задает уникальный идентификатор, с помощью которого можно удалить одиночную запись из файла хронологии.

iCallerAction
Входной. Указывает, какое надо выполнить действие. Допустимые значения (определены в db2ApiDf):

DB2PRUNE_ACTION_HISTORY
Удаляет записи из файла хронологии.

DB2PRUNE_ACTION_LOG
Удаляет файлы журналов из каталога активных журналов.

iOptions
Входной. Допустимые значения (определены в db2ApiDf):

DB2PRUNE_OPTION_FORCE
Принудительно удалить последнюю резервную копию.

DB2PRUNE_OPTION_LSNSTRING
Указывает, что значение *piString* - это LSN; используется, когда при вызове указано действие DB2PRUNE_ACTION_LOG.

Синтаксис API REXX

```
PRUNE RECOVERY HISTORY BEFORE :отметка-времени [WITH FORCE OPTION]
```

Параметры API REXX

отметка-времени
Переменная хоста, содержащая отметку времени. Все записи с отметкой времени, равной или меньшей указанной, удаляются из файла хронологии восстановлений.

WITH FORCE OPTION
Если задан этот параметр, файл хронологии восстановлений будет сокращен в соответствии с указанной отметкой времени, даже если придется удалить некоторые записи из последнего набора восстановления. Если этот параметр не указан, набор последнего восстановления будет сохранен, даже если его отметка времени меньше или равна заданной отметке времени.

Замечания по использованию

При сокращении файла хронологии не удаляются сами резервные копии и файлы загрузки. Чтобы освободить место на носителе, пользователь должен удалить эти файлы вручную.

ОСТОРОЖНО:
Если последняя полная резервная копия базы данных удалена с носителя (кроме того, что она удалена из файла хронологии), надо убедиться, что у всех табличных пространств, включая табличное пространство каталога и пользовательские табличные пространства, есть резервные копии. Если это не сделать, может образоваться база данных, которую нельзя восстановить, в результате могут быть утеряна часть пользовательских данных из базы данных.

Смотрите также

“db2HistoryCloseScan - API Закрыть просмотр файла хронологии восстановлений” на стр. 358

“db2HistoryGetEntry - API Получить следующую запись файла хронологии восстановлений” на стр. 360

“db2HistoryOpenScan - API Открыть просмотр файла истории восстановления” на стр. 364

“db2HistoryUpdate - API Изменить файл хронологии восстановлений” на стр. 370.

sqlurlog - API асинхронное чтение журнала

Предоставляет вызывающей программе возможность извлекать определенные записи из журналов базы данных и запрашивать у менеджера журналов информацию о текущем состоянии журналов. Этот API можно использовать только с восстановимыми базами данных. База данных является восстановимой, если в ее конфигурации для параметра *logretain* задано значение RECOVERY или для параметра *userexit* задано значение ON.

Полномочия

Одни из следующих:

- *sysadm*
- *dbadm*

Необходимое соединение

База данных

Файл включения API

sqlutil.h

Синтаксис API на языке C

```
/* Файл: sqlutil.h */
/* API: Асинхронное чтение журнала */
/* ... */
SQL_API_RC SQL_API_FN
  sqlurlog (
    sqluint32 CallerAction,
    SQLU_LSN * pStartLsn,
    SQLU_LSN * pEndLsn,
    char * pLogBuffer,
    sqluint32 LogBufferSize,
    SQLU_RLOG_INFO * pReadLogInfo,
    struct sqlca * pSqlca);
/* ... */
```

Параметры API

CallerAction

Входной. Задает выполняемую операцию.

SQLU_RLOG_READ

Читает журнал базы данных с начального до конечного номеров последовательности журнала и возвращает все распространяемые записи журнала в этом диапазоне.

SQLU_RLOG_READ_SINGLE

Читает одну запись журнала (распространяемую или нет), идентифицируемую начальным номером последовательности журнала.

SQLU_RLOG_QUERY
Запрашивает журнал базы данных. Результаты запроса будут посланы назад в структуре SQLU_RLOG_INFO (смотрите раздел “Структура данных: SQLU-RLOG-INFO” на стр. 388).

pStartLsn
Входной. Начальный номер последовательности журнала задает относительный начальный адрес в байтах для чтения журнала. Это значение должно быть началом действительной записи журнала.

pEndLsn
Входной. Конечный номер последовательности журнала задает относительный конечный адрес в байтах для чтения журнала. Это значение должно быть больше, чем *startLsn*, и не обязательно должно соответствовать концу действительной записи журнала.

pLogBuffer
Выходной. Буфер, в котором последовательно сохраняются все записи журнала с возможностью распространения в указанном диапазоне. Этот буфер должен быть достаточно большим, чтобы вмещать одну запись журнала. Имейте в виду, что размер этого буфера должен быть не меньше 32 байтов. Его максимальный размер зависит от размера запрашиваемого диапазона. В начале каждой записи журнала в буфере ставится шестибайтный номер последовательности журнала (log sequence number - LSN), соответствующий LSN следующей за ним записи журнала.

LogBufferSize
Выходной. Указывает размер буфера журнала в байтах.

pReadLogInfo
Выходной. Структура, содержащая подробную информацию относительно вызова и журнала базы данных. Дополнительную информацию об этой структуре смотрите в разделе “Структура данных: SQLU-RLOG-INFO” на стр. 388.

pSqlca Выходной. Указатель на структуру *sqlca*. Дополнительную информацию об этой структуре смотрите в публикациях *Administrative API Reference* или *SQL Reference*.

Программы примеров

C \sqllib\samples\c\asynrlog.sqc

Замечания по использованию

Если запрошенное действие - чтение журнала, вызывающая программа задает диапазон номеров записей журнала и буфер для этих записей журнала. Этот API последовательно читает журнал в запрошенном диапазоне LSN и возвращает записи, связанные с таблицами с опцией DATA CAPTURE CHANGES и структуру SQLU_RLOG_INFO с информацией об активном в данное время

журнале. Если требуемое действие - запрос, API возвращает структуру SQLU_RLOG_INFO с информацией об активном в данное время журнале.

Чтобы использовать программу асинхронного чтения журнала, сначала запросите в журнале базы данных действительный LSN начала записи. После вызова для запроса в информационной структуре чтения журнала (SQLU-RLOG-INFO) будет содержаться действительный LSN начала записи (в элементе initialLSN), который будет использоваться при вызове для чтения. Конец текущего активного журнала будет в элементе curActiveLSN информационной структуры чтения журнала. Значение, используемое при чтении в качестве конечного LSN, может быть одним из следующих:

- Значение curActiveLSN
- Значение, большее initialLSN
- FFFF FFFF FFFF, что интерпретируется программой асинхронного чтения журнала как конец текущего журнала.

Дополнительную информацию об информационной структуре чтения журнала смотрите в разделе “Структура данных: SQLU-RLOG-INFO” на стр. 388.

Прочитанные распространяемые записи журнала в диапазоне от начального до конечного LSN возвращаются в буфере журнала. В записи журнала не содержится ее LSN; он находится в буфере перед действительной записью журнала. Описания различных записей журнала DB2, возвращаемых **sqlurlog**, можно найти в руководстве *Administrative API Reference*.

Чтобы после первого чтения прочитать следующие последовательные записи журнала, прибавьте 1 к последнему LSN чтения, возвращенному в SQLU-RLOG-INFO. Повторите вызов с новым начальным LSN и действительным конечным LSN. Будет прочитан следующий блок записей. Код *sqlca* SQLU_RLOG_READ_TO_CURRENT означает, что программа чтения журнала выполнила чтение до конца текущего активного журнала.

Структура данных: db2HistData

Эта структура используется для возврата информации после вызова “db2HistoryGetEntry - API Получить следующую запись файла хронологии восстановлений” на стр. 360.

Таблица 17. Поля структуры db2HistData

Имя поля	Тип данных	Описание
ioHistDataID	char(8)	8-битный идентификатор структуры и маркер для дампов памяти. Единственное допустимое значение - “SQLUHINF”. Для этой строки нет символического определения.
oObjectPart	db2Char	Первые 14 символов - отметка времени начала операции в виде *ггггммддччннсс*. Следующие 3 символа - последовательный номер. Каждая операция резервного копирования может генерировать несколько записей в этом файле, если образ резервной копии сохраняется в нескольких файлах или на нескольких лентах. Последовательный номер позволяет указывать несколько положений. Для операций восстановления и загрузки в этом файле имеется единственная запись, соответствующая последовательному номеру ’001’ резервной копии. Отметка времени в сочетании с последовательным номером должна быть уникальной.
oEndTime	db2Char	Отметка времени завершения операции в виде *ггггммддччннсс*.
oFirstLog	db2Char	ID самого раннего из файлов журналов (в диапазоне от S0000000 до S9999999): • Требуется для восстановления с повтором транзакций оперативной резервной копии • Требуется для восстановления с повтором транзакций автономной резервной копии • Применяется после восстановления резервной копии полной базы данных или табличного пространства, которая была текущей в момент начала загрузки.

Таблица 17. Поля структуры db2HistData (продолжение)

Имя поля	Тип данных	Описание
oLastLog	db2Char	ID самого позднего из файлов журналов (в диапазоне от S0000000 до S9999999): • Требуется для восстановления с повтором транзакций оперативной резервной копии • Требуется для восстановления с повтором транзакций автономной резервной копии до текущего момента времени • Применяется после восстановления резервной копии полной базы данных или табличного пространства, которая была текущей в момент завершения загрузки (если повтор транзакций не применяется, совпадает с *oFirstLog*).
oID	db2Char	Уникальный идентификатор резервной копии или таблицы.
oTableQualifier	db2Char	Спецификатор таблицы
oTableName	db2Char	Имя таблицы.
oLocation	db2Char	Для резервных копий и копий загрузки это поле указывает, где сохраняются данные. Для операций, требующих нескольких записей в этом файле, последовательный номер, определяемый *oObjectPart*, указывает, какая часть резервной копии находится в указанном месте. Для операций восстановления и загрузки положение всегда указывает, где находится первая (соответствующая последовательному номеру '001' резервной копии из нескольких частей) часть восстанавливаемых или загружаемых данных. Данные в *oLocation* интерпретируются по-разному в зависимости от *oDeviceType*: • Для диска или дискеты (`D` или `K`) это полное имя файла. • Для ленты (`T`) это метка тома • Для TSM (`A`) это имя сервера • Для обработчика пользователя или устройства другого типа (`U` или`O`) это произвольный текст.
oComment	db2Char	Произвольный текстовый комментарий.
oCommandText	db2Char	Текст команды или DDL.
oLastLSN	SQLU_LSN	Последовательный номер последнего журнала.
oEID	Structure	Уникальный идентификатор записи.
poEventSQLCA	Structure	Полученная *sqlca* для записанного события. Информацию о структуре *sqlca* смотрите в книгах *Administrative API Reference* и *SQL Reference*.

Таблица 17. Поля структуры db2HistData (продолжение)

Имя поля	Тип данных	Описание
poTablespace	db2Char	Список имен табличных пространств.
ioNumTablespaces	db2Uint32	Число записей в списке *poTablespace*. Каждая резервная копия уровня табличного пространства содержит одно или несколько табличных пространств. При каждой операции восстановления табличных пространств заменяется одно или несколько табличных пространств. Если это поле содержит не ноль (что указывает на резервное копирование или восстановление уровня табличного пространства), следующие строки этого файла содержат имена копируемых или восстанавливаемых табличных пространств в виде 18-символьных строк, по строке на пространство. На каждой строке приводится имя одного табличного пространства.
oOperation	char	Смотрите Табл. 18.
oObject	char	Уровень операции: `D` - вся база данных, `P` - табличное пространство, `T` - таблица.
oOptype	char	Смотрите Табл. 19 на стр. 384.
oStatus	char	Статус записи: `D` - удаленная (для использования в будущем), `E` - с истекшим сроком, `I` - неактивная, `N` - еще не принятая, `Y` - принятая или активная.
oDeviceType	char	Тип устройства. Это поле определяет, как будет интерпретироваться поле *oLocation*: `A` - TSM, `C` - клиент, `D` - диск, `K` - дискета, `L` - локальный файл, `O` - прочее устройство (поддерживаемое другим поставщиком), `P` - конвейер, `S` - сервер, `T` - лента, `U` - обработчик пользователя.

Таблица 18. Допустимые значения oOperation в структуре db2HistData

Значение	Описание	Определение на C	Определение на COBOL/FORTRAN
A	добавление табличного пространства	DB2HISTORY_OP_ADD_ TABLESPACE	DB2HIST_OP_ADD_ TABLESPACE
B	резервное копирование	DB2HISTORY_OP_BACKUP	DB2HIST_OP_BACKUP
C	загрузка копии	DB2HISTORY_OP_LOAD_COPY	DB2HIST_OP_LOAD_COPY
D	отбрасывание таблицы	DB2HISTORY_OP_DROPPED_ TABLE	DB2HIST_OP_DROPPED_TABLE
F	повтор транзакций	DB2HISTORY_OP_ROLLFWD	DB2HIST_OP_ROLLFWD
G	реорганизация таблицы	DB2HISTORY_OP_REORG	DB2HIST_OP_REORG

Таблица 18. Допустимые значения oOperation в структуре db2HistData (продолжение)

Значение	Описание	Определение на C	Определение на COBOL/FORTRAN
L	загрузка	DB2HISTORY_OP_LOAD	DB2HIST_OP_LOAD
N	переименование табличного пространства	DB2HISTORY_OP_REN_ TABLESPACE	DB2HIST_OP_REN_ TABLESPACE
O	отбрасывание табличного пространства	DB2HISTORY_OP_DROP_ TABLESPACE	DB2HIST_OP_DROP_ TABLESPACE
Q	стабилизация	DB2HISTORY_OP_QUIESCE	DB2HIST_OP_QUIESCE
R	восстановление	DB2HISTORY_OP_RESTORE	DB2HIST_OP_RESTORE
S	запуск статистики	DB2HISTORY_OP_RUNSTATS	DB2HIST_OP_RUNSTATS
T	изменение табличного пространства	DB2HISTORY_OP_ALT_ TABLESPACE	DB2HIST_OP_ALT_TBS
U	выгрузка	DB2HISTORY_OP_UNLOAD	DB2HIST_OP_UNLOAD

Таблица 19. Допустимые значения oOptype в структуре db2HistData

oOperation	oOptype	Описание	Определение C/COBOL/FORTRAN
B	F	автономная	DB2HISTORY_OPTYPE_OFFLINE
	N	оперативная	DB2HISTORY_OPTYPE_ONLINE
	I	инкрементная автономная	DB2HISTORY_OPTYPE_INCR_ OFFLINE
	O	инкрементная оперативная	DB2HISTORY_OPTYPE_INCR_ ONLINE
	D	разностная автономная	DB2HISTORY_OPTYPE_DELTA_ OFFLINE
	E	разностная оперативная	DB2HISTORY_OPTYPE_DELTA_ ONLINE
F	E	до конца журналов	DB2HISTORY_OPTYPE_EOL
	P	до момента времени	DB2HISTORY_OPTYPE_PIT
L	I	вставка	DB2HISTORY_OPTYPE_INSERT
	R	замена	DB2HISTORY_OPTYPE_REPLACE

Таблица 19. Допустимые значения oOptype в структуре db2HistData (продолжение)

oOperation	oOptype	Описание	Определение C/COBOL/FORTRAN
Q	S	стабилизация в совместном режиме	DB2HISTORY_OPTYPE_SHARE
	U	изменение стабилизации	DB2HISTORY_OPTYPE_UPDATE
	X	стабилизация в монопольном режиме	DB2HISTORY_OPTYPE_EXCL
	Z	сброс стабилизации	DB2HISTORY_OPTYPE_RESET
R	F	автономная	DB2HISTORY_OPTYPE_OFFLINE
	N	оперативная	DB2HISTORY_OPTYPE_ONLINE
	I	инкрементная автономная	DB2HISTORY_OPTYPE_INCR_ OFFLINE
	O	инкрементная оперативная	DB2HISTORY_OPTYPE_INCR_ ONLINE
T	C	добавление контейнеров	DB2HISTORY_OPTYPE_ADD_CONT
	R	перебалансировка	DB2HISTORY_OPTYPE_REB

Таблица 20. Поля структуры db2Char

Имя поля	Тип данных	Описание
pioData	char	Указатель на буфер символьных данных. Если он пуст (NULL), данные не будут возвращены.
iLength	db2Uint32	Входной. Размер буфера *pioData*.
oLength	db2Uint32	Выходной. Число допустимых символов данных в буфере *pioData*.

Таблица 21. Поля структуры db2HistoryEID

Имя поля	Тип данных	Описание
ioNode	SQL_PDB_NODE_TYPE	Номер узла.
ioHID	db2Uint32	ID записи локального файла хронологии.

Структура данных: db2HistData

Синтаксис языка

Структура на языке C

```
/* Файл: db2ApiDf.h */
/* ... */
typedef SQL_STRUCTURE db2HistoryData
{
  char ioHistDataID[8];
  db2Char oObjectPart;
  db2Char oEndTime;
  db2Char oFirstLog;
  db2Char oLastLog;
  db2Char oID;
  db2Char oTableQualifier;
  db2Char oTableName;
  db2Char oLocation;
  db2Char oComment;
  db2Char oCommandText;
  SQLU_LSN oLastLSN;
  db2HistoryEID oEID;
  struct sqlca * poEventSQLCA;
  db2Char * poTablespace;
  db2Uint32 ioNumTablespaces;
  char oOperation;
  char oObject;
  char oOptype;
  char oStatus;
  char oDeviceType
} db2HistoryData;

typedef SQL_STRUCTURE db2Char
{
  char * pioData;
  db2Uint32 ioLength
} db2Char;

typedef SQL_STRUCTURE db2HistoryEID
{
  SQL_PDB_NODE_TYPE ioNode;
  db2Uint32 ioHID
} db2HistoryEID;
/* ... */
```

Структура данных: SQLU-LSN

Это объединение, которое использует “sqlurlog - API асинхронное чтение журнала” на стр. 378, содержит определение номера последовательности файлов журнала. Номер последовательности файлов журнала (log sequence number - LSN) представляет собой относительный адрес в байтах в журнале базы данных. Эти числа идентифицируют каждую запись журнала. Они представляют собой смещение в байтах относительно начала журнала базы данных.

Таблица 22. Поля в объединении SQLU-LSN

Имя поля	Тип данных	Описание
lsnChar	Массив UNSIGNED CHAR	Задает 6-членный последовательный номер журнала для символьного массива.
lsnWord	Массив для UNSIGNED SHORT	Задает краткий 3-членный последовательный номер журнала для массива.

Синтаксис языка

Структура на языке C

```
typedef union SQLU_LSN
{
unsigned char  lsnChar  [6] ;
unsigned short lsnWord  [3] ;
} SQLU_LSN;
```

Структура данных: SQLU-RLOG-INFO

Эта структура содержит информацию о состоянии вызовов асинхронного чтения журнала “sqlurlog - API асинхронное чтение журнала” на стр. 378, и журнала базы данных.

Таблица 23. Поля структуры SQLU-RLOG-INFO

Имя поля	Тип данных	Описание
initialLSN	SQLU_LSN	Указывает значение LSN первой записи журнала, сделанной после первого использования оператора CONNECT для базы данных. Дополнительную информацию смотрите в разделе “Структура данных: SQLU-LSN” на стр. 387.
firstReadLSN	SQLU_LSN	Указывает значение LSN первой прочитанной записи журнала.
lastReadLSN	SQLU_LSN	Указывает значение LSN последней прочитанной записи журнала.
curActiveLSN	SQLU_LSN	Указывает значение LSN текущего (активного) журнала.
logRecsWritten	sqluint32	Указывает число записей журнала, записанных в буфер.
logBytesWritten	sqluint32	Указывает число байтов, записанных в буфер.

Синтаксис языка

Структура на языке C

```
typedef SQL_STRUCTURE SQLU_RLOG_INFO
{
SQLU_LSN       initialLSN ;
SQLU_LSN       firstReadLSN ;
SQLU_LSN       lastReadLSN ;
SQLU_LSN       curActiveLSN ;
sqluint32      logRecsWritten ;
sqluint32      logBytesWritten ;
} SQLU_RLOG_INFO;
```

Приложение E. Программы примеров восстановления

Пример программы без встроенного SQL (backrest.c)

В следующей программе примера показано, как использовать API резервного копирования и восстановления DB2 для:

- Резервного копирования базы данных
- Восстановления базы данных
- Восстановления базы данных с повтором транзакций

Подробную информацию о базе данных SAMPLE смотрите в справочнике *SQL Reference*.

Исходный файл данной программы примера (`backrest.c`) в операционных системах Windows и OS/2 расположен в каталоге `\sqllib\samples\c`, а в системах на основе UNIX - в каталоге `/sqllib/samples/c`. Он содержит ряд API DB2. Файл `makefile` в том же каталоге содержит команды для построения этой и других программ примеров. Общую информацию о создании программ, содержащих API управления DB2, и подробную информацию об опциях компилирования и компоновки можно найти в книге *Application Building Guide*. Чтобы построить программу примера `backrest` из исходного файла `backrest.c` в Windows NT или Windows 2000:

1. Скопируйте файлы `backrest.c`, `makefile`, `utilapi.c` и `utilapi.h` в рабочий каталог.
2. Если менеджер баз данных не запущен, введите из командного окна DB2 команду **db2start**. Чтобы открыть окно процессора командной строки DB2 и инициализировать среду командной строки DB2, введите в командной строке операционной системы Windows команду **db2cmd**.
3. Введите команду `nmake backrest`. Будут сгенерированы следующие файлы:

```
backrest.exe
backrest.ilk
backrest.obj
backrest.pdb
utilapi.obj
```

Чтобы запустить программу примера (выполняемый файл), введите:

```
backrest база_данных ID_пользователя пароль
```

Например:

```
backrest sample db2admin db2admin
```

Вот пример вывода, возвращаемого данной программой:

© Copyright IBM Corp. 2001

Пример программы без встроенного SQL (backrest.c)

```
This is sample program : backrest.c

NOTE:   Ensure the database is not in use prior to running this program.

Updating the sample database configuration parameter LOGRETAIN
to 'ON' to enable rollforward recovery.

Backing up the 'sample' database.
The database has been successfully backed up.

Updating the sample database configuration parameter LOGRETAIN
to 'OFF'.

Restoring the database 'sample' as 'TESTBACK' (1st pass).
Should get returned value = SQLUD_INACCESSABLE_CONTAINER.
SQLUD_INACCESSABLE_CONTAINER is returned.

Need to SET TABLESPACES CONTAINERS
Tablespace container information for tablespace 1 obtained.
Tablespace containers have been set for tablespace 1.

Restoring the database 'sample' as 'TESTBACK' (2nd pass).
Database sample has been restored as 'TESTBACK'.

The database 'TESTBACK' has been successfully rolled forward.

The database 'TESTBACK' has been successfully dropped.
```

Исходный текст программы примера:

```
/*****************************************************************************
**
** Имя исходного файла = backrest.c
**
** Лицензионные материалы - собственность IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1995, 2000
** Все права защищены.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**
**    ЦЕЛЬ :
**       пример показывает, как использовать API резервного копирования и
**       восстановления для:
**          - резервного копирования базы данных
**          - восстановления базы данных
**          - повтора транзакций базы данных для возврата в состояние,
**            предшествующее повреждению...
**
**    ИСПОЛЬЗУЕМЫЕ API :
**       РЕЗЕРВНОЕ КОПИРОВАНИЕ БАЗЫ ДАННЫХ                       sqlubkp()
**       ВОССТАНОВЛЕНИЕ БАЗЫ ДАННЫХ                              sqlurestore()
**       ИЗМЕНЕНИЕ КОНФИГУРАЦИИ БАЗЫ ДАННЫХ                      sqlfudb()
**       ЗАПРОС ПОЛУЧЕНИЯ КОНТЕЙНЕРОВ ТАБЛИЧНОГО ПРОСТРАНСТВА    sqlbtcq()
```

```
**       ЗАДАНИЕ КОНТЕЙНЕРОВ ТАБЛИЧНОГО ПРОСТРАНСТВА              sqlbstsc()
**       ПОВТОР ТРАНЗАКЦИЙ БАЗЫ ДАННЫХ                            sqluroll()
**
**    ИСПОЛЬЗУЕМЫЕ СТРУКТУРЫ :
**       sqlu_tablespace_bkrst_list
**       SQLB_TBSCONTQRY_DATA
**       sqlu_media_list
**       rfwd_input
**       rfwd_output
**       sqlurf_info
**       sqlca
**
**    ДРУГИЕ ОБЪЯВЛЕННЫЕ ФУНКЦИИ :
**       БИБЛИОТЕКА КОМПИЛЯТОРА 'C' :
**          stdio.h  -  printf
**
**       встроенные :
**
**       внешние :
**          check_error :     Проверяет ошибку SQLCODE и выводит
**          [в файле UTIL.C] всю соответствующую доступную информацию.
**                               Эта процедура содержится в файле UTIL.C.
**
**    ВНЕШНИЕ ЗАВИСИМОСТИ :
**       - Проверьте существование базы данных для целей прекомпиляции.
**       - Для компиляции и компоновки используйте компилятор IBM Cset++
**         (AIX и OS/2), компилятор Microsoft Visual C++ (Windows)
**         или компилятор, поддерживаемый на вашей платформе.
**
** Дополнительную информацию об этих примерах смотрите в файле README.
**
** Дополнительную информацию о программировании на языке C смотрите в разделе
**   -  "Programming in C and C++" руководства Application Development Guide
** Дополнительную информацию о разработке программ на языке C смотрите в разделе
**   -  "Building C Applications" руководства Application Building Guide.
**
** Дополнительную информацию о языке SQL смотрите в справочнике SQL Reference.
**
******************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlutil.h>
#include <sqlenv.h>
#include "utilapi.h"

/* Можно заменить каталог другим, на который */
/* у вас есть разрешение. Создайте подкаталог "backup" в том же каталоге. */

   #define TEMPDIR "."

int main( int argc, char * argv[] ) {

   /* Переменные для API резервного копирования */
```

Пример программы без встроенного SQL (backrest.c)

```
 sqluint32 buff_size;
 sqluint32 num_buff;
 struct sqlu_tablespace_bkrst_list *tablespace_list;
 struct SQLB_TBSCONTQRY_DATA *cont;
struct sqlca sqlca;
 struct sqlu_media_entry media_entry;
 struct sqlu_media_list media_list;
 char applid[SQLU_APPLID_LEN+1];
 char timestamp[SQLU_TIME_STAMP_LEN+1];
 char *target_path;

 sqluint32 tsc_id;
 sqluint32 num_cont;

 /* переменные для API изменения конфигурации базы данных */
 struct sqlfupd itemList;
 short log_retain;

 /* переменные для API повтора транзакций */
 struct rfwd_input rfwdInput;
 char dbAlias[] = "TESTBACK";
 char overFlowLogPath[SQL_PATH_SZ] = TEMPDIR;

 struct rfwd_output rfwdOutput;
 sqlint32   numReplies;
 struct sqlurf_info nodeInfo;

 printf ("\nThis is sample program : backrest.c\n\n");

 printf ("NOTE:   Ensure the database is not in use prior to running
  this program.\n\n");

 if (argc != 4) {
    printf ("\nUSAGE: backrest database userID password\n\n");
    return 1;
 } /* endif */

 /* Изменение конфигурации базы данных по умолчанию для разрешения восстановления
    с повтором транзакций базы данных TESTBACK. */

 log_retain = 1;
 itemList.token = SQLF_DBTN_LOG_RETAIN;

 itemList.ptrvalue = (char *) &log_retain;


 printf ("Updating the %s database configuration parameter
  LOGRETAIN \n", argv[1]);
 printf ("to 'ON' to enable rollforward recovery.\n\n");

 /*****************************************\
 * Вызываемые API ИЗМЕНЕНИЯ КОНФИГУРАЦИИ БАЗЫ ДАННЫХ *
 \*****************************************/
 sqlfudb(argv[1], 1, &itemList, &sqlca);
 API_SQL_CHECK("updating the database configuration");
```

```
/* Инициализация переменных API резервного копирования */
 buff_size = 16;
 num_buff = 1;
 tablespace_list = NULL;
 media_list.media_type = 'L';
 media_list.sessions = 1;

 strcpy (media_entry.media_entry, TEMPDIR);

 media_list.target.media = &media_entry;
 printf ("Backing up the '%s' database.\n", argv[1]);
 /*****************\
 * РЕЗЕРВНОЕ КОПИРОВАНИЕ БАЗЫ ДАННЫХ *
 \*****************/
 sqlubkp (argv[1],
           buff_size,
           SQLUB_OFFLINE,
           SQLUB_FULL,
           SQLUB_BACKUP,
           applid,
           timestamp,
           num_buff,
           tablespace_list,
           &media_list,
           argv[2],
           argv[3],
           NULL,
           0,
           NULL,
           1,
           NULL,
           NULL,
           NULL,
           &sqlca);

 API_SQL_CHECK("backing up the database");
 printf ("The database has been successfully backed up.\n\n");



 /* Изменение конфигурации базы данных по умолчанию для запрета восстановления
    с повтором транзакций базы данных TESTBACK. */

 log_retain = 0;
 itemList.token = SQLF_DBTN_LOG_RETAIN;
 itemList.ptrvalue = (char *) &log_retain;


 printf ("Updating the %s database configuration parameter
  LOGRETAIN \n", argv[1]);
 printf ("to 'OFF'.\n\n");
```

Пример программы без встроенного SQL (backrest.c)

```
/*****************************************\
* Вызываемые API ИЗМЕНЕНИЯ КОНФИГУРАЦИИ БАЗЫ ДАННЫХ *
\*****************************************/
sqlfudb(argv[1], 1, &itemList, &sqlca);
API_SQL_CHECK("updating the database configuration");



/* Инициализация переменных для API восстановления */
buff_size = 1024;
target_path = NULL;
printf ("Restoring the database '%s' as 'TESTBACK' (1st pass).\n", argv[1]);
printf ("Should get returned value = SQLUD_INACCESSABLE_CONTAINER.\n");
/******************\
* ВОССТАНОВЛЕНИЕ БАЗЫ ДАННЫХ *
\******************/
sqlurestore(argv[1],
          "TESTBACK",
          buff_size,
          SQLUD_ROLLFWD,
          SQLUD_DATALINK,
          SQLUD_FULL,
          SQLUD_OFFLINE,
          SQLUD_RESTORE_STORDEF,
          applid,
          timestamp,
          target_path,
          num_buff,
          NULL,
          tablespace_list,
          &media_list,
          argv[2],
          argv[3],
          NULL,
          0,
          NULL,
          1,
          NULL,
          NULL,
          NULL,
          &sqlca);
if (sqlca.sqlcode != SQLUD_INACCESSABLE_CONTAINER)
   API_SQL_CHECK("restoring database");

printf ("SQLUD_INACCESSABLE_CONTAINER is returned.\n\n");
printf ("Need to SET TABLESPACES CONTAINERS\n");
/* Задание структуры контейнеров для нового списка контейнеров */
tsc_id = 1;
cont = (struct SQLB_TBSCONTQRY_DATA *) malloc
   (sizeof(struct SQLB_TBSCONTQRY_DATA));

/*******************************\
* ПОЛУЧЕНИЕ ЗАПРОСА КОНТЕЙНЕРОВ ТАБЛИЧНОГО ПРОСТРАНСТВА *
\*******************************/
```

```
sqlbtcq (&sqlca, tsc_id, &num_cont, &cont);
API_SQL_CHECK("GET TABLESPACE CONTAINER QUERY");
printf ("Tablespace container information for tablespace 1 obtained.\n");
/*************************\
* ЗАДАНИЕ КОНТЕЙНЕРОВ ТАБЛИЧНОГО ПРОСТРАНСТВА *
\*************************/
sqlbstsc (&sqlca,
          SQLB_SET_CONT_INIT_STATE,
          tsc_id,
          num_cont,
          cont);
API_SQL_CHECK("SET TABLESPACE CONTAINERS");
printf ("Tablespace containers have been set for tablespace 1.\n\n");

printf ("Restoring the database '%s' as 'TESTBACK' (2nd pass).\n", argv[1]);
/******************\
* ВОССТАНОВЛЕНИЕ БАЗЫ ДАННЫХ *
\******************/
sqlurestore (argv[1],
          "TESTBACK",
          buff_size,
          SQLUD_ROLLFWD,
          SQLUD_DATALINK,
          SQLUD_FULL,
          SQLUD_OFFLINE,
          SQLUD_CONTINUE,
          applid,
          timestamp,
          target_path,
          num_buff,
          NULL,
          tablespace_list,
          &media_list,
          argv[2],
          argv[3],
          NULL,
          0,
          NULL,
          1,
          NULL,
          NULL,
          NULL,
          &sqlca);
API_SQL_CHECK("restoring database 2");
printf ("Database %s has been restored as 'TESTBACK'.\n\n", argv[1]);


/**********************\
*  ПОВТОР ТРАНЗАКЦИЙ БАЗЫ ДАННЫХ *
\**********************/
rfwdInput.version=SQLUM_RFWD_VERSION;
rfwdInput.pDbAlias=dbAlias;
rfwdInput.CallerAction=SQLUM_ROLLFWD;
rfwdInput.pStopTime=SQLUM_INFINITY_TIMESTAMP;
rfwdInput.pUserName=argv[2];
```

Пример программы без встроенного SQL (backrest.c)

```
    rfwdInput.pPassword=argv[3];
    rfwdInput.pOverflowLogPath=overFlowLogPath;
    rfwdInput.NumChngLgOvrflw=0;
    rfwdInput.pChngLogOvrflw=NULL;
    rfwdInput.ConnectMode=SQLUM_OFFLINE;
    rfwdInput.pTablespaceList=NULL;
    rfwdInput.AllNodeFlag= SQLURF_ALL_NODES;
    rfwdInput.NumNodes=0;
    rfwdInput.pNodeList=NULL;
    rfwdInput.NumNodeInfo=1;
    rfwdInput.DlMode=0;
    rfwdInput.pReportFile=NULL;
    rfwdInput.pDroppedTblID=NULL;
    rfwdInput.pExportDir=NULL;


    rfwdOutput.pApplicationId=applid;
    rfwdOutput.pNumReplies=&numReplies;
    rfwdOutput.pNodeInfo=&nodeInfo;


    sqluroll( &rfwdInput,
              &rfwdOutput,
              &sqlca);
    API_SQL_CHECK("rolling database forward");
    printf ("The database 'TESTBACK' has been successfully rolled
     forward.\n\n", argv[1]);


    /***************\
    * ОТБРАСЫВАНИЕ БАЗЫ ДАННЫХ *
    \***************/
    sqledrpd ("TESTBACK",
              &sqlca);
    API_SQL_CHECK("DROP DATABASE");
    printf ("The database 'TESTBACK' has been successfully dropped.\n");


  return 0;
}
```

Программа примера со встроенным SQL (dbrecov.sqc)

В следующей программе примера показано, как использовать API резервного копирования и восстановления DB2 для:

- Резервного копирования базы данных
- Восстановления базы данных
- Восстановления базы данных с повтором транзакций

Подробную информацию о базе данных SAMPLE смотрите в справочнике *SQL Reference*.

Примечание: Файлы примеров dbrecov не устанавливаются с DB2 Версии 7.2. Загрузить их можно только из Web.

Чтобы загрузить необходимые файлы для программы примера dbrecov:

1. Зайдите на страницу http://www.ibm.com/software/data/db2/udb/ad/v7/abg.html.
2. На странице “Recent Updates” выберите “dbrecov sample files by platform”.
3. На появившейся странице выберите ссылку для загрузки файлов, соответствующих вашей платформе. Для операционных систем Windows и OS/2 эти файлы содержатся в самораспаковывающемся выполняемом файле; для систем на основе UNIX файлы содержатся в файле tar.gz.

ОСТОРОЖНО:
Не запускайте этот выполняемый файл в каталоге, где находятся файлы примеров DB2 UDB Версии 7, так как существующие файлы утилит DB2 UDB Версии 7 будут заменены новыми файлами утилит. Новые файлы утилит несовместимы с другими программами примеров.

Файл источника этой программы примера (dbrecov.sqc) содержит вызовы API и встроенного SQL. Общую информацию о создании программ, содержащих API управления DB2, и подробную информацию об опциях компилирования и компоновки можно найти в книге *Application Building Guide*.

Ниже приводятся инструкции для построения и запуска программы dbrecov в операционной системе Windows. Как построить эту программу в других поддерживаемых операционных системах, описано в файле README, который поставляется с файлами примеров.

1. Запустите загруженный выполняемый файл dbrecov_win.exe в своем рабочем каталоге. При этом будут извлечены следующие файлы:

```
dbrecov.sqc - Исходный файл встроенного SQL для программы dbrecov
utilapi.c   - Файл утилиты проверки ошибок для программ API DB2
utilapi.h   - Файл заголовков для utilapi.c
utilemb.sqc - Файл утилиты проверки ошибок для программ встроенного SQL
utilemb.h   - Файл заголовков для utilemb.sqc
bldmapp.bat - Файл построения программ Microsoft Visual C++
bldvapp.bat - Файл построения программ VisualAge C++
embprep.bat - Прекомпилирование и связывание программ встроенного SQL
README      - Содержит информацию о файлах программ
```

2. Если менеджер баз данных не запущен, введите из командного окна DB2 команду **db2start**. Чтобы открыть окно процессора командной строки DB2 и инициализировать среду командной строки DB2, введите в командной строке операционной системы Windows команду **db2cmd**.
3. Введите команду bldmapp dbrecov или bldvapp dbrecov в зависимости от вашего компилятора. Будут сгенерированы следующие файлы:

```
dbrecov.exe
dbrecov.ilk
dbrecov.c
dbrecov.obj
dbrecov.bnd
dbrecov.pdb
utilemb.c
utilemb.obj
```

Чтобы запустить программу примера (выполняемый файл), введите:

```
dbrecov
```

Вывод будет отличаться в разных операционных средах. Вот пример вывода, возвращаемого данной программой в системе Windows:

```
THIS SAMPLE SHOWS HOW TO RECOVER A DATABASE.

USE THE DB2 API:
  sqlfxdb -- Get Database Configuration
TO GET THE DATABASE CONFIGURATION AND DETERMINE
THE SERVER WORKING PATH.

NOTE: Backup images will be created on the server
      in the directory D:\DB2,
      and will not be deleted by the program.

**************************************
*** PRUNE THE RECOVERY HISTORY FILE ***
**************************************

USE THE DB2 API:
  db2Prune -- Prune Recovery History File
AND THE SQL STATEMENTS:
  CONNECT
  CONNECT RESET
TO PRUNE THE RECOVERY HISTORY FILE.

  Connecting to 'sample' database...
  Connected to 'sample' database.

  Prune the recovery history file for 'sample' database.

  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.

**************************************
*** BACK UP AND RESTORE A DATABASE ***
**************************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlurestore -- Restore Database
TO BACK UP AND RESTORE A DATABASE.
```

```
  Update 'sample' database configuration:
    - Disable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = OFF/NO

  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162032

  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162032
    - target database        : sample

-- The following warning report is expected! --
---- warning report ---------------------------

application message = database restore -- start
line                = 506
file                = dbrecov.sqc
SQLCODE             = 2539

SQL2539W  Предупреждение!  Восстанавливается существующая база, которая совпадает
с базой резервной копии. Файлы базы данных будут удалены.

---- end warning report -----------------------

  Continuing the restore operation...

  Restore finished.

*************************
*** REDIRECTED RESTORE ***
*************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlecrea -- Create Database
  sqlurestore -- Restore Database
  sqlbmtsq -- Tablespace Query
  sqlbtcq -- Tablespace Container Query
  sqlbstsc -- Set Tablespace Containers
  sqlefmem -- Free Memory
  sqledrpd -- Drop Database
TO BACK UP AND DO A REDIRECTED RESTORE OF A DATABASE.

  Update 'sample' database configuration:
    - Disable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = OFF/NO

  Backing up the 'sample' database...
  Backup finished.
```

```
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162125

  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162125
    - target database        : RRDB

-- The following warning report is expected! --
---- warning report ---------------------------

application message = database restore -- start
line                = 776
file                = dbrecov.sqc
SQLCODE             = 1277

SQL1277W  При восстановлении в табличных пространствах обнаружен один или
несколько недоступных контейнеров, или контейнеров в состоянии с неопределенным
размером.

---- end warning report ----------------------

  Continuing the restore operation...

  Find and redefine inaccessible containers.

    Redefine inaccessible container:
      - table space name: SYSCATSPACE
      - default container name: D:\DB2\NODE0000\SQL00003\SQLT0000.0
      - container type: path
      - new container name: D:\DB2\SQLT0000.0

    Redefine inaccessible container:
      - table space name: TEMPSPACE1
      - default container name: D:\DB2\NODE0000\SQL00003\SQLT0001.0
      - container type: path
      - new container name: D:\DB2\SQLT0001.0

    Redefine inaccessible container:
      - table space name: USERSPACE1
      - default container name: D:\DB2\NODE0000\SQL00003\SQLT0002.0
      - container type: path
      - new container name: D:\DB2\SQLT0002.0

  Restore finished.

  Drop the 'RRDB' database.

****************************
*** ROLLFORWARD RECOVERY ***
****************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
```

```
  sqlubkp -- Backup Database
  sqlecrea -- Create Database
  sqlurestore -- Restore Database
  sqluroll -- Rollforward Database
  sqledrpd -- Drop Database
TO BACK UP, RESTORE, AND ROLL A DATABASE FORWARD.

  Update 'sample' database configuration:
    - Enable the configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = RECOVERY/YES

  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162223

  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162223
    - target database        : RFDB

  Restore finished.

  Rolling 'RFDB' database forward ...
  Rollforward finished.

  Drop the 'RFDB' database.

*****************************
*** ASYNCHRONOUS READ LOG ***
*****************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlurlog -- Asynchronous Read Log
AND THE SQL STATEMENTS:
  CONNECT
  ALTER TABLE
  COMMIT
  INSERT
  DELETE
  ROLLBACK
  CONNECT RESET
TO READ LOG RECORDS FOR THE CURRENT CONNECTION.

  Update 'sample' database configuration:
    - Enable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = RECOVERY/YES

  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
```

```
    - backup image time stamp: 20010506162247

  Connecting to 'sample' database...
  Connected to 'sample' database.

  Invoke the following SQL statements:
    ALTER TABLE emp_resume DATA CAPTURE CHANGES;
    COMMIT;
    INSERT INTO emp_resume
      VALUES('000777', 'ascii', 'This is a new resume.');
          ('777777', 'ascii', 'This is another new resume');
    COMMIT;
    DELETE FROM emp_resume WHERE empno = '000777';
    DELETE FROM emp_resume WHERE empno = '777777';
    COMMIT;
    DELETE FROM emp_resume WHERE empno = '000140';
    ROLLBACK;
    ALTER TABLE emp_resume DATA CAPTURE NONE;
    COMMIT;

  Start reading database log.

-- The following warning report is expected! --
---- warning report ---------------------------

application message = database log info -- get
line               = 1457
file               = dbrecov.sqc
SQLCODE            = 2653

SQL2653W  При восстановлении, восстановлении с повтором или
аварийном восстановлении могли повторно использоваться диапазоны
последовательных номеров. Код причины "1".
---- end warning report ----------------------

    Record type: Normal
      component ID: DMS log record
      function ID:  Alter Table Attribute
        Propagation attribute is changed to: ON

    Record type: Normal
      component ID: DMS log record
      function ID: Update Record
        oldRID: 2
        old subrecord length: 76
        old subrecord offset: 0
        subrecord type: Updatable, Internal control
        newRID: 2
        new subrecord length: 76
        new subrecord offset: 2916
        subrecord type: Updatable, Internal control

    Record type: Local pending list
      UTC transaction committed (in seconds since 01/01/70): 989180591
      authorization ID of the application: MELNYK
```

```
Record type: Normal
  component ID: DMS log record
  function ID: Insert Record
    RID: 12
    subrecord length: 88
    subrecord offset: 486
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 37 37 37 0F 00 05 00 *000777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 15 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 15 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*

Record type: Normal
  component ID: DMS log record
  function ID: Insert Record
    RID: 13
    subrecord length: 88
    subrecord offset: 398
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    37 37 37 37 37 37 0F 00 05 00 *777777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 1A 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 1A 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 01 00 00 00 *..........*

Record type: Normal commit
  UTC transaction committed (in seconds since 01/01/70): 989180591
  authorization ID of the application: MELNYK

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 12
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 37 37 37 0F 00 05 00 *000777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 15 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 15 00 *..........*
```

```
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 13
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    37 37 37 37 37 37 0F 00 05 00 *777777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 1A 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 1A 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 01 00 00 00 *..........*

Record type: Normal commit
  UTC transaction committed (in seconds since 01/01/70): 989180591
  authorization ID of the application: MELNYK

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 6
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 05 00 *000140....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 24 05 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 01 3C 00 02 00 00 00 24 05 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 14 00 00 00 *..........*

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 7
    subrecord length: 89
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 06 00 *000140....*
    15 00 3C 00 00 73 63 72 69 70 *.....scrip*
```

```
    74 49 06 01 00 00 00 00 00 56 *tI.......V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 01 3C 00 02 00 00 00 56 *.........V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 16 00 00 *..........*
    00                            *.         *

Record type: Compensation
  component ID: DMS log record
  function ID: Undo Delete Record
    RID: 7
    subrecord length: 89
    subrecord offset: 397
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 06 00 *000140....*
    15 00 3C 00 00 73 63 72 69 70 *.....scrip*
    74 49 06 01 00 00 00 00 00 56 *tI.......V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 01 3C 00 02 00 00 00 56 *.........V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 16 00 00 *..........*
    00                            *.         *

Record type: Compensation
  component ID: DMS log record
  function ID: Undo Delete Record
    RID: 6
    subrecord length: 88
    subrecord offset: 309
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 05 00 *000140....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 24 05 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 01 3C 00 02 00 00 00 24 05 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 14 00 00 00 *..........*

Record type: Normal abort
  authorization ID of the application: MELNYK

Record type: Normal
  component ID: DMS log record
  function ID:  Alter Table Attribute
    Propagation attribute is changed to: OFF

Record type: Local pending list
  UTC transaction committed (in seconds since 01/01/70): 989180591
```

```
      authorization ID of the application: MELNYK

  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.

*******************************************
*** READ A DATABASE RECOVERY HISTORY FILE ***
*******************************************

USE THE DB2 APIs:
  db2HistoryOpenScan -- Open Recovery History File Scan
  db2HistoryGetEntry -- Get Next Recovery History File Entry
  db2HistoryCloseScan -- Close Recovery History File Scan
TO READ A DATABASE RECOVERY HISTORY FILE.

  Open recovery history file for 'sample' database.

  Read entry number 0.

  Display entry number 0.
    object part: 20010506162032001
    end time: 200105061620
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
    history file entry ID: 48
    table spaces:
      SYSCATSPACE
      USERSPACE1
    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Read entry number 1.

  Display entry number 1.
    object part: 20010506162058001
    end time: 200105061621
    first log: S0000000
    last log: S0000000
    ID: 20010506162032
    table qualifier:
    table name:
    location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
```

```
  comment: DB2 RESTORE SAMPLE NO RF
  command text:
  history file entry ID: 49
  table spaces:
    SYSCATSPACE
    USERSPACE1
  type of operation: R
  granularity of the operation: D
  operation type: F
  entry status: A
  device type: D
  SQLCA:
    sqlcode: 0
    sqlstate:
    message:

Read entry number 2.

Display entry number 2.
  object part: 20010506162125001
  end time: 200105061622
  first log: S0000000
  last log: S0000000
  ID:
  table qualifier:
  table name:
  location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
  comment: DB2 BACKUP SAMPLE OFFLINE
  command text:
  history file entry ID: 50
  table spaces:
    SYSCATSPACE
    USERSPACE1
  type of operation: B
  granularity of the operation: D
  operation type: F
  entry status: A
  device type: D
  SQLCA:
    sqlcode: 0
    sqlstate:
    message:

Read entry number 3.

Display entry number 3.
  object part: 20010506162223001
  end time: 200105061622
  first log: S0000000
  last log: S0000000
  ID:
  table qualifier:
  table name:
  location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
  comment: DB2 BACKUP SAMPLE OFFLINE
```

```
    command text:
    history file entry ID: 51
    table spaces:
      SYSCATSPACE
      USERSPACE1
    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Read entry number 4.

  Display entry number 4.
    object part: 20010506162247001
    end time: 200105061623
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
    history file entry ID: 52
    table spaces:
      SYSCATSPACE
      USERSPACE1
    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Close recovery history file for 'sample' database.

******************************************************
*** UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY ***
******************************************************

USE THE DB2 APIs:
  db2HistoryOpenScan -- Open Recovery History File Scan
  db2HistoryGetEntry -- Get Next Recovery History File Entry
  db2HistoryUpdate -- Update Recovery History File
  db2HistoryCloseScan -- Close Recovery History File Scan
TO UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY.
```

```
Open the recovery history file for 'sample' database.

Read the first entry in the recovery history file.

Display the first entry.
  object part: 20010506162032001
  end time: 200105061620
  first log: S0000000
  last log: S0000000
  ID:
  table qualifier:
  table name:
  location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
  comment: DB2 BACKUP SAMPLE OFFLINE
  command text:
  history file entry ID: 48
  table spaces:
    SYSCATSPACE
    USERSPACE1
  type of operation: B
  granularity of the operation: D
  operation type: F
  entry status: A
  device type: D
  SQLCA:
    sqlcode: 0
    sqlstate:
    message:

Connecting to 'sample' database...
Connected to 'sample' database.

Update the first entry in the history file:
  new location = 'this is the NEW LOCATION'
  new comment = 'this is the NEW COMMENT'

Disconnecting from 'sample' database...
Disconnected from 'sample' database.

Close recovery history file for 'sample' database.

Read the first recovery history file entry.

Display the first entry.
  object part: 20010506162032001
  end time: 200105061620
  first log: S0000000
  last log: S0000000
  ID:
  table qualifier:
  table name:
  location: this is the NEW LOCATION
  comment: this is the NEW COMMENT
  command text:
  history file entry ID: 48
```

```
    table spaces:
      SYSCATSPACE
      USERSPACE1
    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Close the recovery history file for 'sample' database.

**************************************
*** PRUNE THE RECOVERY HISTORY FILE ***
**************************************

USE THE DB2 API:
  db2Prune -- Prune Recovery History File
AND THE SQL STATEMENTS:
  CONNECT
  CONNECT RESET
TO PRUNE THE RECOVERY HISTORY FILE.

  Connecting to 'sample' database...
  Connected to 'sample' database.

  Prune the recovery history file for 'sample' database.

  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.
```

Исходный текст программы примера:

```
/*****************************************************************************
**
** Имя исходного файла = dbrecov.sqc
**
** Лицензионные материалы - собственность IBM
**
** (C) COPYRIGHT International Business Machines Corp. 2001
** Все права защищены.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**
**
**    ЦЕЛЬ :
**    Этот пример показывает, как восстановить базу данных.
**
**    ИСПОЛЬЗУЕМЫЕ API :
**        db2HistoryCloseScan - Закрыть просмотр файла хронологии восстановлений
**        db2HistoryGetEntry - Получить следующую запись файла хронологии
```

```
**        восстановлений
**        db2HistoryOpenScan - Открыть просмотр файла истории восстановлений
**        db2HistoryUpdate - Изменить файл хронологии восстановлений
**        db2Prune - Сократить файл хронологии восстановлений
**        sqlbmtsq - Запрос табличного пространства
**        sqlbstsc - Задать контейнеры табличного пространства
**        sqlbtcq - Запрос контейнера табличного пространства
**        sqledrpd - Отбросить базу данных
**        sqlefmem - Освободить память
**        sqlfudb - Изменить конфигурацию базы данных
**        sqlfxdb - Получить конфигурацию базы данных
**        sqlubkp - Резервное копирование базы данных
**        sqlurestore - Восстановление базы данных
**        sqlurlog - Асинхронное чтение журнала
**        sqluroll - Повтор транзакций базы данных
**
** Подробную информацию о резервном копировании и восстановлении баз данных
** смотрите в книге "Справочное руководство по восстановлению данных и высокой
** доступности". Это руководство поможет определить методы восстановления базы
** данных и табличных пространств, лучше всего подходящие для вашей среды.
**
** Дополнительную информацию об этих примерах программ смотрите в файле README.
**
** Дополнительную информацию о программировании на языке C смотрите в разделе
** "Programming in C and C++" руководства "Application Development Guide".
**
** Дополнительную информацию по разработке программ на языке C смотрите в разделе
** о компиляторе в главе "Building Applications" для вашей платформы руководства
** "Application Building Guide".
**
** Дополнительную информацию о языке SQL смотрите в справочнике "SQL Reference".
**
** Дополнительную информацию об API DB2 смотрите в справочнике "Administrative
** API Reference".
**
** Самая свежая информация о программировании, компиляции и запуске программ DB2
** находится на Web-сайте разработки программ DB2 по адресу:
**     http://www.software.ibm.com/data/db2/udb/ad
**
******************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlenv.h>
#include <sqlutil.h>
#include <db2ApiDf.h>
#include "utilemb.h"

int DbRecoveryHistoryFilePrune(char *, char *, char *);
int DbBackupAndRestore(char *, char *, char *, char *, char *);
int DbBackupAndRedirectedRestore(char *, char *, char *, char *, char *);
int DbBackupRestoreAndRollforward(char *, char *, char *, char *, char *);
int DbLogRecordsForCurrentConnectionRead(char *, char *, char *, char *);
int DbRecoveryHistoryFileRead(char *);
int DbFirstRecoveryHistoryFileEntryUpdate(char *, char *, char *);
```

```
/* функция поддержки, вызываемая main() */
int ServerWorkingPathGet(char *, char *);

/* функция поддержки, вызываемая DbBackupAndRedirectedRestore() */
int InaccessableContainersRedefine(char *);

/* функция поддержки, вызываемая DbBackupAndRedirectedRestore() и
   DbBackupRestoreAndRollforward() */
int DbDrop(char *);

/* функция поддержки, вызываемая DbLogRecordsForCurrentConnectionRead() */
int LogBufferDisplay(char *, sqluint32);
int LogRecordDisplay(char *, sqluint32, sqluint16, sqluint16);
int SimpleLogRecordDisplay(sqluint16, sqluint16, char *, sqluint32);
int ComplexLogRecordDisplay(sqluint16, sqluint16, char *, sqluint32,
                            sqluint8, char *, sqluint32);
int LogSubRecordDisplay(char *, sqluint16);
int UserDataDisplay(char *, sqluint16);

/* функции поддержки, вызываемые DbRecoveryHistoryFileRead() и
   DbFirstRecoveryHistoryFileEntryUpdate() */
int HistoryEntryDataFieldsAlloc(struct db2HistoryData *);
int HistoryEntryDisplay(struct db2HistoryData);
int HistoryEntryDataFieldsFree(struct db2HistoryData *);

/* DbCreate создаст новую базу данных на сервере с
   кодовой страницей сервера.
   Используйте эту функцию, только если хотите восстановить удаленную базу данных.
   Эту функцию поддержки вызывает DbBackupAndRedirectedRestore()
   и DbBackupRestoreAndRollforward().  */
int DbCreate(char *, char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1];
  char serverWorkingPath[SQL_PATH_SZ + 1];
  char restoredDbAlias[SQL_ALIAS_SZ + 1];
  char redirectedRestoredDbAlias[SQL_ALIAS_SZ + 1];
  char rolledForwardDbAlias[SQL_ALIAS_SZ + 1];
  sqluint16 savedLogRetainValue;
  char dbAlias[SQL_ALIAS_SZ + 1];
  char user[USERID_SZ + 1];
  char pswd[PSWD_SZ + 1];

  /* проверка аргументов командной строки */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
      if (rc != 0)
  {
        return rc;
  }

  printf("\nTHIS SAMPLE SHOWS HOW TO RECOVER A DATABASE.\n");
```

```
strcpy(restoredDbAlias, dbAlias);
strcpy(redirectedRestoredDbAlias, "RRDB");
strcpy(rolledForwardDbAlias, "RFDB");

/* присоединение к локальному или удаленному экземпляру */
rc = InstanceAttach(nodeName, user, pswd);
    if (rc != 0)
{
      return rc;
}

printf("\nUSE THE DB2 API:\n");
printf("  sqlfxdb -- Get Database Configuration\n");
printf("TO GET THE DATABASE CONFIGURATION AND DETERMINE\n");
printf("THE SERVER WORKING PATH.\n");

/* получение рабочего пути сервера */
rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
    if (rc != 0)
{
      return rc;
}

printf("\nNOTE: Backup images will be created on the server\n");
printf("      in the directory %s,\n", serverWorkingPath);
printf("      and will not be deleted by the program.\n");

/* вызов функций примеров */
rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);

rc = DbBackupAndRestore(dbAlias,
                        restoredDbAlias,
                        user,
                        pswd,
                        serverWorkingPath);

rc = DbBackupAndRedirectedRestore(dbAlias,
                                  redirectedRestoredDbAlias,
                                  user,
                                  pswd,
                                  serverWorkingPath);

rc = DbBackupRestoreAndRollforward(dbAlias,
                                   rolledForwardDbAlias,
                                   user,
                                   pswd,
                                   serverWorkingPath);

rc = DbLogRecordsForCurrentConnectionRead(dbAlias,
                                          user,
                                          pswd,
                                          serverWorkingPath);

rc = DbRecoveryHistoryFileRead(dbAlias);
```

```
  rc = DbFirstRecoveryHistoryFileEntryUpdate(dbAlias, user, pswd);

  rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);

  /* отсоединение от локального или удаленного экземпляра */
  rc = InstanceDetach(nodeName);
      if (rc != 0)
  {
        return rc;
  }

  return 0;
} /* конец main */

int ServerWorkingPathGet(char dbAlias[], char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbConfigFields[1];
  char serverLogPath[SQL_PATH_SZ + 1];
  int len;

  /* получение пути журнала сервера */
  /* SQLF_DBTN_LOGPATH - это маркер неизменяемого параметра конфигурации
     'logpath' базы данных; он используется для получения пути журнала сервера */
  dbConfigFields[0].token = SQLF_DBTN_LOGPATH;
  dbConfigFields[0].ptrvalue =
    (char *)malloc((SQL_PATH_SZ + 1) * sizeof(char));

  /* получение конфигурации базы данных */
  /* API sqlfxdb возвращает значения отдельных записей в
     файл конфигурации базы данных */
  sqlfxdb(dbAlias, 1, &dbConfigFields[0], &sqlca);
  DB2_API_CHECK("server log path -- get");

  strcpy(serverLogPath, dbConfigFields[0].ptrvalue);
  free(dbConfigFields[0].ptrvalue);

  /* выбор пути журнала сервера; если, например, serverLogPath =
     "C:\DB2\NODE0001\....", для переменной serverWorkingPath мы сохраняем
     "C:\DB2"; образы резервных копий, созданные в этом примере, будут помещены
     в каталог 'serverWorkingPath' */
  len = (int)(strstr(serverLogPath, "NODE") - serverLogPath - 1);
  memcpy(serverWorkingPath, serverLogPath, len);
  serverWorkingPath[len] = '\0';

  return 0;
} /* ServerWorkingPathGet */

int DbCreate(char existingDbAlias[], char newDbAlias[])
{
  struct sqlca sqlca;
  char dbName[SQL_DBNAME_SZ + 1];
  char dbLocalAlias[SQL_ALIAS_SZ + 1];
```

```
char dbPath[SQL_PATH_SZ + 1];
struct sqledbdesc dbDescriptor;
struct sqledbcountryinfo countryInfo;
struct sqlfupd dbConfigFields[2];

printf("\n  Create '%s' empty database with the same code set as
 '%s' database.\n",
       newDbAlias, existingDbAlias);

/* инициализация dbConfigFields */
dbConfigFields[0].token = SQLF_DBTN_TERRITORY;
dbConfigFields[0].ptrvalue = (char *)malloc(10 * sizeof(char));
memset(dbConfigFields[0].ptrvalue, '\0', 10);
dbConfigFields[1].token = SQLF_DBTN_CODESET;
dbConfigFields[1].ptrvalue = (char *)malloc(20 * sizeof(char));
memset(dbConfigFields[1].ptrvalue, '\0', 20);

/* получение двух полей конфигурации базы данных */
sqlfxdb(existingDbAlias, 2, &dbConfigFields[0], &sqlca);
DB2_API_CHECK("DB Config. -- Get");

/* создание новой базы данных */
strcpy(dbName, newDbAlias);
strcpy(dbLocalAlias, newDbAlias);
strcpy(dbPath, "");

strcpy(dbDescriptor.sqldbdid, SQLE_DBDESC_2);
dbDescriptor.sqldbccp = 0;
dbDescriptor.sqldbcss = SQL_CS_NONE;

strcpy(dbDescriptor.sqldbcmt, "");
dbDescriptor.sqldbsgp = 0;
dbDescriptor.sqldbnsg = 10;
dbDescriptor.sqltsext = -1;
dbDescriptor.sqlcatts = NULL;
dbDescriptor.sqlusrts = NULL;
dbDescriptor.sqltmpts = NULL;

strcpy(countryInfo.sqldbcodeset, (char *)dbConfigFields[1].ptrvalue);
strcpy(countryInfo.sqldblocale, (char *)dbConfigFields[0].ptrvalue);

/* создание базы данных */
sqlecrea(dbName,
         dbLocalAlias,
         dbPath,
         &dbDescriptor,
         &countryInfo,
         '\0',
           NULL,
           &sqlca);
DB2_API_CHECK("Database -- Create");

/* освобождение выделенной памяти */
free(dbConfigFields[0].ptrvalue);
free(dbConfigFields[1].ptrvalue);
```

```
  return 0;
} /* DbCreate */

int DbDrop(char dbAlias[])
{
  struct sqlca sqlca;

  printf("\n  Drop the '%s' database.\n", dbAlias);

  /* отбрасывание базы данных и удаление ее из каталога */
  sqledrpd(dbAlias, &sqlca);
  DB2_API_CHECK("Database -- Drop");

  return 0;
} /* DbDrop */

int DbBackupAndRestore(char dbAlias[],
                       char restoredDbAlias[],
                       char user[],
                       char pswd[],
                       char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  sqluint32 restoreBufferSize;
  sqluint32 rollforwardPendingState;
  sqluint32 datalinkMode;
  sqluint32 restoreType;
  sqluint32 restoreMode;
  sqluint32 restoreCallerAction;
  char restoreAppId[SQLU_APPLID_LEN + 1];
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
  char *restoreTargetPath;
  sqluint32 restoreBuffersNb;
  struct sqlu_tablespace_bkrst_list restoreTablespaceList;
  struct sqlu_media_list restoreSourceMediaList;
  struct sqlu_media_entry restoreSourceMediaEntries[1];
  sqluint32 restoreParallelismDegree;

  printf("\n************************************\n");
```

```
printf("*** BACK UP AND RESTORE A DATABASE ***\n");
printf("**************************************\n");
printf("\nUSE THE DB2 APIs:\n");
printf("  sqlfudb -- Update Database Configuration\n");
printf("  sqlubkp -- Backup Database\n");
printf("  sqlurestore -- Restore Database\n");
printf("TO BACK UP AND RESTORE A DATABASE.\n");

printf("\n  Update \'%s\' database configuration:\n", dbAlias);
printf("    - Disable the database configuration parameter LOGRETAIN\n");
printf("        i.e., set LOGRETAIN = OFF/NO\n");

/* SQLF_DBTN_LOG_RETAIN - это маркер изменяемого параметра конфигурации
   'logretain' базы данных; он используется для изменения файла конфигурации
    базы данных */
dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* отключение параметра 'logretain' конфигурации базы данных */
logretain = SQLF_LOGRETAIN_DISABLE;

/* API sqlfudb используется для изменения отдельных записей в конкретном
   файле конфигурации базы данных. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Disable");

/*****************************/
/*    РЕЗЕРВНОЕ КОПИРОВАНИЕ БАЗЫ ДАННЫХ    */
/*****************************/
printf("\n  Backing up the '%s' database...\n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* число табличных пространств */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* API sqlubkp создает резервную копию базы данных.
   Этот API автоматически устанавливает соединение с заданной
   базой данных.
   (Кроме того, этот API можно использовать для создания резервной копии
    табличного пространства). */
  sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
```

```
            backupAppId,
            backupStartTimestamp,
            backupBuffersNb,
            &backupTablespaceList,
            &backupTargetMediaList,
            user,
            pswd,
               NULL,
               0,
               NULL,
            backupParallelismDegree,
            &backupImageSize,
               NULL,
               NULL,
               &sqlca);
      DB2_API_CHECK("Database -- Backup");

    while (sqlca.sqlcode != 0)
    {
      /* продолжение операции резервного копирования */

      /* в зависимости от значения sqlca.sqlcode может потребоваться */
      /* вмешательство пользователя, например, монтирование новой ленты */

      printf("\n  Continuing the backup operation...\n");

      backupCallerAction = SQLUB_CONTINUE;

      /* резервное копирование базы данных */
      sqlubkp(dbAlias,
              backupBufferSize,
              backupMode,
              backupType,
              backupCallerAction,
              backupAppId,
              backupStartTimestamp,
              backupBuffersNb,
              &backupTablespaceList,
              &backupTargetMediaList,
              user,
              pswd,
               NULL,
               0,
               NULL,
              backupParallelismDegree,
              &backupImageSize,
               NULL,
               NULL,
               &sqlca);
      DB2_API_CHECK("Database -- Backup");
    }

    printf("  Backup finished.\n");
    printf("    - backup image size      : %d MB\n", backupImageSize);
    printf("    - backup image path      : %s\n",
```

```
       backupTargetMediaEntries[0].media_entry);
printf("    - backup image time stamp: %s\n",
       backupStartTimestamp);

/***********************************/
/*    ВОССТАНОВЛЕНИЕ БАЗЫ ДАННЫХ    */
/***********************************/

restoreBufferSize = 1024; /*  1024 x 4KB */
rollforwardPendingState = SQLUD_NOROLLFWD;
datalinkMode = SQLUD_NODATALINK;
restoreType = SQLUD_FULL;
restoreMode = SQLUD_OFFLINE;
strcpy(restoreTimestamp, backupStartTimestamp);
restoreTargetPath = NULL;
restoreBuffersNb = 1;
restoreTablespaceList.num_entry = 0; /* число табличных пространств */
restoreTablespaceList.tablespace = NULL;
restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
restoreSourceMediaList.sessions = 1; /* число элементов на устройстве
                                        назначения */
restoreSourceMediaList.target.media = restoreSourceMediaEntries;
strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
restoreParallelismDegree = 1;

printf("\n  Restoring a database ...\n");
printf("    - source image alias     : %s\n", dbAlias);
printf("    - source image time stamp: %s\n", restoreTimestamp);
printf("    - target database        : %s\n", restoredDbAlias);

restoreCallerAction = SQLUD_RESTORE;

/* API sqlurestore используется для восстановления базы данных, резервная копия
   которой была создана с использованием API sqlubkp. */
sqlurestore(dbAlias,
            restoredDbAlias,
            restoreBufferSize,
            rollforwardPendingState,
            datalinkMode,
            restoreType,
            restoreMode,
            restoreCallerAction,
            restoreAppId,
            restoreTimestamp,
            restoreTargetPath,
            restoreBuffersNb,
           NULL,
            &restoreTablespaceList,
            &restoreSourceMediaList,
            user,
            pswd,
           NULL,
           0,
           NULL,
            restoreParallelismDegree,
```

```
            NULL,
            NULL,
            NULL,
            &sqlca);
/*  DB2_API_CHECK("database restore -- start"); */
  EXPECTED_WARN_CHECK("database restore -- start");

  while (sqlca.sqlcode != 0)
  {
    /* продолжение операции восстановления */
    printf("\n  Continuing the restore operation...\n");

    /* в зависимости от значения sqlca.sqlcode может потребоваться
       вмешательство пользователя, например, монтирование новой ленты */

    restoreCallerAction = SQLUD_CONTINUE;

    /* восстановление базы данных */
    sqlurestore(dbAlias,
                restoredDbAlias,
                restoreBufferSize,
                rollforwardPendingState,
                datalinkMode,
                restoreType,
                restoreMode,
                restoreCallerAction,
                restoreAppId,
                restoreTimestamp,
                restoreTargetPath,
                restoreBuffersNb,
              NULL,
                &restoreTablespaceList,
                &restoreSourceMediaList,
                user,
                pswd,
              NULL,
              0,
              NULL,
                restoreParallelismDegree,
              NULL,
              NULL,
              NULL,
              &sqlca);
    DB2_API_CHECK("database restore -- continue");
  }

  printf("\n  Restore finished.\n");

  return 0;
} /* DbBackupAndRestore */

int DbBackupAndRedirectedRestore(char dbAlias[],
                                 char restoredDbAlias[],
                                 char user[],
                                 char pswd[],
```

```
                                    char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  sqluint32 restoreBufferSize;
  sqluint32 rollforwardPendingState;
  sqluint32 datalinkMode;
  sqluint32 restoreType;
  sqluint32 restoreMode;
  sqluint32 restoreCallerAction;
  char restoreAppId[SQLU_APPLID_LEN + 1];
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
  char *restoreTargetPath;
  sqluint32 restoreBuffersNb;
  struct sqlu_tablespace_bkrst_list restoreTablespaceList;
  struct sqlu_media_list restoreSourceMediaList;
  struct sqlu_media_entry restoreSourceMediaEntries[1];
  sqluint32 restoreParallelismDegree;

  printf("\n*************************\n");
  printf("*** REDIRECTED RESTORE ***\n");
  printf("*************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  sqlfudb -- Update Database Configuration\n");
  printf("  sqlubkp -- Backup Database\n");
  printf("  sqlecrea -- Create Database\n");
  printf("  sqlurestore -- Restore Database\n");
  printf("  sqlbmtsq -- Tablespace Query\n");
  printf("  sqlbtcq -- Tablespace Container Query\n");
  printf("  sqlbstsc -- Set Tablespace Containers\n");
  printf("  sqlefmem -- Free Memory\n");
  printf("  sqledrpd -- Drop Database\n");
  printf("TO BACK UP AND DO A REDIRECTED RESTORE OF A DATABASE.\n");

  printf("\n  Update \'%s\' database configuration:\n", dbAlias);
  printf("    - Disable the database configuration parameter LOGRETAIN \n");
  printf("        i.e., set LOGRETAIN = OFF/NO\n");

  /* SQLF_DBTN_LOG_RETAIN - это маркер изменяемого параметра конфигурации
     'logretain' базы данных; он используется для изменения файла конфигурации
     базы данных */
```

```
dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* отключение параметра 'logretain' конфигурации базы данных */
logretain = SQLF_LOGRETAIN_DISABLE;

/* API sqlfudb используется для изменения отдельных записей в конкретном
   файле конфигурации базы данных. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Disable");

/*****************************************/
/*    РЕЗЕРВНОЕ КОПИРОВАНИЕ БАЗЫ ДАННЫХ    */
/*****************************************/
printf("\n  Backing up the '%s' database...\n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* число табличных пространств */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* число элементов на устройстве назначения */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* API sqlubkp создает резервную копию базы данных.
   Этот API автоматически устанавливает соединение с заданной
   базой данных.
   (Кроме того, этот API можно использовать для создания резервной копии
   табличного пространства). */
  sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
           NULL,
           0,
           NULL,
        backupParallelismDegree,
        &backupImageSize,
           NULL,
           NULL,
           &sqlca);
```

```
    DB2_API_CHECK("Database -- Backup");

  while (sqlca.sqlcode != 0)
  {
    /* продолжение операции резервного копирования */

    /* в зависимости от значения sqlca.sqlcode может потребоваться
       вмешательство пользователя, например, монтирование новой ленты */

    printf("\n  Continuing the backup operation...\n");

    backupCallerAction = SQLUB_CONTINUE;

    /* резервное копирование базы данных */
    sqlubkp(dbAlias,
            backupBufferSize,
            backupMode,
            backupType,
            backupCallerAction,
            backupAppId,
            backupStartTimestamp,
            backupBuffersNb,
            &backupTablespaceList,
            &backupTargetMediaList,
            user,
            pswd,
             NULL,
             0,
             NULL,
            backupParallelismDegree,
            &backupImageSize,
             NULL,
             NULL,
             &sqlca);
    DB2_API_CHECK("Database -- Backup");
  }

  printf("  Backup finished.\n");
  printf("    - backup image size      : %d MB\n", backupImageSize);
  printf("    - backup image path      : %s\n",
         backupTargetMediaEntries[0].media_entry);
  printf("    - backup image time stamp: %s\n",
         backupStartTimestamp);

  /* Если кодовая страница клиента отличается от
     кодовой страницы сервера, чтобы восстановить удаленную базу данных,
     сначала нужно создать пустую базу данных.
     В этом случае удалите знаки комментария вокруг вызова DbCreate().
     Пустая база данных будет создана на сервере с кодовой страницей этого
     сервера. */

/*
  rc = DbCreate(dbAlias, restoredDbAlias);
      if (rc != 0)
  {
```

```
        return rc;
    }
*/

    /***********************************/
    /*    ВОССТАНОВЛЕНИЕ БАЗЫ ДАННЫХ    */
    /***********************************/

    restoreBufferSize = 1024; /*  1024 x 4KB */
    rollforwardPendingState = SQLUD_NOROLLFWD;
    datalinkMode = SQLUD_NODATALINK;
    restoreType = SQLUD_FULL;
    restoreMode = SQLUD_OFFLINE;
    strcpy(restoreTimestamp, backupStartTimestamp);
    restoreTargetPath = NULL;
    restoreBuffersNb = 1;
    restoreTablespaceList.num_entry = 0; /* число табличных пространств */
    restoreTablespaceList.tablespace = NULL;
    restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
    restoreSourceMediaList.sessions = 1; /* число элементов на устройстве назначения */
    restoreSourceMediaList.target.media = restoreSourceMediaEntries;
    strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
    restoreParallelismDegree = 1;

    printf("\n  Restoring a database ...\n");
    printf("    - source image alias     : %s\n", dbAlias);
    printf("    - source image time stamp: %s\n", restoreTimestamp);
    printf("    - target database        : %s\n", restoredDbAlias);

    restoreCallerAction = SQLUD_RESTORE_STORDEF;

    /* API sqlurestore используется для восстановления базы данных, резервная копия
       которой была создана с использованием API sqlubkp. */
    sqlurestore(dbAlias,
                restoredDbAlias,
                restoreBufferSize,
                rollforwardPendingState,
                datalinkMode,
                restoreType,
                restoreMode,
                restoreCallerAction,
                restoreAppId,
                restoreTimestamp,
                restoreTargetPath,
                restoreBuffersNb,
               NULL,
                &restoreTablespaceList,
                &restoreSourceMediaList,
                user,
                pswd,
               NULL,
               0,
               NULL,
                restoreParallelismDegree,
               NULL,
```

```
            NULL,
            NULL,
            &sqlca);
/*  DB2_API_CHECK("database restore -- start"); */
  EXPECTED_WARN_CHECK("database restore -- start");

  while (sqlca.sqlcode != 0)
  {
    /* продолжение операции восстановления */
    printf("\n  Continuing the restore operation...\n");

    /* в зависимости от значения sqlca.sqlcode может потребоваться
       вмешательство пользователя, например, монтирование новой ленты */

    if (sqlca.sqlcode == SQLUD_INACCESSABLE_CONTAINER)
    {
      /* переопределение расположения контейнеров табличного пространства */
      printf("\n  Find and redefine inaccessable containers.\n");
      rc = InaccessableContainersRedefine(serverWorkingPath);
      if (rc != 0)
      {
        return rc;
      }
    }

    restoreCallerAction = SQLUD_CONTINUE;

    /* восстановление базы данных */
    sqlurestore(dbAlias,
                restoredDbAlias,
                restoreBufferSize,
                rollforwardPendingState,
                datalinkMode,
                restoreType,
                restoreMode,
                restoreCallerAction,
                restoreAppId,
                restoreTimestamp,
                restoreTargetPath,
                restoreBuffersNb,
              NULL,
                &restoreTablespaceList,
                &restoreSourceMediaList,
                user,
                pswd,
              NULL,
              0,
              NULL,
                restoreParallelismDegree,
              NULL,
              NULL,
              NULL,
              &sqlca);
    DB2_API_CHECK("database restore -- continue");
  }
```

```
  printf("\n  Restore finished.\n");

  /* отбрасывание восстановленной базы данных */
  rc = DbDrop(restoredDbAlias);

  return 0;
} /* DbBackupAndRedirectedRestore */

int InaccessableContainersRedefine(char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  sqluint32 numTablespaces;
  struct SQLB_TBSPQRY_DATA **ppTablespaces;
  sqluint32 numContainers;
  struct SQLB_TBSCONTQRY_DATA *pContainers;
  int tspNb;
  int contNb;
  char pathSep[2];

  /* API sqlbmtsq обеспечивает интерфейс одного вызова для данных запроса
      табличного пространства. Данные запроса для всех табличных пространств
      в базе данных возвращаются в массив. */
  sqlbmtsq(&sqlca,
           &numTablespaces,
           &ppTablespaces,
           SQLB_RESERVED1,
           SQLB_RESERVED2);
  DB2_API_CHECK("tablespaces -- get");

  /* переопределение недоступных контейнеров */
  for (tspNb = 0; tspNb < numTablespaces; tspNb++)
  {
    /* API sqlbtcq обеспечивает интерфейс одного вызова для данных запроса
       контейнеров табличного пространства. Данные запроса для всех контейнеров в
       табличном пространстве или для всех контейнеров во всех табличных
       пространствах возвращаются в массив. */
    sqlbtcq(&sqlca, ppTablespaces[tspNb]->id, &numContainers, &pContainers);
    DB2_API_CHECK("tablespace containers -- get");

    for (contNb = 0; contNb < numContainers; contNb++)
    {
      if (!pContainers[contNb].ok)
      {
        /* переопределение недоступного контейнера */
        printf("\n    Redefine inaccessible container:\n");
        printf("      - table space name: %s\n",
               ppTablespaces[tspNb]->name);
        printf("      - default container name: %s\n",
             pContainers[contNb].name);
        if (strstr(pContainers[contNb].name, "/"))
        { /* UNIX */
          strcpy(pathSep, "/");
        }
```

```
  else
        { /* Intel */
          strcpy(pathSep, "\\");
        }
        switch (pContainers[contNb].contType)
        {
          case SQLB_CONT_PATH:
            printf("      - container type: path\n");

            sprintf(pContainers[contNb].name, "%s%sSQLT%04d.%d",
                                              serverWorkingPath, pathSep,
             ppTablespaces[tspNb]->id,
                                              pContainers[contNb].id);
            printf("      - new container name: %s\n",
             pContainers[contNb].name);
      break;
          case SQLB_CONT_DISK:
          case SQLB_CONT_FILE:
    default:
            printf("    Unknown container type.\n");
      break;
        }
      }
    }

    /* API sqlbstsc используется для задания или переопределения контейнеров
       табличных пространств при перенаправленном восстановлении базы данных. */
    sqlbstsc(&sqlca,
             SQLB_SET_CONT_FINAL_STATE,
             ppTablespaces[tspNb]->id,
             numContainers,
             pContainers);
    DB2_API_CHECK("tablespace containers -- redefine");

    /* API sqlefmem используется для высвобождения памяти, выделенной DB2 под
       использование с API sqlbtcq (Запрос контейнеров табличного пространства). */
    sqlefmem(&sqlca, pContainers);
    DB2_API_CHECK("tablespace containers memory -- free");
  }

    /* API sqlefmem используется для высвобождения памяти, выделенной DB2
     под использование с API sqlbmtsq (Запрос табличных пространств). */
  sqlefmem(&sqlca, ppTablespaces);
  DB2_API_CHECK("tablespaces memory -- free");

  return 0;
} /* InaccessibleContainersRedefine */

int DbBackupRestoreAndRollforward(char dbAlias[],
                                  char rolledForwardDbAlias[],
                                  char user[],
                                  char pswd[],
                                  char serverWorkingPath[])
{
  int rc = 0;
```

```
struct sqlca sqlca;
struct sqlfupd dbCfgParameters[1];
unsigned short logretain;
sqluint32 backupBufferSize;
sqluint32 backupMode;
sqluint32 backupType;
sqluint32 backupCallerAction;
char backupAppId[SQLU_APPLID_LEN + 1];
char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
sqluint32 backupBuffersNb;
struct sqlu_tablespace_bkrst_list backupTablespaceList;
struct sqlu_media_list backupTargetMediaList;
struct sqlu_media_entry backupTargetMediaEntries[1];
sqluint32 backupParallelismDegree;
sqluint32 backupImageSize;
sqluint32 restoreBufferSize;
sqluint32 rollforwardPendingState;
sqluint32 datalinkMode;
sqluint32 restoreType;
sqluint32 restoreMode;
sqluint32 restoreCallerAction;
char restoreAppId[SQLU_APPLID_LEN + 1];
char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
char *restoreTargetPath;
sqluint32 restoreBuffersNb;
struct sqlu_tablespace_bkrst_list restoreTablespaceList;
struct sqlu_media_list restoreSourceMediaList;
struct sqlu_media_entry restoreSourceMediaEntries[1];
sqluint32 restoreParallelismDegree;
struct rfwd_input rfwdInput;
struct rfwd_output rfwdOutput;
char rollforwardAppId[SQLU_APPLID_LEN + 1];
sqlint32 numReplies;
struct sqlurf_info nodeInfo;

printf("\n***************************\n");
printf("*** ROLLFORWARD RECOVERY ***\n");
printf("***************************\n");
printf("\nUSE THE DB2 APIs:\n");
printf("  sqlfudb -- Update Database Configuration\n");
printf("  sqlubkp -- Backup Database\n");
printf("  sqlecrea -- Create Database\n");
printf("  sqlurestore -- Restore Database\n");
printf("  sqluroll -- Rollforward Database\n");
printf("  sqledrpd -- Drop Database\n");
printf("TO BACK UP, RESTORE, AND ROLL A DATABASE FORWARD. \n");

printf("\n  Update \'%s\' database configuration:\n", dbAlias);
printf("    - Enable the configuration parameter LOGRETAIN \n");
printf("        i.e., set LOGRETAIN = RECOVERY/YES\n");


dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;
```

```
/* включение параметра конфигурации 'logretain' */
logretain = SQLF_LOGRETAIN_RECOVERY;

/* API sqlfudb используется для изменения отдельных записей в конкретном
   файле конфигурации базы данных. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Enable");

/* запуск операции резервного копирования */
printf("\n  Backing up the '%s' database...\n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* число табличных пространств */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* число элементов на устройстве назначения */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* API sqlubkp создает резервную копию базы данных.
   Этот API автоматически устанавливает соединение с заданной
   базой данных.
   (Кроме того, этот API можно использовать для создания резервной копии
    табличного пространства). */
  sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
           NULL,
           0,
           NULL,
        backupParallelismDegree,
        &backupImageSize,
           NULL,
           NULL,
           &sqlca);
  DB2_API_CHECK("Database -- Backup");

while (sqlca.sqlcode != 0)
{
  /* продолжение операции резервного копирования */
```

```
    printf("\n  Continuing the backup operation...\n");

    /* в зависимости от значения sqlca.sqlcode может потребоваться
       вмешательство пользователя, например, монтирование новой ленты */

    backupCallerAction = SQLUB_CONTINUE;

    /* резервное копирование базы данных */
    sqlubkp(dbAlias,
            backupBufferSize,
            backupMode,
            backupType,
            backupCallerAction,
            backupAppId,
            backupStartTimestamp,
            backupBuffersNb,
            &backupTablespaceList,
            &backupTargetMediaList,
            user,
            pswd,
             NULL,
             0,
             NULL,
            backupParallelismDegree,
            &backupImageSize,
             NULL,
             NULL,
             &sqlca);
    DB2_API_CHECK("Database -- Backup");
  }

  printf("  Backup finished.\n");
  printf("    - backup image size      : %d MB\n", backupImageSize);
  printf("    - backup image path      : %s\n",
         backupTargetMediaEntries[0].media_entry);
  printf("    - backup image time stamp: %s\n",
         backupStartTimestamp);

  /* Если кодовая страница клиента отличается от
     кодовой страницы сервера, чтобы восстановить удаленную базу данных,
     сначала нужно создать пустую базу данных.
     В этом случае удалите знаки комментария вокруг вызова DbCreate().
     Пустая база данных будет создана на сервере с кодовой страницей этого
     сервера. */

/*
  rc = DbCreate(dbAlias, rolledForwardDbAlias);
      if (rc != 0)
  {
        return rc;
  }
*/

  /***********************************/
  /*    ВОССТАНОВЛЕНИЕ БАЗЫ ДАННЫХ    */
```

```
/***********************************/

restoreBufferSize = 1024; /*  1024 x 4KB */
rollforwardPendingState = SQLUD_ROLLFWD;
datalinkMode = SQLUD_NODATALINK;
restoreType = SQLUD_FULL;
restoreMode = SQLUD_OFFLINE;
strcpy(restoreTimestamp, backupStartTimestamp);
restoreTargetPath = NULL;
restoreBuffersNb = 1;
restoreTablespaceList.num_entry = 0; /* число табличных пространств */
restoreTablespaceList.tablespace = NULL;
restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
restoreSourceMediaList.sessions = 1; /* число элементов на устройстве
                                        назначения */
restoreSourceMediaList.target.media = restoreSourceMediaEntries;
strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
restoreParallelismDegree = 1;

printf("\n  Restoring a database ...\n");
printf("    - source image alias     : %s\n", dbAlias);
printf("    - source image time stamp: %s\n", restoreTimestamp);
printf("    - target database        : %s\n", rolledForwardDbAlias);

restoreCallerAction = SQLUD_RESTORE;

/* API sqlurestore используется для восстановления базы данных, резервная копия
   которой была создана с использованием API sqlubkp. */
sqlurestore(dbAlias,
            rolledForwardDbAlias,
            restoreBufferSize,
            rollforwardPendingState,
            datalinkMode,
            restoreType,
            restoreMode,
            restoreCallerAction,
            restoreAppId,
            restoreTimestamp,
            restoreTargetPath,
            restoreBuffersNb,
           NULL,
            &restoreTablespaceList,
            &restoreSourceMediaList,
            user,
            pswd,
           NULL,
           0,
           NULL,
            restoreParallelismDegree,
           NULL,
           NULL,
           NULL,
           &sqlca);
DB2_API_CHECK("database restore -- start");
```

```
while (sqlca.sqlcode != 0)
{
  /* продолжение операции восстановления */
  printf("\n  Continuing the restore operation...\n");

  /* В зависимости от значения sqlca.sqlcode может потребоваться
     вмешательство пользователя, например, монтирование новой ленты */

  restoreCallerAction = SQLUD_CONTINUE;

  /* восстановление базы данных */
  sqlurestore(dbAlias,
              rolledForwardDbAlias,
              restoreBufferSize,
              rollforwardPendingState,
              datalinkMode,
              restoreType,
              restoreMode,
              restoreCallerAction,
              restoreAppId,
              restoreTimestamp,
              restoreTargetPath,
              restoreBuffersNb,
           NULL,
              &restoreTablespaceList,
              &restoreSourceMediaList,
              user,
              pswd,
           NULL,
           0,
           NULL,
              restoreParallelismDegree,
           NULL,
           NULL,
           NULL,
           &sqlca);
  DB2_API_CHECK("database restore -- continue");
}

printf("\n  Restore finished.\n");


/*********************************************/
/*    ВОССТАНОВЛЕНИЕ С ПОВТОРОМ ТРАНЗАКЦИЙ    */
/*********************************************/

printf("\n  Rolling '%s' database forward ...\n", rolledForwardDbAlias);

rfwdInput.version = SQLUM_RFWD_VERSION;
rfwdInput.pDbAlias = rolledForwardDbAlias;
rfwdInput.CallerAction = SQLUM_ROLLFWD_STOP;
rfwdInput.pStopTime = SQLUM_INFINITY_TIMESTAMP;
rfwdInput.pUserName = user;
rfwdInput.pPassword = pswd;
rfwdInput.pOverflowLogPath = serverWorkingPath;
```

```
  rfwdInput.NumChngLgOvrflw = 0;
  rfwdInput.pChngLogOvrflw = NULL;
  rfwdInput.ConnectMode = SQLUM_OFFLINE;
  rfwdInput.pTablespaceList = NULL;
   rfwdInput.AllNodeFlag= SQLURF_ALL_NODES;
  rfwdInput.NumNodes = 0;
  rfwdInput.pNodeList = NULL;
  rfwdInput.NumNodeInfo = 1;
  rfwdInput.DlMode = 0;
  rfwdInput.pReportFile = NULL;
  rfwdInput.pDroppedTblID = NULL;
  rfwdInput.pExportDir = NULL;

  rfwdOutput.pApplicationId = rollforwardAppId;
  rfwdOutput.pNumReplies = &numReplies;
  rfwdOutput.pNodeInfo = &nodeInfo;

  /* повтор транзакций базы данных */
  /* API sqluroll восстанавливает базу данных с повтором транзакций, используя
     транзакции, записанные в файлах журналов базы данных. */
  sqluroll(&rfwdInput, &rfwdOutput, &sqlca);
  DB2_API_CHECK("rollforward -- start");

  printf("  Rollforward finished.\n");

  /* отбрасывание восстановленной базы данных */
  rc = DbDrop(rolledForwardDbAlias);

  return 0;
} /* DbBackupRestoreAndRollforward */

int DbLogRecordsForCurrentConnectionRead(char dbAlias[],
                                         char user[],
                                         char pswd[],
                                         char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  SQLU_LSN startLSN;
  SQLU_LSN endLSN;
  char *logBuffer;
```

```
sqluint32 logBufferSize;
SQLU_RLOG_INFO readLogInfo;
      int i;

printf("\n*****************************\n");
printf("*** ASYNCHRONOUS READ LOG ***\n");
printf("*****************************\n");
printf("\nUSE THE DB2 APIs:\n");
printf("  sqlfudb -- Update Database Configuration\n");
printf("  sqlubkp -- Backup Database\n");
printf("  sqlurlog -- Asynchronous Read Log\n");
printf("AND THE SQL STATEMENTS:\n");
printf("  CONNECT\n");
printf("  ALTER TABLE\n");
printf("  COMMIT\n");
printf("  INSERT\n");
printf("  DELETE\n");
printf("  ROLLBACK\n");
printf("  CONNECT RESET\n");
printf("TO READ LOG RECORDS FOR THE CURRENT CONNECTION.\n");

printf("\n  Update \'%s\' database configuration:\n", dbAlias);
printf("    - Enable the database configuration parameter LOGRETAIN \n");
printf("        i.e., set LOGRETAIN = RECOVERY/YES\n");

dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* включение LOGRETAIN */
logretain = SQLF_LOGRETAIN_RECOVERY;

/* API sqlfudb используется для изменения отдельных записей в конкретном
   файле конфигурации базы данных. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Enable");

/* запуск операции резервного копирования */
printf("\n  Backing up the '%s' database...\n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* число табличных пространств */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* API sqlubkp создает резервную копию базы данных.
   Этот API автоматически устанавливает соединение с заданной
```

```
   базой данных.
   (Кроме того, этот API можно использовать для создания резервной копии
    табличного пространства). */
  sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
           NULL,
           0,
           NULL,
        backupParallelismDegree,
        &backupImageSize,
           NULL,
           NULL,
           &sqlca);
  DB2_API_CHECK("Database -- Backup");

while (sqlca.sqlcode != 0)
{
  /* продолжение операции резервного копирования */
  printf("\n  Continuing the backup operation...\n");

  /* В зависимости от значения sqlca.sqlcode может потребоваться
     вмешательство пользователя, например, монтирование новой ленты */

  backupCallerAction = SQLUB_CONTINUE;

  /* резервное копирование базы данных */
  sqlubkp(dbAlias,
          backupBufferSize,
          backupMode,
          backupType,
          backupCallerAction,
          backupAppId,
          backupStartTimestamp,
          backupBuffersNb,
          &backupTablespaceList,
          &backupTargetMediaList,
          user,
          pswd,
           NULL,
           0,
           NULL,
          backupParallelismDegree,
          &backupImageSize,
           NULL,
           NULL,
```

```
            &sqlca);
  DB2_API_CHECK("Database -- Backup");
}

printf("  Backup finished.\n");
printf("    - backup image size      : %d MB\n", backupImageSize);
printf("    - backup image path      : %s\n",
       backupTargetMediaEntries[0].media_entry);
printf("    - backup image time stamp: %s\n",
       backupStartTimestamp);

/* соединение с базой данных */
rc = DbConn(dbAlias, user, pswd);
    if (rc != 0)
{
      return rc;
}

/* вызов операторов SQL для заполнения журнала базы данных */
printf("\n  Invoke the following SQL statements:\n"
       "    ALTER TABLE emp_resume DATA CAPTURE CHANGES;\n"
       "    COMMIT;\n"
       "    INSERT INTO emp_resume\n"
       "      VALUES('000777', 'ascii', 'This is a new resume.');\n"
       "            ('777777', 'ascii', 'This is another new resume');\n"
       "    COMMIT;\n"
       "    DELETE FROM emp_resume WHERE empno = '000777';\n"
       "    DELETE FROM emp_resume WHERE empno = '777777';\n"
       "    COMMIT;\n"
       "    DELETE FROM emp_resume WHERE empno = '000140';\n"
       "    ROLLBACK;\n"
       "    ALTER TABLE emp_resume DATA CAPTURE NONE;\n"
       "    COMMIT;\n");

EXEC SQL ALTER TABLE emp_resume DATA CAPTURE CHANGES;
EMB_SQL_CHECK("SQL statement 1 -- invoke");

EXEC SQL COMMIT;
EMB_SQL_CHECK("SQL statement 2 -- invoke");

EXEC SQL INSERT INTO emp_resume
  VALUES('000777', 'ascii', 'This is a new resume.'),
        ('777777', 'ascii', 'This is another new resume');
EMB_SQL_CHECK("SQL statement 3 -- invoke");

EXEC SQL COMMIT;
EMB_SQL_CHECK("SQL statement 4 -- invoke");

EXEC SQL DELETE FROM emp_resume WHERE empno = '000777';
EMB_SQL_CHECK("SQL statement 5 -- invoke");

EXEC SQL DELETE FROM emp_resume WHERE empno = '777777';
EMB_SQL_CHECK("SQL statement 6 -- invoke");

EXEC SQL COMMIT;
```

```
  EMB_SQL_CHECK("SQL statement 7 -- invoke");

  EXEC SQL DELETE FROM emp_resume WHERE empno = '000140';
  EMB_SQL_CHECK("SQL statement 8 -- invoke");

  EXEC SQL ROLLBACK;
  EMB_SQL_CHECK("SQL statement 9 -- invoke");

  EXEC SQL ALTER TABLE emp_resume DATA CAPTURE NONE;
  EMB_SQL_CHECK("SQL statement 10 -- invoke");

  EXEC SQL COMMIT;
  EMB_SQL_CHECK("SQL statement 11 -- invoke");

  printf("\n  Start reading database log.\n");

  logBuffer = NULL;
  logBufferSize = 0;

  /* API sqlurlog (Асинхронное чтение журнала) используется для извлечения
     записей из журналов базы данных и для запроса у менеджера журналов текущей
     информации о состоянии журналов.
     Этот API можно использовать только для восстановимых баз данных. */

  /* Запрос у менеджера журналов текущей информации о состоянии журналов: */
  rc = sqlurlog(SQLU_RLOG_QUERY,
                &startLSN,
                &endLSN,
                logBuffer,
                logBufferSize,
                &readLogInfo,
              &sqlca);
/*  DB2_API_CHECK("database log info -- get"); */
  EXPECTED_WARN_CHECK("database log info -- get");

  logBufferSize = 64 * 1024;
  logBuffer = (char *)malloc(logBufferSize);

  memcpy(&startLSN, &(readLogInfo.initialLSN), sizeof(startLSN));
  memcpy(&endLSN, &(readLogInfo.curActiveLSN), sizeof(endLSN));

    /* Извлечение записи журналов из журналов базы данных и
     асинхронное чтение первого журнала */
  rc = sqlurlog(SQLU_RLOG_READ,
                &startLSN,
                &endLSN,
                logBuffer,
                logBufferSize,
                &readLogInfo,
              &sqlca);
  if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
      DB2_API_CHECK("database logs -- read");
  }
  else
```

```
{
  if (readLogInfo.logRecsWritten == 0)
  {
    printf("\n  Database log empty.\n");
  }
}

  /* вывод буфера журналов */
  rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);

while (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
{
  /* чтение следующего журнала */

  memcpy(&startLSN, &(readLogInfo.lastReadLSN), sizeof(startLSN));
  /* увеличение startLSN на 1 */
  startLSN.lsnChar[5] = startLSN.lsnChar[5] + 1;
  i = 5;
  while (startLSN.lsnChar[i] == 0 && i > 0)
  {
    startLSN.lsnChar[i - 1] = startLSN.lsnChar[i - 1] + 1;
    i = i - 1;
  }

  /* Извлечение записи журналов из журналов базы данных и
   асинхронное чтение следующего журнала: */
rc = sqlurlog(SQLU_RLOG_READ,
              &startLSN,
              &endLSN,
              logBuffer,
              logBufferSize,
              &readLogInfo,
         &sqlca);
  if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
    DB2_API_CHECK("database logs -- read");
  }

  /* вывод буфера журналов */
  rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
}

/* освобождение буфера журналов */
free(logBuffer);

/* отсоединение от базы данных */
rc = DbDisconn(dbAlias);
    if (rc != 0)
{
      return rc;
}

return 0;
} /* DbLogRecordsForCurrentConnectionRead */
```

```
int LogBufferDisplay(char *logBuffer, sqluint32 numLogRecords)
{
  int rc = 0;
  sqluint32 logRecordNb;
  sqluint32 recordSize;
  sqluint16 recordType;
  sqluint16 recordFlag;
  char *recordBuffer;

  /* инициализация recordBuffer */
  recordBuffer = logBuffer + sizeof(SQLU_LSN);

  for (logRecordNb = 0; logRecordNb < numLogRecords; logRecordNb++)
  {
    recordSize = *(sqluint32 *)(recordBuffer);
    recordType = *(sqluint16 *)(recordBuffer + 4);
    recordFlag = *(sqluint16 *)(recordBuffer + 6);

    rc = LogRecordDisplay(recordBuffer, recordSize, recordType, recordFlag);
    /* обновление recordBuffer */
    recordBuffer = recordBuffer + recordSize + sizeof(SQLU_LSN);
  }

  return 0;
} /* LogBufferDisplay */

int LogRecordDisplay(char *recordBuffer,
                     sqluint32 recordSize,
                     sqluint16 recordType,
                     sqluint16 recordFlag)
{
  int rc = 0;
  sqluint32 logManagerLogRecordHeaderSize;
  char *recordDataBuffer;
  sqluint32 recordDataSize;
  char *recordHeaderBuffer;
  sqluint8 componentIdentifier;
  sqluint32 recordHeaderSize;

  /* определение logManagerLogRecordHeaderSize */
  if ((char)recordType == 'C')
  { /* компенсация */
    if (recordFlag & 0x0002)
    { /* распространимая */
      logManagerLogRecordHeaderSize = 32;
    }
  else
    {
      logManagerLogRecordHeaderSize = 26;
    }
  }
  else
  { /* без компенсации */
    logManagerLogRecordHeaderSize = 20;
  }
```

```
  switch ((char)recordType)
  {
    case 'E':
    case 'M':
    case 'A':
      recordDataBuffer = recordBuffer + logManagerLogRecordHeaderSize;
      recordDataSize = recordSize - logManagerLogRecordHeaderSize;
      rc = SimpleLogRecordDisplay(recordType,
                                  recordFlag,
                                  recordDataBuffer,
                                  recordDataSize);
      break;
    case 'N':
    case 'C':
      recordHeaderBuffer = recordBuffer + logManagerLogRecordHeaderSize;
      componentIdentifier = *(sqluint8 *)recordHeaderBuffer;
      switch (componentIdentifier)
      {
        case 1:
          recordHeaderSize = 6;
      break;
    default:
          printf("    Unknown complex log record: %lu %c %u\n",
                 recordSize, recordType, componentIdentifier);
      return 1;
      }
      recordDataBuffer = recordBuffer +
                         logManagerLogRecordHeaderSize +
                         recordHeaderSize;
      recordDataSize = recordSize -
                       logManagerLogRecordHeaderSize -
                       recordHeaderSize;
      rc = ComplexLogRecordDisplay(recordType,
                                   recordFlag,
                                   recordHeaderBuffer,
                                   recordHeaderSize,
                                   componentIdentifier,
                                   recordDataBuffer,
                                   recordDataSize);
      break;
    default:
      printf("    Unknown log record: %lu %c\n",
             recordSize, (char)recordType);
      break;
  }

  return 0;
} /* LogRecordDisplay */

int SimpleLogRecordDisplay(sqluint16 recordType,
                           sqluint16 recordFlag,
                           char *recordDataBuffer,
                           sqluint32 recordDataSize)
{
```

```
  int rc = 0;
  sqluint32 timeTransactionCommited;
  sqluint16 authIdLen;
  char authId[129];

  switch ((char)recordType)
  {
    case 'E':
      printf("\n    Record type: Local pending list\n");
      timeTransactionCommited = *(sqluint32 *)(recordDataBuffer);
      authIdLen = *(sqluint16 *)(recordDataBuffer + 4);
      memcpy(authId, (char *)(recordDataBuffer + 6), authIdLen);
      authId[authIdLen] = '\0';
      printf("      %s: %lu\n",
             "UTC transaction committed (in seconds since 01/01/70)",
             timeTransactionCommited);
      printf("      authorization ID of the application: %s\n", authId);
      break;
    case 'M':
      printf("\n    Record type: Normal commit\n");
      timeTransactionCommited = *(sqluint32 *)(recordDataBuffer);
      authIdLen = (sqluint16) (recordDataSize - 4);
      memcpy(authId, (char *)(recordDataBuffer + 4), authIdLen);
      authId[authIdLen] = '\0';
      printf("      %s: %lu\n",
             "UTC transaction committed (in seconds since 01/01/70)",
             timeTransactionCommited);
      printf("      authorization ID of the application: %s\n", authId);
      break;
    case 'A':
      printf("\n    Record type: Normal abort\n");
      authIdLen = (sqluint16) (recordDataSize);
      memcpy(authId, (char *)(recordDataBuffer), authIdLen);
      authId[authIdLen] = '\0';
      printf("      authorization ID of the application: %s\n", authId);
      break;
    default:
      printf("    Unknown simple log record: %c %lu\n",
             (char)recordType, recordDataSize);
      break;
  }

  return 0;
} /* SimpleLogRecordDisplay */

int ComplexLogRecordDisplay(sqluint16 recordType,
                            sqluint16 recordFlag,
                            char *recordHeaderBuffer,
                            sqluint32 recordHeaderSize,
                            sqluint8 componentIdentifier,
                            char *recordDataBuffer,
                            sqluint32 recordDataSize)
{
  int rc = 0;
  sqluint8 functionIdentifier;
```

```
/* для вставки, удаления и восстановления после удаления */
sqluint32 RID;
sqluint16 subRecordLen;
sqluint16 subRecordOffset;
char *subRecordBuffer;
/* для изменения */
sqluint32 newRID;
sqluint16 newSubRecordLen;
sqluint16 newSubRecordOffset;
char *newSubRecordBuffer;
sqluint32 oldRID;
sqluint16 oldSubRecordLen;
sqluint16 oldSubRecordOffset;
char *oldSubRecordBuffer;
/* для изменения атрибутов таблицы */
sqluint32 alterBitMask;
sqluint32 alterBitValues;

switch ((char)recordType)
{
  case 'N':
    printf("\n    Record type: Normal\n");
    break;
  case 'C':
    printf("\n    Record type: Compensation\n");
    break;
  default:
    printf("\n    Unknown complex log record type: %c\n", recordType);
    break;
}

switch (componentIdentifier)
{
  case 1:
    printf("      component ID: DMS log record\n");
    break;
  default:
    printf("      unknown component ID: %d\n", componentIdentifier);
    break;
}

functionIdentifier = *(sqluint8 *)(recordHeaderBuffer + 1);
switch (functionIdentifier)
{
  case 106:
    printf("      function ID: Delete Record\n");
    RID = *(sqluint32 *)(recordDataBuffer + 2);
    subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    printf("        RID: %lu\n", RID);
    printf("        subrecord length: %u\n", subRecordLen);
    printf("        subrecord offset: %u\n", subRecordOffset);
    subRecordBuffer = recordDataBuffer + 12;
    rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
    break;
```

```
case 111:
  printf("      function ID: Undo Delete Record\n");
  RID = *(sqluint32 *)(recordDataBuffer + 2);
  subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
  subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
  printf("        RID: %lu\n", RID);
  printf("        subrecord length: %u\n", subRecordLen);
  printf("        subrecord offset: %u\n", subRecordOffset);
  subRecordBuffer = recordDataBuffer + 12;
  rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
  break;
case 118:
  printf("      function ID: Insert Record\n");
  RID = *(sqluint32 *)(recordDataBuffer + 2);
  subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
  subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
  printf("        RID: %lu\n", RID);
  printf("        subrecord length: %u\n", subRecordLen);
  printf("        subrecord offset: %u\n", subRecordOffset);
  subRecordBuffer = recordDataBuffer + 12;
  rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
  break;
case 120:
  printf("      function ID: Update Record\n");
  oldRID = *(sqluint32 *)(recordDataBuffer + 2);
  oldSubRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
  oldSubRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
  newRID = *(sqluint32 *)(recordDataBuffer +
                          12 + oldSubRecordLen + recordHeaderSize + 2);
  newSubRecordLen = *(sqluint16 *)(recordDataBuffer +
                                   12 + oldSubRecordLen +
                                   recordHeaderSize + 6);
  newSubRecordOffset =
    *(sqluint16 *)(recordDataBuffer + 12 + oldSubRecordLen +
                   recordHeaderSize + 10);
  printf("        oldRID: %lu\n", oldRID);
  printf("        old subrecord length: %u\n", oldSubRecordLen);
  printf("        old subrecord offset: %u\n",
         oldSubRecordOffset);
  oldSubRecordBuffer = recordDataBuffer + 12;
  rc = LogSubRecordDisplay(oldSubRecordBuffer, oldSubRecordLen);
  printf("        newRID: %lu\n", newRID);
  printf("        new subrecord length: %u\n", newSubRecordLen);
  printf("        new subrecord offset: %u\n",
         newSubRecordOffset);
  newSubRecordBuffer = recordDataBuffer +
    12 + oldSubRecordLen + recordHeaderSize + 12;
  rc = LogSubRecordDisplay(newSubRecordBuffer, newSubRecordLen);
  break;
case 124:
  printf("      function ID:  Alter Table Attribute\n");
  alterBitMask = *(sqluint32 *)(recordDataBuffer + 2);
  alterBitValues = *(sqluint32 *)(recordDataBuffer + 6);
  if (alterBitMask & 0x00000001)
  {
```

```
      /* Изменение значения атрибута 'propagation': */
      printf("        Propagation attribute is changed to: ");
      if (alterBitValues & 0x00000001)
      {
        printf("ON\n");
      }
else
      {
        printf("OFF\n");
      }
    }
    if (alterBitMask & 0x00000002)
    {
      /* Изменение значения атрибута 'pending': */
      printf("        Pending attribute is changed to: ");
      if (alterBitValues & 0x00000002)
      {
        printf("ON\n");
      }
else
      {
        printf("OFF\n");
      }
    }
    if (alterBitMask & 0x00010000)
    {
      /* Изменение значения атрибута 'append mode': */
      printf("        Append Mode attribute is changed to: ");
      if (alterBitValues & 0x00010000)
      {
        printf("ON\n");
      }
else
      {
        printf("OFF\n");
      }
    }
    if (alterBitMask & 0x00200000)
    {
      /* Изменение значения атрибута 'LF Propagation': */
      printf("        LF Propagation attribute is changed to: ");
      if (alterBitValues & 0x00200000)
      {
        printf("ON\n");
      }
else
      {
        printf("OFF\n");
      }
    }
    if (alterBitMask & 0x00400000)
    {
      /* Изменение значения атрибута 'LOB Propagation': */
      printf("        LOB Propagation attribute is changed to: ");
      if (alterBitValues & 0x00400000)
```

```
        {
          printf("ON\n");
        }
  else
        {
          printf("OFF\n");
        }
      }
      break;
    default:
      printf("      unknown function identifier: %u\n",
             functionIdentifier);
      break;
  }

  return 0;
} /* ComplexLogRecordDisplay */

int LogSubRecordDisplay(char *recordBuffer, sqluint16 recordSize)
{
  int rc = 0;
  sqluint8 recordType;
  sqluint8 updatableRecordType;
  sqluint16 userDataFixedLength;
  char *userDataBuffer;
  sqluint16 userDataSize;

  recordType = *(sqluint8 *)(recordBuffer);
  if (recordType != 0 && recordType != 4)
  {
    printf("        Unknown subrecord type.\n");
  }
  else if (recordType == 4)
  {
    printf("        subrecord type: Special control\n");
  }
  else
  {
    /* recordType == 0 */
    printf("        subrecord type: Updatable, ");
    updatableRecordType = *(sqluint8 *)(recordBuffer + 4);
    if (updatableRecordType != 1)
    {
      printf("Internal control\n");
    }
  else
    {
      printf("Formatted user data\n");
      userDataFixedLength = *(sqluint16 *)(recordBuffer + 6);
      printf("        user data fixed length: %u\n",
             userDataFixedLength);
      userDataBuffer = recordBuffer + 8;
      userDataSize = recordSize - 8;
      rc = UserDataDisplay(userDataBuffer, userDataSize);
    }
```

```
  }

  return 0;
} /* LogSubRecordDisplay */

int UserDataDisplay(char *dataBuffer, sqluint16 dataSize)
{
  int rc = 0;

  sqluint16 line, col;

  printf("        user data:\n");

  for (line = 0; line * 10 < dataSize; line = line + 1)
  {
    printf("        ");
    for (col = 0; col < 10; col = col + 1)
    {
      if (line * 10 + col < dataSize)
      {
        printf("%02X ", dataBuffer[line * 10 + col]);
      }
  else
      {
        printf("   ");
      }
    }
    printf("*");
    for (col = 0; col < 10; col = col + 1)
    {
      if (line * 10 + col < dataSize)
      {
        if (isalpha(dataBuffer[line * 10 + col]) ||
            isdigit(dataBuffer[line * 10 + col]))
        {
          printf("%c", dataBuffer[line * 10 + col]);
        }
  else
        {
                printf(".");
        }
      }
  else
      {
        printf(" ");
      }
    }
    printf("*");
       printf("\n");
  }

  return 0;
} /* UserDataDisplay */

int DbRecoveryHistoryFileRead(char dbAlias[])
```

```
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2HistoryOpenStruct dbHistoryOpenParam;
  sqluint32 numEntries;
  sqluint16 recoveryHistoryFileHandle;
  sqluint32 entryNb;
  struct db2HistoryGetEntryStruct dbHistoryEntryGetParam;
  struct db2HistoryData histEntryData;

  printf("\n*********************************************\n");
  printf("*** READ A DATABASE RECOVERY HISTORY FILE ***\n");
  printf("*********************************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  db2HistoryOpenScan -- Open Recovery History File Scan\n");
  printf("  db2HistoryGetEntry -- Get Next Recovery History File Entry\n");
  printf("  db2HistoryCloseScan -- Close Recovery History File Scan\n");
  printf("TO READ A DATABASE RECOVERY HISTORY FILE.\n");

  /* initialize the data structures */
  dbHistoryOpenParam.piDatabaseAlias = dbAlias;
  dbHistoryOpenParam.piTimestamp = NULL;
  dbHistoryOpenParam.piObjectName = NULL;
  dbHistoryOpenParam.iCallerAction = DB2HISTORY_LIST_HISTORY;

  dbHistoryEntryGetParam.pioHistData = &histEntryData;
  dbHistoryEntryGetParam.iCallerAction = DB2HISTORY_GET_ALL;
  rc = HistoryEntryDataFieldsAlloc(&histEntryData);
      if (rc != 0)
  {
        return rc;
  }

  /*****************************************************/
  /* ОТКРЫТИЕ ФАЙЛА ХРОНОЛОГИИ ВОССТАНОВЛЕНИЙ БАЗЫ ДАННЫХ */
  /*****************************************************/
  printf("\n  Open recovery history file for '%s' database.\n", dbAlias);

  /* Открытие файла хронологии восстановлений для просмотра */
  db2HistoryOpenScan(db2Version710, &dbHistoryOpenParam, &sqlca);
  DB2_API_CHECK("database recovery history file -- open");

  numEntries = dbHistoryOpenParam.oNumRows;

  /* dbHistoryOpenParam.oHandle возвращает хэндл для доступа к просмотру */
  recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
  dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;

  /************************************************/
  /* ЧТЕНИЕ ЗАПИСИ В ФАЙЛЕ ХРОНОЛОГИИ ВОССТАНОВЛЕНИЙ */
  /************************************************/
  for (entryNb = 0; entryNb < numEntries; entryNb = entryNb + 1)
  {
    printf("\n  Read entry number %u.\n", entryNb);
```

```
    /* получение следующей записи из файла хронологии восстановлений */
    db2HistoryGetEntry(db2Version710, &dbHistoryEntryGetParam, &sqlca);
    DB2_API_CHECK("database recovery history file entry -- read")

    /* Вывод записей в файле хронологии восстановлений */
    printf("\n  Display entry number %u.\n", entryNb);
    rc = HistoryEntryDisplay(histEntryData);
    }

    /*******************************************************/
    /* ЗАКРЫТИЕ ФАЙЛА ХРОНОЛОГИИ ВОССТАНОВЛЕНИЙ БАЗЫ ДАННЫХ */
    /*******************************************************/
    printf("\n  Close recovery history file for '%s' database.\n", dbAlias);

    /* API db2HistoryCloseScan заканчивает просмотр файла хронологии восстановлений
       и освобождает ресурсы DB2, требовавшиеся для просмотра. */
    db2HistoryCloseScan(db2Version710, &recoveryHistoryFileHandle, &sqlca);
    DB2_API_CHECK("database recovery history file -- close");

    /* освобождение выделенной памяти */
    rc = HistoryEntryDataFieldsFree(&histEntryData);

    return 0;
  } /* DbRecoveryHistoryFileRead */

  int HistoryEntryDataFieldsAlloc(struct db2HistoryData *pHistEntryData)
  {
    int rc = 0;
    sqluint32 tsNb;

    strcpy(pHistEntryData->ioHistDataID, "SQLUHINF");

    pHistEntryData->oObjectPart.pioData = malloc(17 + 1);
    pHistEntryData->oObjectPart.iLength = 17 + 1;

    pHistEntryData->oEndTime.pioData = malloc(12 + 1);
    pHistEntryData->oEndTime.iLength = 12 + 1;

    pHistEntryData->oFirstLog.pioData = malloc(8 + 1);
    pHistEntryData->oFirstLog.iLength = 8 + 1;

    pHistEntryData->oLastLog.pioData = malloc(8 + 1);
    pHistEntryData->oLastLog.iLength = 8 + 1;

    pHistEntryData->oID.pioData = malloc(128 + 1);
    pHistEntryData->oID.iLength = 128 + 1;

    pHistEntryData->oTableQualifier.pioData = malloc(128 + 1);
    pHistEntryData->oTableQualifier.iLength = 128 + 1;

    pHistEntryData->oTableName.pioData = malloc(128 + 1);
    pHistEntryData->oTableName.iLength = 128 + 1;

    pHistEntryData->oLocation.pioData = malloc(128 + 1);
    pHistEntryData->oLocation.iLength = 128 + 1;
```

```
  pHistEntryData->oComment.pioData = malloc(128 + 1);
  pHistEntryData->oComment.iLength = 128 + 1;

  pHistEntryData->oCommandText.pioData = malloc(128 + 1);
  pHistEntryData->oCommandText.iLength = 128 + 1;

  pHistEntryData->poEventSQLCA =
    (struct sqlca *)malloc(sizeof(struct sqlca));

  pHistEntryData->poTablespace = (db2Char *)malloc(3 * sizeof(db2Char));
  for (tsNb = 0; tsNb < 3; tsNb = tsNb + 1)
  {
    pHistEntryData->poTablespace[tsNb].pioData = malloc(18 + 1);
    pHistEntryData->poTablespace[tsNb].iLength = 18 + 1;
  }

  pHistEntryData->iNumTablespaces = 3;

  return 0;
} /* HistoryEntryDataFieldsAlloc */

int HistoryEntryDisplay(struct db2HistoryData histEntryData)
{
  int rc = 0;
  char buf[129];
  sqluint32 tsNb;

  memcpy(buf, histEntryData.oObjectPart.pioData,
              histEntryData.oObjectPart.oLength);
  buf[histEntryData.oObjectPart.oLength] = '\0';
  printf("    object part: %s\n", buf);

  memcpy(buf, histEntryData.oEndTime.pioData,
              histEntryData.oEndTime.oLength);
  buf[histEntryData.oEndTime.oLength] = '\0';
  printf("    end time: %s\n", buf);

  memcpy(buf, histEntryData.oFirstLog.pioData,
              histEntryData.oFirstLog.oLength);
  buf[histEntryData.oFirstLog.oLength] = '\0';
  printf("    first log: %s\n", buf);

  memcpy(buf, histEntryData.oLastLog.pioData,
              histEntryData.oLastLog.oLength);
  buf[histEntryData.oLastLog.oLength] = '\0';
  printf("    last log: %s\n", buf);

  memcpy(buf, histEntryData.oID.pioData, histEntryData.oID.oLength);
  buf[histEntryData.oID.oLength] = '\0';
  printf("    ID: %s\n", buf);

  memcpy(buf, histEntryData.oTableQualifier.pioData,
              histEntryData.oTableQualifier.oLength);
  buf[histEntryData.oTableQualifier.oLength] = '\0';
```

```
  printf("    table qualifier: %s\n", buf);

  memcpy(buf, histEntryData.oTableName.pioData,
              histEntryData.oTableName.oLength);
  buf[histEntryData.oTableName.oLength] = '\0';
  printf("    table name: %s\n", buf);

  memcpy(buf, histEntryData.oLocation.pioData,
              histEntryData.oLocation.oLength);
  buf[histEntryData.oLocation.oLength] = '\0';
  printf("    location: %s\n", buf);

  memcpy(buf, histEntryData.oComment.pioData,
              histEntryData.oComment.oLength);
  buf[histEntryData.oComment.oLength] = '\0';
  printf("    comment: %s\n", buf);

  memcpy(buf, histEntryData.oCommandText.pioData,
              histEntryData.oCommandText.oLength);
  buf[histEntryData.oCommandText.oLength] = '\0';
  printf("    command text: %s\n", buf);
  printf("    history file entry ID: %u\n", histEntryData.oEID.ioHID);
  printf("    table spaces:\n");

  for (tsNb = 0; tsNb < histEntryData.oNumTablespaces; tsNb = tsNb + 1)
  {
    memcpy(buf, histEntryData.poTablespace[tsNb].pioData,
                histEntryData.poTablespace[tsNb].oLength);
    buf[histEntryData.poTablespace[tsNb].oLength] = '\0';
    printf("      %s\n", buf);
  }

  printf("    type of operation: %c\n", histEntryData.oOperation);
  printf("    granularity of the operation: %c\n", histEntryData.oObject);
  printf("    operation type: %c\n", histEntryData.oOptype);
  printf("    entry status: %c\n", histEntryData.oStatus);
  printf("    device type: %c\n", histEntryData.oDeviceType);
  printf("    SQLCA:\n");
  printf("      sqlcode: %ld\n", histEntryData.poEventSQLCA->sqlcode);
  memcpy(buf, histEntryData.poEventSQLCA->sqlstate, 5);
  buf[5] = '\0';
  printf("      sqlstate: %s\n", buf);
  memcpy(buf, histEntryData.poEventSQLCA->sqlerrmc,
              histEntryData.poEventSQLCA->sqlerrml);
  buf[histEntryData.poEventSQLCA->sqlerrml] = '\0';
  printf("      message: %s\n", buf);

  return 0;
} /* HistoryEntryDisplay */

int HistoryEntryDataFieldsFree(struct db2HistoryData *pHistEntryData)
{
  int rc = 0;
  sqluint32 tsNb;
```

```
  free(pHistEntryData->oObjectPart.pioData);
  free(pHistEntryData->oEndTime.pioData);
  free(pHistEntryData->oFirstLog.pioData);
  free(pHistEntryData->oLastLog.pioData);
  free(pHistEntryData->oID.pioData);
  free(pHistEntryData->oTableQualifier.pioData);
  free(pHistEntryData->oTableName.pioData);
  free(pHistEntryData->oLocation.pioData);
  free(pHistEntryData->oComment.pioData);
  free(pHistEntryData->oCommandText.pioData);
  free(pHistEntryData->poEventSQLCA);

  for (tsNb = 0; tsNb < 3; tsNb = tsNb + 1)
  {
    free(pHistEntryData->poTablespace[tsNb].pioData);
  }

  free(pHistEntryData->poTablespace);

  return 0;
} /* HistoryEntryDataFieldsFree */

int DbFirstRecoveryHistoryFileEntryUpdate(char dbAlias[],
                                          char user[],
                                          char pswd[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2HistoryOpenStruct dbHistoryOpenParam;
  sqluint16 recoveryHistoryFileHandle;
  struct db2HistoryGetEntryStruct dbHistoryEntryGetParam;
  struct db2HistoryData histEntryData;
  char newLocation[DB2HISTORY_LOCATION_SZ + 1];
  char newComment[DB2HISTORY_COMMENT_SZ + 1];
  struct db2HistoryUpdateStruct dbHistoryUpdateParam;

  printf("\n**************************************************\n");
  printf("*** UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY ***\n");
  printf("**************************************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  db2HistoryOpenScan -- Open Recovery History File Scan\n");
  printf("  db2HistoryGetEntry -- Get Next Recovery History File Entry\n");
  printf("  db2HistoryUpdate -- Update Recovery History File\n");
  printf("  db2HistoryCloseScan -- Close Recovery History File Scan\n");
  printf("TO UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY.\n");

  /* инициализация структур данных */
  dbHistoryOpenParam.piDatabaseAlias = dbAlias;
  dbHistoryOpenParam.piTimestamp = NULL;
  dbHistoryOpenParam.piObjectName = NULL;
  dbHistoryOpenParam.iCallerAction = DB2HISTORY_LIST_HISTORY;
  dbHistoryEntryGetParam.pioHistData = &histEntryData;
  dbHistoryEntryGetParam.iCallerAction = DB2HISTORY_GET_ALL;
  rc = HistoryEntryDataFieldsAlloc(&histEntryData);
      if (rc != 0)
```

```
{
      return rc;
}

/*******************************************************/
/* ОТКРЫТИЕ ФАЙЛА ХРОНОЛОГИИ ВОССТАНОВЛЕНИЙ БАЗЫ ДАННЫХ */
/*******************************************************/
printf("\n  Open the recovery history file for '%s' database.\n", dbAlias);

/* API db2HistoryOpenScan запускает просмотр файла хронологии восстановлений */
db2HistoryOpenScan(db2Version710, &dbHistoryOpenParam, &sqlca);
DB2_API_CHECK("database recovery history file -- open");

/* dbHistoryOpenParam.oHandle возвращает хэндл для доступа к просмотру */
recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;

/*******************************************************/
/* ЧТЕНИЕ ПЕРВОЙ ЗАПИСИ В ФАЙЛЕ ХРОНОЛОГИИ ВОССТАНОВЛЕНИЙ */
/*******************************************************/
printf("\n  Read the first entry in the recovery history file.\n");

/* API db2HistoryGetEntry получает следующую запись из файла
   хронологии восстановлений. */
db2HistoryGetEntry(db2Version710, &dbHistoryEntryGetParam, &sqlca);
DB2_API_CHECK("first recovery history file entry -- read");
printf("\n  Display the first entry.\n");

/* HistoryEntryDisplay - это функция поддержки, используемая для вывода записей
   в файле хронологии восстановлений. */
rc = HistoryEntryDisplay(histEntryData);

/* изменение первой записи файла хронологии */
rc = DbConn(dbAlias, user, pswd);
    if (rc != 0)
{
      return rc;
}

strcpy(newLocation, "this is the NEW LOCATION");
strcpy(newComment, "this is the NEW COMMENT");
printf("\n  Update the first entry in the history file:\n");
printf("    new location = '%s'\n", newLocation);
printf("    new comment = '%s'\n", newComment);
dbHistoryUpdateParam.piNewLocation = newLocation;
dbHistoryUpdateParam.piNewDeviceType = NULL;
dbHistoryUpdateParam.piNewComment = newComment;
dbHistoryUpdateParam.iEID.ioNode = histEntryData.oEID.ioNode;
dbHistoryUpdateParam.iEID.ioHID = histEntryData.oEID.ioHID;

/* API db2HistoryUpdate может использоваться для изменения положения,
   типа устройства или комментария в записи файла хронологии. */

/* Этот API вызывается для изменения положения и комментария первой
   записи в файле хронологии: */
```

```
db2HistoryUpdate(db2Version710, &dbHistoryUpdateParam, &sqlca);
DB2_API_CHECK("first history file entry -- update");

rc = DbDisconn(dbAlias);
    if (rc != 0)
{
      return rc;
}

/*******************************************************/
/* ЗАКРЫТИЕ ФАЙЛА ХРОНОЛОГИИ ВОССТАНОВЛЕНИЙ БАЗЫ ДАННЫХ */
/*******************************************************/
printf("\n  Close recovery history file for '%s' database.\n", dbAlias);

/* API db2HistoryCloseScan заканчивает просмотр файла хронологии восстановлений
   и освобождает ресурсы DB2, требовавшиеся для просмотра. */
db2HistoryCloseScan(db2Version710, &recoveryHistoryFileHandle, &sqlca);
DB2_API_CHECK("database recovery history file -- close");


/*****************************************************************/
/* ПОВТОРНОЕ ОТКРЫТИЕ ФАЙЛА ХРОНОЛОГИИ ВОССТАНОВЛЕНИЙ БАЗЫ ДАННЫХ */
/*****************************************************************
printf("\n  Open the recovery history file for '%s' database.\n", dbAlias);

/* запускает просмотр файла хронологии восстановлений */
db2HistoryOpenScan(db2Version710, &dbHistoryOpenParam, &sqlca);
DB2_API_CHECK("database recovery history file -- open");

recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;

dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;
printf("\n  Read the first recovery history file entry.\n");

/*************************************************************************/
/* ЧТЕНИЕ ПЕРВОЙ ЗАПИСИ В ФАЙЛЕ ХРОНОЛОГИИ ВОССТАНОВЛЕНИЙ ПОСЛЕ ИЗМЕНЕНИЯ */
/*************************************************************************/
db2HistoryGetEntry(db2Version710, &dbHistoryEntryGetParam, &sqlca);
DB2_API_CHECK("first recovery history file entry -- read");

printf("\n  Display the first entry.\n");
rc = HistoryEntryDisplay(histEntryData);

/*******************************************************/
/* ЗАКРЫТИЕ ФАЙЛА ХРОНОЛОГИИ ВОССТАНОВЛЕНИЙ БАЗЫ ДАННЫХ */
/*******************************************************/
printf("\n  Close the recovery history file for '%s' database.\n",
       dbAlias);

/* завершение просмотра файла хронологии восстановлений */
db2HistoryCloseScan(db2Version710, &recoveryHistoryFileHandle, &sqlca);
DB2_API_CHECK("database recovery history file -- close");

/* освобождение выделенной памяти */
rc = HistoryEntryDataFieldsFree(&histEntryData);
```

```
  return 0;
} /* DbFirstRecoveryHistoryFileEntryUpdate */

int DbRecoveryHistoryFilePrune(char dbAlias[], char user[], char pswd[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2PruneStruct histPruneParam;
  char timeStampPart[14 + 1];

  printf("\n*************************************\n");
  printf("*** PRUNE THE RECOVERY HISTORY FILE ***\n");
  printf("*************************************\n");
  printf("\nUSE THE DB2 API:\n");
  printf("  db2Prune -- Prune Recovery History File\n");
  printf("AND THE SQL STATEMENTS:\n");
  printf("  CONNECT\n");
  printf("  CONNECT RESET\n");
  printf("TO PRUNE THE RECOVERY HISTORY FILE.\n");

  /* Соединение с базой данных: */
  rc = DbConn(dbAlias, user, pswd);
      if (rc != 0)
  {
        return rc;
  }

  /* Сокращение файла хронологии восстановлений: */
  printf("\n  Prune the recovery history file for '%s' database.\n",
         dbAlias);

  /* timeStampPart - это указатель строки, задающей отметку времени
     или последовательный номер журнала. Отметка времени используется здесь
     для выбора удаляемых записей. Будут удалены все записи до заданной отметки
     времени включительно. */
  histPruneParam.piString = timeStampPart;
  strcpy(timeStampPart, "2010"); /* 2010 год */

  /* Действие DB2PRUNE_ACTION_HISTORY удаляет записи файла хронологии: */
  histPruneParam.iAction = DB2PRUNE_ACTION_HISTORY;

  /* Опция DB2PRUNE_OPTION_FORCE принудительно удаляет последнюю резервную копию: */
  histPruneParam.iOptions = DB2PRUNE_OPTION_FORCE;

  /* db2Prune можно вызывать для удаления записей из файла журнала хронологии
     или файлов журналов из активного пути журналов. Здесь мы вызываем dbPrune
     для удаления записей из файла хронологии восстановлений.
     Для сокращения файла хронологии восстановлений у вас должны быть
     полномочия SYSADM, SYSCTRL, SYSMAINT или DBADM. */
  db2Prune(db2Version710, &histPruneParam, &sqlca);
  DB2_API_CHECK("recovery history file -- prune");

  /* Отсоединение от базы данных: */
  rc = DbDisconn(dbAlias);
```

```
      if (rc != 0)
  {
        return rc;
  }

  return 0;
} /* DbRecoveryHistoryFilePrune */
```

Программа примера со встроенным SQL (dbrecov.sqc)

Приложение F. Сценарий восстановления CLP

Следующий сценарий команд DB2 показывает, как использовать команды CLP для:

- Резервного копирования базы данных
- Восстановления базы данных из резервной копии
- Восстановления базы данных из резервной копии с повтором транзакций

Проверьте, что база данных SAMPLE существует и в данное время не используется. Подробную информацию о базе данных SAMPLE смотрите в справочнике *SQL Reference*. Общую информацию о процессоре командной строки DB2 смотрите в руководстве *Command Reference*.

Описываются версии сценария как для Windows, так и для UNIX.

Пример командного сценария для операционной системы Windows

Чтобы запустить сценарий в Windows NT или Windows 2000:

1. Сохраните сценарий в файле, например, под именем `backrest.db2`.
2. Если менеджер баз данных еще не запущен, введите команду **db2start** в командном окне DB2. Чтобы в операционной системе Windows открыть окно DB2 с поддержкой CLP и инициализировать среду командной строки DB2, введите в командной строке **db2cmd**.
3. Введите `db2 -f backrest.db2 -t`.

Пример вывода, возвращаемого этим сценарием:

```
D:\>db2 -f backrest.db2 -t
Это сценарий CLP: backrest.db2


Удаляются старые резервные копии SAMPLE...

обработка SAMPLE.0\DB2\NODE0000\CATN0000
обработка 20010403

Изменение значения параметра конфигурации базы данных LOGRETAIN на 'ON'...

DB20000I  Команда UPDATE DATABASE CONFIGURATION выполнена успешно.
DB21026I  Для большинства параметров конфигурации до того, как сделанные
          изменения вступят в силу, все программы должны быть отсоединены
          от этой базы данных.


Резервное копирование базы данных SAMPLE...
```

© Copyright IBM Corp. 2001

```
Резервное копирование завершилось успешно. Отметка времени копии : 20010403131027


Восстановление базы данных SAMPLE как TESTBACK (1-й проход)...

SQL1277W  При восстановлении в табличных пространствах обнаружен один или
несколько недоступных контейнеров, или контейнеров в состоянии с неопределенным
размером.
DB20000I  Команда RESTORE DATABASE выполнена успешно.

Составление списка табличных пространств для базы данных TESTBACK...


Табличные пространства текущей базы данных

 ID табличного пространства           = 0
 Имя                                  = SYSCATSPACE
 Тип                                  = Пространство, управляемое системой
 Содержимое                           = Любые данные
 Состояние                            = 0x2001100
Подробное объяснение:
Отложенное восстановление
Должна быть определена память
Может быть определена память

 ID табличного пространства           = 1
 Имя                                  = TEMPSPACE1
 Тип                                  = Пространство, управляемое системой
 Содержимое                           = Временные данные системы
 Состояние                            = 0x2001100
Подробное объяснение:
Отложенное восстановление
Должна быть определена память
Может быть определена память

 ID табличного пространства           = 2
 Имя                                  = USERSPACE1
 Тип                                  = Пространство, управляемое системой
 Содержимое                           = Любые данные
 Состояние                            = 0x2001100
Подробное объяснение:
Отложенное восстановление
Должна быть определена память
Может быть определена память


Определение новых контейнеров табличного пространства для табличного
пространства 2...

DB20000I  Команда SET TABLESPACE CONTAINERS выполнена успешно.

Составление списка контейнеров табличного пространства для табличного
пространства 2 (база данных TESTBACK)...
```

```
Контейнеры табличных пространств для табличного пространства 2
 ID контейнера                        = 0
 Имя                                  = c:\ts2con1
 Тип                                  = Путь



Восстановление базы данных SAMPLE как TESTBACK (2-й проход)...

DB20000I  Команда RESTORE DATABASE выполнена успешно.

Повтор транзакций для базы данных TESTBACK...


                                      Состояние повтора

 Алиас входной базы данных            = testback
 Число узлов с возвращенным состоянием  = 1

 Номер узла                           = 0
 Состояние повтора транзакций         = не отложенное
 Следующий файл журнала на чтение     =
 Обработано файлов журналов           = -
 Последняя принятая транзакция        = 2001-04-03-03.16.07.000000

DB20000I  Команда ROLLFORWARD выполнена успешно.

Отбрасывание базы данных TESTBACK...

DB20000I  Команда DROP DATABASE выполнена успешно.

Завершение серверной части процесса процессора командной строки...

DB20000I  Команда TERMINATE выполнена успешно.


D:\>
```

Исходный текст сценария:

```
-- Перед выполнением убедитесь, что:
--   Запущен менеджер баз данных
--   База данных SAMPLE существует и в данное время не используется.

-- Запустите сценарий, введя:
--   db2 -f backrest.db2 -t
--     где -f заставляет процессор командной строки читать ввод команды
--            из файла вместо стандартного ввода, а
--         -t заставляет процессор командной строки использовать точку
--            с запятой (;) в качестве символа - ограничителя оператора.

!ECHO Это сценарий CLP: backrest.db2;
```

Пример командного сценария для операционной системы Windows

```
-- Убедитесь, что для переменной реестра профиля DB2 DB2_ENABLE_LDAP
--   установлено значение 'NO':
!db2set DB2_ENABLE_LDAP=NO;

!ECHO Удаляются старые резервные копии SAMPLE...;
!rd! SAMPLE.0\DB2\NODE0000\CATN0000;

!ECHO Изменение значения параметра конфигурации базы данных LOGRETAIN на 'ON'...;
update db cfg for sample using logretain on;

!ECHO Резервное копирование базы данных SAMPLE...;
backup db sample;

!ECHO Восстановление базы данных SAMPLE как TESTBACK (1-й проход)...;
restore db sample into testback redirect;

!ECHO Составление списка табличных пространств для базы данных TESTBACK...;
list tablespaces;

!ECHO Определение новых контейнеров табличного пространства для табличного
пространства 2...;
set tablespace containers for 2 using (path "c:\ts2con1");

!ECHO Составление списка контейнеров табличного пространства для табличного
пространства 2 (база данных TESTBACK)...;
list tablespace containers for 2;

!ECHO Восстановление базы данных SAMPLE как TESTBACK (2-й проход)...;
restore db sample continue;

!ECHO Повтор транзакций для базы данных TESTBACK...;
rollforward db testback stop;

!ECHO Отбрасывается база данных TESTBACK...;
drop db testback;

!ECHO Завершение серверной части процесса процессора командной строки...;
terminate;

-- Конец файла
```

Пример сценария команд для систем на основе UNIX

Чтобы запустить сценарий в системе на основе UNIX:

1. Сохраните сценарий в файле с именем, например, `backrest.db2`.
2. Если менеджер баз данных еще не запущен, введите команду **db2start** в командной строке.
3. Введите `db2 -f backrest.db2 -t`.

Пример вывода, возвращаемого этим сценарием:

```
sunfish /export/home2/falexand/samples/clp>db2 -f backrest.db2 -t
Это сценарий CLP: backrest.db2

Удаляются старые резервные копии SAMPLE...

Изменение значения параметра конфигурации базы данных LOGRETAIN на 'ON'...
DB20000I  Команда UPDATE DATABASE CONFIGURATION выполнена успешно.
DB21026I  Для большинства параметров конфигурации до того, как сделанные изменения
вступят в силу, все программы должны быть отсоединены от этой базы данных.


Резервное копирование базы данных SAMPLE...

Резервное копирование завершилось успешно. Отметка времени копии : 20010531172525


Восстановление базы данных SAMPLE как TESTBACK (1-й проход)...
SQL1277W  При восстановлении в табличных пространствах обнаружен один или
несколько недоступных контейнеров, или контейнеров в состоянии с неопределенным
размером.
DB20000I  Команда RESTORE DATABASE выполнена успешно.

Составление списка табличных пространств для базы данных TESTBACK...

Табличные пространства текущей базы данных

 ID табличного пространства           = 0
 Имя                                  = SYSCATSPACE
 Тип                                  = Пространство, управляемое системой
 Содержимое                           = Любые данные
 Состояние                            = 0x2001100
Подробное объяснение:
Отложенное восстановление
Должна быть определена память
Может быть определена память

 ID табличного пространства           = 1
 Имя                                  = TEMPSPACE1
 Тип                                  = Пространство, управляемое системой
 Содержимое                           = Временные данные системы
 Состояние                            = 0x2001100
Подробное объяснение:
Отложенное восстановление
Должна быть определена память
Может быть определена память

 ID табличного пространства           = 2
 Имя                                  = USERSPACE1
 Тип                                  = Пространство, управляемое системой
 Содержимое                           = Любые данные
 Состояние                            = 0x2001100
Подробное объяснение:
Отложенное восстановление
Должна быть определена память
Может быть определена память
```

```
Определение новых контейнеров табличного пространства для табличного
пространства 2...
DB20000I  Команда SET TABLESPACE CONTAINERS выполнена успешно.

Составление списка контейнеров табличного пространства для табличного
пространства 2 (база данных TESTBACK)...

              Контейнеры табличных пространств для табличного пространства 2
 ID контейнера                        = 0
 Имя                                  = /export/home2/falexand/falexand...
                                          .../NODE0000/SQL00020/ts2con1
 Тип                                  = Путь



Восстановление базы данных SAMPLE как TESTBACK (2-й проход)...
DB20000I  Команда RESTORE DATABASE выполнена успешно.

Повтор транзакций для базы данных TESTBACK...

                                       Состояние повтора транзакций

 Алиас входной базы данных             = testback
 Число узлов с возвращенным состоянием  = 1

 Номер узла                            = 0
 Состояние повтора трансакций           = не отложенное
 Следующий файл журнала на чтение      =
 Обработано файлов журналов            = -
 Последняя принятая транзакция         = 2001-05-29-21.20.16.000000

DB20000I  Команда ROLLFORWARD выполнена успешно.

Отбрасывание базы данных TESTBACK...
DB20000I  Команда DROP DATABASE выполнена успешно.

Завершение серверной части процесса процессора командной строки...
DB20000I  Команда TERMINATE выполнена успешно.
```

Исходный текст сценария:

```
-- Перед выполнением убедитесь, что:
--   Запущен менеджер баз данных
--   База данных SAMPLE существует и в данное время не используется.

-- Запустите сценарий, введя:
--   db2 -f backrest.db2 -t
--     где -f заставляет процессор командной строки читать ввод команды
--            из файла вместо стандартного ввода, а
--         -t заставляет процессор командной строки использовать точку
--            с запятой (;) в качестве символа - ограничителя оператора.

echo Это сценарий CLP: backrest.db2;
```

```
-- Убедитесь, что для переменной реестра профиля DB2 DB2_ENABLE_LDAP
--   установлено значение 'NO':
!db2set DB2_ENABLE_LDAP=NO;

echo Удаляются старые резервные копии SAMPLE...;
!rm -f ./SAMPLE.0.*;

echo Изменение значения параметра конфигурации базы данных LOGRETAIN на 'ON'...;
update db cfg for sample using logretain on;

echo Резервное копирование базы данных SAMPLE...;
backup db sample;

echo Восстановление базы данных SAMPLE как TESTBACK (1-й проход)...;
restore db sample into testback redirect;

echo Составление списка табличных пространств для базы данных TESTBACK...;
list tablespaces;

echo Определение новых контейнеров табличного пространства для табличного
пространства 2...;
set tablespace containers for 2 using (path "ts2con1");

echo Составление списка контейнеров табличного пространства для табличного
пространства 2 (база данных TESTBACK)...;
list tablespace containers for 2;

echo Восстановление базы данных SAMPLE как TESTBACK (2-й проход)...;
restore db sample continue;

echo Повтор транзакций для базы данных TESTBACK...;
rollforward db testback stop;

echo Отбрасывается база данных TESTBACK...;
drop db testback;

echo Завершение серверной части процесса процессора командной строки...;
terminate;

-- Конец файла
```

Приложение G. Tivoli Storage Manager

При вызове утилит DB2 резервного копирования и восстановления можно указать, что для резервного копирования или восстановления вы хотите использовать программный продукт Tivoli Storage Manager (TSM, прежнее название - ADSM). С DB2 можно использовать клиент TSM client Версии 3.1.x.3 и новее.

Настройка клиента Tivoli Storage Manager на платформах на основе UNIX

Чтобы менеджер баз данных смог использовать опцию TSM, необходимо выполнить следующие настройки:

1. В средах SunOS и Solaris выполните следующие действия. (На других платформах на основе UNIX начинайте с шага 2.)
   a. Проверьте, что установлен необходимый уровень операционной системы: SunOS 5.5.1 или Solaris 2.5.1.
   b. Установите клиент TSM client Версии 3.1.x.3 или новее. Перед установкой этой версии клиента проверьте, что вы удалили *все* предыдущие программные пакеты TSM.
   c. Проверьте, что TSM установлен в каталоги `/opt/IBMDSMap5`, `/opt/IBMDSMba5` и `/opt/IBMDSMsa5`.
   d. Создайте следующие символические связи в каталоге `/usr/lib`, если они еще не существуют:

      ```
      libApiDS.so -> libApiDS.so.1
      libApiDS.so.1 -> /opt/IBMDSMap5/api/libApiDS.so.2
      ```

2. Создайте или модифицируйте файл опций конфигурации пользователя TSM `/usr/sbin/dsm.opt` и файл опций конфигурации системы TSM `/usr/sbin/dsm.sys`, соответствующие вашей среде.
3. В средах SunOS и Solaris выполните следующие действия. (На других платформах на основе UNIX продолжайте с шага 4.)
   a. Скопируйте `/usr/sbin/dsm.opt` и `/usr/sbin/dsm.sys` в каталог `/opt/IBMDSMap5`.
   b. Скопируйте `/opt/IBMDSMap5/solaris/dsmaptica` в каталог `/opt/IBMDSMap5`.
4. Установите используемые TSM переменные среды:

   DSMI_DIR Идентифицирует путь к пользовательскому каталогу, в котором расположен файл доверенного агента API (`dsmapicta` или `dsmtca`). Для сред SunOS и Solaris следует устанавливать `/opt/IBMDSMap5`.

© Copyright IBM Corp. 2001

DSMI_CONFIG
Идентифицирует путь к пользовательскому каталогу с файлом `dsm.opt`, в котором содержатся опции пользователя TSM. В отличие от двух других переменных, эта переменная должна содержать полное имя файла с путем к нему. Для сред SunOS и Solaris следует устанавливать `/opt/IBMDSMap5/dsm.opt`.

DSMI_LOG Идентифицирует путь к пользовательскому каталогу, в котором будет создаваться журнал ошибок (`dsierror.log`).

5. Установите пароль TSM.

   Чтобы клиент Tivoli мог взаимодействовать с сервером TSM, у него должен быть пароль для этого сервера. Исполняемый файл `dsmapipw` устанавливается в каталог `INSTHOME/sqllib/adsm` владельца экземпляра. Этот файл позволяет устанавливать и изменять пароль TSM.

   Чтобы выполнить команду `dsmapipw`, вы должны быть зарегистрированы как пользователь “root”. При выполнении этой команды у вас запрашивается следующая информация:

   - *Старый пароль* - текущий пароль для узла TSM в том виде, в котором он распознается сервером TSM. При первом выполнении этой команды это будет пароль, предоставленный администратором TSM во время регистрации вашего узла на сервере TSM.
   - *Новый пароль* - новый пароль для узла, который сохраняется на сервере TSM. (Обратите внимание на то, что новый пароль надо ввести дважды для контроля ошибок при вводе.)

   Примечание: Пользователям, вызывающим утилиты резервного копирования и восстановления, знать этот пароль не требуется. Команду `dsmapipw` надо выполнить только для задания пароля для начального соединения; после этого пароль переустанавливается сервером TSM.

6. Если запущен менеджер баз данных:
   - Остановите менеджер баз данных, введя команду **db2stop**.
   - Запустите менеджер баз данных, введя команду **db2start**.

Настройка клиента Tivoli Storage Manager на других платформах

Чтобы менеджер баз данных смог использовать опцию TSM, необходимо выполнить следующие настройки:

1. Установите используемые TSM переменные среды:

   DSMI_DIR Идентифицирует путь к пользовательскому каталогу, в котором расположен файл доверенного агента API (`dsmapicta` или `dsmtca`).

DSMI_CONFIG
Идентифицирует путь к пользовательскому каталогу с файлом `dsm.opt`, в котором содержатся опции пользователя TSM. В отличие от двух других переменных, эта переменная должна содержать полное имя файла с путем к нему.

DSMI_LOG Идентифицирует путь к пользовательскому каталогу, в котором будет создаваться журнал ошибок (`dsierror.log`).

2. Если это применимо в вашей операционной системе, создайте (или измените) файл опций конфигурации системы TSM (`dsm.sys`).
3. Создайте (или измените) файл опций конфигурации пользователя TSM `dsm.opt`. На этот файл указывает переменная среды `DSMI_CONFIG`.
4. Установите пароль TSM.

   Чтобы клиент Tivoli мог взаимодействовать с сервером TSM, у него должен быть пароль для этого сервера. Исполняемый файл `dsmapipw` устанавливается в каталог `\sqllib\adsm` владельца экземпляра. Этот файл позволяет устанавливать и изменять пароль TSM.

   Чтобы выполнить команду `dsmapipw`, вы должны быть зарегистрированы как локальный администратор. При запуске этой команды у вас запрашивается следующая информация:

   - *Старый пароль* - текущий пароль для узла TSM в том виде, в котором он распознается сервером TSM. При первом вызове этой команды это будет пароль, предоставленный администратором TSM во время регистрации вашего узла на сервере TSM.
   - *Новый пароль* - новый пароль для узла, который сохраняется на сервере TSM. (Обратите внимание на то, что новый пароль надо ввести дважды для контроля ошибок при вводе.)

   Примечание: Пользователям, вызывающим утилиты резервного копирования и восстановления, знать этот пароль не требуется. Команду `dsmapipw` надо выполнить только для задания пароля для начального соединения; после этого пароль переустанавливается сервером TSM.
5. Если запущен менеджер баз данных:
   - Остановите менеджер баз данных, введя команду **db2stop**.
   - Запустите менеджер баз данных, введя команду **db2start**.

Особенности использования Tivoli Storage Manager

Чтобы использовать отдельные возможности TSM, вам может потребоваться указать полное имя объекта, использующего одну из этих возможностей. (Помните, что в операционных системах Windows и OS/2 вместо / используется \.) Полное имя:

- Объекта резервной копии всей базы данных:
  /<database>/NODEnnnn/FULL_BACKUP.timestamp.seq_no
- Объекта резервной копии табличного пространства:
  /<database>/NODEnnnn/TSP_BACKUP.timestamp.seq_no
- Объекта копии загрузки:
  /<база_данных>/NODEnnnn/LOAD_COPY.timestamp.seq_no

где <база_данных - алиас базы данных, а NODEnnnn - номер узла. Имена, показанные в верхнем регистре, необходимо вводить именно в таком виде.

- Когда у вас есть несколько образов резервных копий, использующих одно и то же имя алиаса базы данных, различительной частью полного имени становятся отметка времени и последовательный номер. Потребуется ввести запрос TSM, чтобы определить, какую версию резервной копии использовать.
- Графический интерфейс пользователя TSM не имеет информации об отдельных образах резервных копий. Образы резервных копий помещаются в файловые пространства, которыми управляет TSM. Работа с отдельными образами резервных копий может производиться только через API TSM или через **db2adutl**, который использует эти API (смотрите раздел “db2adutl - Работа с архивными образами TSM” на стр. 326).
- Сервер TSM отключит сеанс из-за истечения срока, если клиент Tivoli не отвечает за время, указанное параметром COMMTIMEOUT в файле конфигурации этого сервера. Это может быть вызвано тремя причинами:
  - Для параметра COMMTIMEOUT на сервере TSM, возможно, установлено слишком низкое значение. Например, истечение срока может произойти во время восстановления из резервной копии при создании больших табличных пространств DMS. Рекомендуемое значение для этого параметра - 6000 секунд.
  - Буфер резервного копирования или восстановления DB2 может быть слишком велик.
  - Активность базы данных при оперативном резервном копировании может быть слишком высокой.
- Менеджер баз данных использует опцию полного резервного копирования TSM; инкрементные операции резервного копирования TSM не поддерживаются.
- Для повышения производительности используйте несколько сеансов.
- На платформах, кроме платформ на основе UNIX, утилиты резервного копирования и восстановления DB2 не допускают выполнять несколько сеансов TSM.

Текущие клиенты Tivoli в операционных системах Windows и в OS/2 являются повторно-входными, поэтому утилиты резервного копирования, восстановления или загрузки позволяют безопасно создать несколько сеансов ввода/вывода на одном компьютере. Тем не менее пользователи должны убедиться в том, что установленная версия клиента TSM поддерживает эту функцию.

При одноузловой конфигурации, когда пользователь пытается выдать команду BACKUP DATABASE типа:

```
db2 backup db sample use tsm open 3 sessions
```

DB2 определит, что TSM не поддерживает несколько сеансов, и возвратит сообщение об ошибке. (Это относится и к операциям загрузки, вызванным с опцией COPY YES и использующим TSM.)

Однако обратите внимание на то, что в конфигурации с несколькими логическими узлами в Windows NT DB2 может оказаться не в состоянии обнаружить использование нескольких сеансов на одном и том же компьютере, если каждый из логических узлов пытается создать всего один сеанс. Поэтому в конфигурациях с несколькими логическими узлами очень важно проверить, что их клиент TSM поддерживает повторные вхождения. Если для нескольких логических узлов параллельно выполняется резервное копирование, восстановление из резервной копии или загрузка с использованием TSM, DB2 позволит выполнение этой операции, если каждый из узлов пытается использовать один сеанс, даже в том случае, когда эти логические узлы расположены на одной и той же аппаратуре. Это может привести к неудачным попыткам резервного копирования и зависанию процессов загрузки, поэтому без самой последней версии клиента TSM лучше таких попыток не предпринимать.

Управление резервными копиями и архивами журналов в TSM

Утилита **db2adutl** позволяет запрашивать, извлекать и удалять образы резервных копий, файлов журнала и копий загрузки, сохраненные при помощи TSM. Эта утилита в системах на основе UNIX установлена в каталоге `INSTHOME/sqllib/misc`, а в системах Windows и OS/2 - в каталоге `\sqllib\misc`. Дополнительную информацию об этой утилите смотрите в разделе “db2adutl - Работа с архивными образами TSM” на стр. 326.

Опция QUERY позволяет получать списки образов резервных копий, копий загрузки или журналов. Можно выбрать диапазон выводимых журналов. Можно также запросить просмотр списка неактивных резервных копий.

Опция EXTRACT позволяет копировать образы резервных копий или журналы из TSM в текущий каталог. Можно выбирать, какие образы резервных копий или журналы извлекать.

Опция DELETE позволяет деактивировать образы резервных копий или удалять журналы из TSM. Можно выбирать, какие образы резервных копий или журналы обрабатывать. Опцию KEEP *n* можно использовать для сохранения *n* последних резервных копий. С требованием DELETE можно также использовать опции OLDER THAN *отметка-времени* или *n* DAYS.

Интеграция Tivoli Space Manager Integration со связями данных

Менеджер связей данных DB2 может использовать Tivoli Space Manager (TSM) и его виртуальную файловую систему (FSM), располагаемую поверх собственной журнальной файловой системы (JFS). Обращение к FSM и ее конфигурирование производится аналогично JFS.

Эта возможность полезна тем, кто использует файловый системы с большими файлами, которые периодически переносятся на другие носители. Пространство для этих файловых систем должно управляться на регулярной основе. Поддержка TSM в менеджере связей данных DB2 повышает гибкость управления пространством для файлов DATALINK. Вместо предварительного размещения достаточной памяти в файловой системе менеджера связей данных для всех возможных файлов, TSM позволяет размещать управляемую менеджером связей данных файловую систему, периодически корректируемую без риска ее переполнения при обычном использовании.

Ограничения

- В настоящее время эта возможность поддерживается только в AIX.
- Избирательное перемещение (**dsmmigrate**) и вызов связанных файлов FC (базы данных разрешений на чтение) может выполняться только пользователем root.

  Избирательное перемещение может выполнять только владелец файла, то есть для файлов базы данных разрешений на чтение - администратор менеджера связей данных (dlfm). Для обращения к этим файлам необходим маркер со стороны базы данных хоста. Единственный пользователь, которому такой маркер не требуется - это пользователь “root”. Избирательное перемещение и восстановление таких файлов проще выполнять пользователю “root”. Пользователь dlfm может выполнить только первое перемещение файла FC при помощи действитеьного маркера. При повторной попытке перемещения (после восстановления) операция завершится неудачно с сообщением об ошибке “ANS1028S Internal program error. Please see your service representative.” Если кто-либо, кроме пользователя root, попытается применить **dsmmigrate** к файлу FC, операция завершится неудачно. Это ограничение не очень существенно, поскольку к файлам на файл-сервере обычно обращаются только администраторы.
- Системные вызовы **stat** и **statfs** показывают тип *Vfs* как `fsm`, а не как `dlfs`, даже если dlfs устанавливается поверх fsm.

  Такое поведение обеспечивает нормальную работу демонов **dsmrecalld**, которые выполняют вызовы **statfs** для файловой системы, проверяя, является ли тип Vfs fsm или нет.
- Команда **dsmls** не выводит ничего, если файл с минимальным номером *inode* связан с FC (базой данных разрешений на чтение).

  Команда **dsmls** подобна команде **ls**: она выводит список файлов, управляемых TSM. Никаких действий пользователя не требуется.

Приложение H. Обработчик пользователя для восстановления баз данных

Для автоматического архивирования и восстановления файлов журнала можно написать *программу обработчика пользователя*. (В OS/2 программы обработчика пользователя можно также использовать для операций резервного копирования и восстановления из резервной копии.) Перед вызовом программы обработчика пользователя для архивирования или получения файлов журнала убедитесь, что параметр конфигурации базы данных *userexit* имеет значение YES. Этот же параметр разрешает для базы данных восстановление с повтором транзакций.

Когда вызывается программа обработчика пользователя, менеджер баз данных передает управление выполняемому файлу db2uext2. (В OS/2 при операциях резервного копирования и восстановления сначала вызывается файл db2uexit.cmd, который, в свою очередь, вызывает файл db2uext2.) Менеджер баз данных передает параметры программе db2uext2, а она по завершении передает менеджеру баз данных код возврата. Поскольку менеджер баз данных обрабатывает только ограниченный набор возвращаемых состояний, состояния ошибок должна обрабатывать программа обработчика пользователя (смотрите раздел “Обработка ошибок” на стр. 477). Кроме того, поскольку экземпляр менеджера баз данных может вызывать только одну программу обработчика пользователя, в ней должны быть разделы для всех операций, которые от нее,возможно, потребуется выполнять.

В этом приложении рассматриваются следующие темы:

Примеры программ обработчика пользователя

Примеры программ обработчика пользователя есть для всех поддерживаемых платформ. Вы можете изменять эти программы в соответствии со своими потребностями. Примеры программ хорошо откомментированы, что поможет вам использовать их наиболее эффективно.

Учтите, что программы обработчика пользователя должны *копировать* файлы журнала из активного каталога журнала в архивный каталог журнала. Не удаляйте файлы журнала из активного каталога журнала. (Это может привести

© Copyright IBM Corp. 2001

к проблемам при восстановлении базы данных.) DB2 удаляет архивированные файлы журнала из активного каталога журнала, когда они больше не нужны для восстановления базы данных,

Ниже приводится описание примеров программ обработчика пользователя, поставляемых вместе с DB2.

- **Системы на основе UNIX**

  Примеры программ обработчика пользователя для DB2 для систем на основе UNIX находятся в подкаталоге `sqllib/samples/c`. Хотя большинство поставляемых примеров написаны на языке C, ваша программа обработчика пользователя может быть написана на другом языке программирования.

  Ваша программа обработчика пользователя должна быть исполняемым файлом с именем `db2uext2`.

  Для систем на основе UNIX есть четыре примера программы обработчика пользователя:

  – `db2uext2.cadsm`

    В этом примере для архивирования и восстановления файлов журнала базы данных используется Tivoli Storage Manager.

  – `db2uext2.ctape`

    В этой программе для архивирования и восстановления файлов журнала базы данных используется ленточное устройство.

  – `db2uext2.cdisk`

    В этой программе для архивирования и восстановления файлов журнала базы данных применяются команда COPY операционной системы и диски.

  – `db2uxt2.cxbsa`

    В этой программе используется утилита Legato NetWorker** Версии 4.2.5 фирмы Legato** Systems, Inc. Ее можно использовать для архивирования и восстановления файлов журнала базы данных. Этот пример поддерживается только в системе AIX.

- **Операционные системы Windows**

  Примеры программ обработчика пользователя для DB2 для операционных систем Windows находятся в подкаталоге `sqllib\samples\c`. Хотя большинство поставляемых примеров написаны на языке C, ваша программа обработчика пользователя может быть написана на другом языке программирования.

  Ваша программа обработчика пользователя должна быть исполняемым файлом с именем `db2uext2`.

  Для операционных систем Windows есть два примера программы обработчика пользователя :

  – `db2uext2.cadsm`

В этом примере для архивирования и восстановления файлов журнала базы данных используется Tivoli Storage Manager.

– `db2uext2.cdisk`

  В этой программе для архивирования и восстановления файлов журнала базы данных применяются команда COPY операционной системы и диски.

- **OS/2**

  Примеры программ обработчика пользователя для DB2 для OS/2 находятся в подкаталоге экземпляра в каталоге `\sqllib\samples\rexx`. (Исключение - программа `dbuexit.CAD`: она находится в подкаталоге экземпляра в каталоге `\sqllib\samples\c`.) Несмотря на то, что большинство поставляемых примеров - это командные файлы REXX, ваша программа обработчика пользователя может быть написана на другом языке программирования.

  Пример, выбранный для выполнения, следует переименовать в `db2uexit` с расширением `.cmd` или `.exe`. Переместите переименованный файл в каталог `\sqllib\bin`.

  Для OS/2 есть пять примеров программ обработчика пользователя :

  – `db2uexit.ex1`

    В этом примере использована программа Sytos Premium** Версии 2.2 фирмы Seagate** Software Inc. Ее можно применять для записи данных на внешнее ленточное устройство IBM и чтения с этого устройства. В настоящее время поддерживается только Версия 2.2 программы Sytos Premium. (Чтобы использовать этот продукт, вам потребуется FixPak 26 для OS/2.) Просмотрите текст программы примера, чтобы узнать прочие требования.

  – `db2uexit.ex2`

    В этом примере используется программа Filesafe** фирмы Mountain** Corporation. Ее можно применять для записи данных на ленточное устройство Mountain и чтения с этого устройства. Каждой резервной копии базы данных назначается уникальная метка тома, поэтому на одной ленте можно хранить несколько резервных копий одной или нескольких баз данных.

  – `db2uexit.ex3`

    В этом примере используется программа MaynStream** фирмы Maynard** Corporation. Ее можно применять для записи данных на ленточное устройство Maynard и чтения с этого устройства. Программа MaynStream не может восстанавливать базу данных на диск, отличный от того, с которого была получена резервная копия.

  – `db2uexit.ex4`

    В этом примере использована команда OS/2 XCOPY. Устройством хранения может быть любое устройство, поддерживаемое OS/2, например

жесткий диск, дискета или оптический диск. Если сконфигурировать рабочую станцию соответствующим образом, этим устройством может быть переназначенный сетевой диск.

Для резервного копирования и восстановления баз данных команду XCOPY использовать нельзя.

– `db2uexit.CAD`

Этот пример, написанный на C, эквивалентен примеру программы, использующему Tivoli Storage Manager (TSM). Его можно использовать для архивирования и восстановления файлов журнала базы данных.

Формат вызова

При вызове программы обработчика пользователя менеджер баз данных передает ей набор параметров (с типом данных CHAR). Формат вызова зависит от операционной системы.

- Формат вызова для операционных систем на основе UNIX или Windows NT/2000:

```
db2uext2 -OS<система> -RL<выпуск> -RQ<требование> -DB<имя_базы_данных>
-NN<номер_узла> -LP<путь_журнала> -LN<имя_журнала> -AP<пароль_tsm>
-SP<начальная_страница> -LS<размер_журнала>
```

система Задает платформу, на которой выполняется экземпляр. Допустимые значения: `AIX`, `Solaris`, `HP-UX`, `SCO`, `Linux`, `Dynix/ptx`, `SGI` и `NT`.

выпуск Задает уровень выпуска DB2. Например, `SQL07020`.

требование Задает тип требования. Допустимые значения: `ARCHIVE` и `RETRIEVE`.

имя_базы_данных
Задает имя базы данных.

номер_узла Задает номер локального узла, например, `5`.

путь_журнала Задает полное имя каталога файлов журнала. Путь должен завершаться конечным разделителем. Например, `/u/database/log/path/` или `d:\logpath\`.

имя_журнала Задает имя файла журнала для архивирования или восстановления, например `S0000123.LOG`.

пароль_tsm Задает пароль TSM. (Если ранее было задано значение параметра конфигурации базы данных *tsm_password*, это значение передается программе обработчика пользователя.)

начальная_страница
Задает смещение в страницах по 4 Кбайта, с которого на данном устройстве начинается экстент журнала.

размер_журнала

Задает объем экстента журнала в страницах по 4 Кбайта. Этот параметр используется, только если для записи журнала применяется непосредственное устройство.

- Формат вызова для OS/2:

```
действие диск алиас_базы_данных путь_журнала файл_журнала индикатор
```

действие Допустимые значения: `BACKUP`, `RESTORE`, `ARCHIVE` и `RETRIEVE`.

диск Для операции резервного копирования задает диск, на котором находится база данных, выбранная для резервного копирования. Для операции восстановления задает диск, на который надо восстановить базу данных. Для операции архивирования или восстановления задает диск, на котором находится база данных. В каждом случае нужно указать букву дисковода с двоеточием (например, `C:`).

алиас_базы_данных

Задает алиас базы данных (или имя базы данных, если алиас не существует).

путь_журнала Для операции резервного копирования или восстановления задает полное имя файла ответов, содержащего список файлов, предназначенных для резервного копирования или восстановления. Каждое имя в этом списке - полное имя, допускается использовать символы подстановки. Для операций восстановления буква диска и путь обозначают диск и путь к исходному файлу базы данных, с которого была снята резервная копия. Например, если в файле ответов содержится запись `C:\SQLUTIL\dbname.MH1`, это означает, что резервная копия файла `dbname.MH1` создавалась из `C:\SQLUTIL`.

Для операции архивирования или восстановления задает полное имя каталога журнала. Например, `C:\SQL00001\SQLOGDIR\`.

файл_журнала Для операции резервного копирования задает метку носителя, созданную утилитой резервного копирования. Эта метка составляется из алиаса базы данных и отметки времени. Для операции восстановления нужно указать путь к подкаталогу базы данных, в который восстанавливаются файлы. Буква диска не включается, поскольку она задается в параметре *диск*. Используется формат `\SQL`*`nnnnn`*`\`.

Для операции архивирования или восстановления задает имя файла журнала. Например, `S0000001.LOG`.

индикатор Задает индикатор, используемый для поддержки нескольких вызовов операции резервного копирования или

восстановления. В качестве значения для первого вызова используется символ 1, а для последующих вызовов - символ 2.

Во время операции резервного копирования или восстановления программа обработчика пользователя вызывается несколько раз. При первом вызове проводится резервное копирование или восстановление файлов заголовка носителя (`.MH`*n*), а при втором вызове - резервное копирование или восстановление всего набора файлов базы данных.

Этот параметр не используется операциями архивирования и восстановления.

Особенности резервного копирования и восстановления (только для DB2 for OS/2)

При написании программы обработчика пользователя, которая вызывается из утилиты резервного копирования или восстановления, следует учитывать следующие особенности:

- Ненулевой код возврата, возвращаемый программой обработчика пользователя, приводит к неудачному завершению данной утилиты, и она не пытается повторить обработку.
- В полных именах файлов используйте только поддерживаемые символы подстановки. Например, в качестве критериев поиска можно использовать как `C:\SQL00001\*.*`, так и `C:\*.MH*`.
- Программа обработчика пользователя должна обрабатывать формат файла ответов: одно полное имя файла в одной строке с завершением каждой строки символами возврата каретки и перевода строки. Символ конца файла отсутствует.
- Если на одном носителе размещено несколько резервных копий одной и той же базы данных, программа обработчика пользователя должна уметь выбрать нужную копию для восстановления. (Смотрите описание `db2uexit.ex2` в разделе “Примеры программ обработчика пользователя” на стр. 471.)
- Две одновременно запущенных операции резервного копирования с совместным использованием одного устройства резервного копирования должны выполняться последовательно.
- Если резервная копия занимает несколько носителей, запрос для носителей такой резервной копии должен обрабатываться программой обработчика пользователя, либо программой, которую она вызывает. Для поддержки этой функции утилиты резервного копирования и восстановления открывают для вызова программы обработчика пользователя приоритетный сеанс операционной системы.

- Программа обработчика пользователя при резервном копировании не должна копировать никакие подкаталоги в каталоге базы данных.
- При восстановлении базы данных с использованием программы обработчика пользователя утилите восстановления требуется полное управление этой базой данных. Однако на рабочей станции помимо восстанавливаемой базы данных могут быть активные соединения и с другими базами данных.
- Если при резервном копировании или восстановлении базы данных с использованием программы обработчика пользователя другая операция воспользуется этим же ленточным устройством, резервное копирование или восстановление может завершиться неудачно, и тогда потребуется запустить процесс сначала. Чтобы избежать подобной ситуации, следует убедиться, что пока выполняется операция резервного копирования или восстановления, другие базы данных, вызывающие программу обработчика пользователя для работы с журналом, не используются, или что программа обработчика пользователя повторит попытку резервного копирования или восстановления в другое время, если устройство не готово.
- Во время операции восстановления буква дисковода и путь могут отличаться от тех значений, что были заданы во время операции резервного копирования. Например, если резервная копия для файла `dbname.MH1` создается из каталога `C:\SQLUTIL`, его можно восстановить в каталог `D:\SQLUTIL2`.

Обработка ошибок

Программа обработчика пользователя должна возвращать определенные коды возврата, чтобы менеджер баз данных мог правильно их интерпретировать. Так как программа обработчика пользователя вызывается командным процессором операционной системы, существует возможность возврата кодов ошибок от самой операционной системы. Поскольку эти коды ошибок никак не переназначаются, чтобы получить о них сведения, воспользуйтесь утилитой справки операционной системы.

В OS/2 любой ненулевой код возврата, возвращаемый программой обработчика пользователя, приводит к неудачному завершению утилиты резервного копирования или восстановления, и она не пытается повторять обработку. Эти утилиты выдают общий SQLCODE -2029, а в их текстовом сообщении отображается код, возвращенный программой обработчика пользователя или операционной системой.

В Табл. 24 на стр. 478 приведены коды, которые может возвращать программа обработчика пользователя, и описано, как эти коды интерпретируются менеджером баз данных. Если кода возврата нет в этой таблице, он обрабатывается, как код со значением 32.

Таблица 24. Коды возврата программы обработчика пользователя. Применяется только для операций архивирования и восстановления.

Код возврата	Объяснение
0	Успешное завершение.
4	Обнаружена ошибка временного ресурса.[a]
8	Требуется вмешательство оператора.[a]
12	Аппаратная ошибка.[b]
16	Ошибка программы обработчика пользователя или программной функции, используемой этой программой.[b]
20	Ошибка в одном или нескольких параметрах, переданных программе обработчика пользователя. Проверьте правильность обработки программой обработчика пользователя заданных параметров.[b]
24	Программа обработчика пользователя не найдена. Для OS/2 это сообщение об ошибке может также означать, что на текущем носителе резервных копий не удалось найти файл, необходимый для выполнения операции восстановления.[b]
28	Ошибка ввода/вывода (I/O) или операционной системы.[b]
32	Программа обработчика пользователя была прервана пользователем.[b]
255	Ошибка, вызванная тем, что программа обработчика пользователя не смогла загрузить библиотечный файл для исполняемого файла.[c]

Таблица 24. Коды возврата программы обработчика пользователя (продолжение). Применяется только для операций архивирования и восстановления.

Код возврата	Объяснение
[a] Для операции архивирования или восстановления код возврата 4 или 8 приводит к повтору через пять минут. Если программа обработчика пользователя продолжает возвращать 4 или 8 на обращения операции восстановления к одному и тому же файлу журнала, DB2 зависает. (Это относится к операциям с повтором транзакций и вызовам API **sqlurlog**, используемого утилитой копирования.) [б] Требования обработчика пользователя приостанавливаются на пять минут. В течение этого времени все требования игнорируются, включая требование, вызвавшее код ошибки. После пятиминутной приостановки обрабатывается следующее требование. Если это требование обрабатывается без ошибки, продолжается обработка последующих требований обработчика пользователя, и DB2 повторно вызывает требование операции архивирования, которое привело к неудачному завершению или было приостановлено ранее. Если во время повтора генерируется код возврата больше 8, требования приостанавливаются еще на пять минут. Пятиминутные приостановки продолжаются, пока не будет исправлена ошибка или не будет остановлена и перезапущена база данных. Когда все программы отсоединятся от базы данных, DB2 вызывает требование архивирования для всех файлов журнала, которые, возможно, не были успешно архивированы прежде. Если программа обработчика пользователя не смогла выполнить архивирование файлов журналов, диск может быть заполнен файлами журналов, и производительность может упасть. Когда диск будет переполнен, менеджер баз данных не будет более принимать запросы прикладных программ на изменение баз данных. Если программа обработчика пользователя вызывалась для получения файлов журнала, восстановление с повтором транзакций будет отложено, но не остановлено, если только не была задана опция ROLLFORWARD STOP. Если опция STOP не задана, вы можете исправить ошибку и продолжить восстановление. [в] Если программа обработчика пользователя возвращает код ошибки 255, скорее всего, программа не смогла загрузить библиотечный файл для исполняемого файла. Чтобы проверить это, вызовите программу обработчика пользователя вручную. Будет выведена более подробная информация. **Примечание:** В операциях архивирования и восстановления для всех кодов возврата, кроме 0, 4 и 24, выдается оповещение. Оповещение содержит код возврата программы обработчика пользователя и копию входных параметров, переданных в программу обработчика пользователя.	

Приложение I. API резервного копирования и восстановления для продуктов других поставщиков

DB2 обеспечивает интерфейсы, при помощи которых продукты других поставщиков, управляющие носителями информации, могут сохранять и получать данные при операциях резервного копирования и восстановления. Это открывает возможность использовать для резервного копирования и восстановления накопители, отличные от дискеты, диска, ленты и Tivoli Storage Manager, входящих в стандартный список DB2.

Продукты других поставщиков, управляющие носителями информации, далее в этом приложении называются продуктами поставщиков.

В DB2 задан набор функций-прототипов, обеспечивающих интерфейс данных общего назначения для резервного копирования и восстановления, который может использоваться многими независимыми производителями. Эти функции должны поставляться производителем в составе совместно используемой библиотеки для систем на основе UNIX или в DLL для операционных систем OS/2 и Windows. При обращении к этим функциям из DB2 происходит загрузка совместно используемой библиотеки или DLL, заданных в программе резервного копирования и восстановления, после чего для выполнения требуемых задач вызываются функции, предоставленные независимым производителем.

Это приложение состоит из четырех частей:

- Обзор взаимодействия DB2 с независимыми продуктами.
- Подробное описание всех API DB2 для независимых продуктов.
- Информация о структурах данных, используемых при вызовах API.
- Подробности о вызове резервного копирования и восстановления при помощи продуктов других поставщиков.

Обзор

Определены пять функций для взаимодействия DB2 с независимыми продуктами:

- sqluvint - Инициализировать устройство и связаться с ним
- sqluvget - Чтение данных с устройства
- sqluvput - Запись данных на устройство
- sqluvend - Отменить связь с устройством
- sqluvdel - Удалить сеанс после принятия

© Copyright IBM Corp. 2001

DB2 будет вызывать эти функциями, и они должны быть реализованы в продукте поставщика в составе совместно используемой библиотеки для систем на основе UNIX или в DLL для операционных систем OS/2 и Windows.

Примечание: Совместно используемая библиотека или программа DLL будут запускаться как часть программы механизма базы данных. Поэтому эта программа должна быть повторно-входной и тщательно отлаженной. Ошибки в функции могут угрожать целостности данных в базе данных.

На последовательность функций, которые будут вызываться DB2 во время конкретной операции резервного копирования или восстановления, влияют:

- Число сеансов.
- Является ли операция резервным копированием или восстановлением.
- Задан ли для операции резервного копирования или восстановления режим PROMPTING.
- Характеристики устройства, на котором сохраняются данные.
- Ошибки, которые могут возникнуть во время операции.

Число сеансов

DB2 поддерживает резервное копирование и восстановление объектов базы данных при помощи одного или нескольких потоков данных (сеансов). Для использования трех сеансов при резервном копировании или восстановлении потребуется доступ к трем физическим или логическим устройствам. При использовании устройств, поддерживаемых продуктом другого поставщика, его функции отвечают за управление интерфейсом к каждому физическому или логическому устройству. DB2 только записывает и считывает данные в буферах обмена с функциями производителя.

Число используемых сеансов задается как параметр в прикладной программе, вызывающей функцию резервного копирования или восстановления базы данных. Это значение содержится в структуре INIT-INPUT, используемой в **sqluvint** (смотрите раздел “sqluvint - Инициализировать устройство и связаться с ним” на стр. 491).

DB2 прекратит инициализировать сеансы, когда будет достигнуто заданное число или при вызове **sqluvint** будет получен код возврата SQLUV_MAX_LINK_GRANT. Чтобы предупреждать DB2 о достижении максимального числа сеансов, которые она может поддерживать, продукту другого поставщика нужна программа, отслеживающая число активных сеансов. Без такого предупреждения DB2 может инициализировать запрос на сеанс, который не будет выполнен, из-за чего все сеансы будут прерваны, и вся операция резервного копирования или восстановления завершится с ошибкой.

Если операция - резервное копирование, DB2 в начале каждого сеанса записывает заголовок носителя. Эта запись содержит информацию, которую DB2 использует для идентификации сеанса при операции восстановления. DB2 создает уникальные идентификаторы сеансов, добавляя к имени резервной копии порядковый номер. Первый сеанс получает номер один, и номер увеличивается на единицу с каждым сеансом, инициируемым вызовом **sqluvint** при операции резервного копирования или восстановления. Дополнительные подробности смотрите в разделе “INIT-INPUT” на стр. 511.

Когда операция резервного копирования завершается успешно, DB2 записывает в последний закрываемый сеанс концевой маркер носителя. Этот концевой маркер включает информацию о том, сколько сеансов использовала DB2 для выполнения операции резервного копирования. Во время операции восстановления эта информация используется для проверки того, что восстановлены все сеансы, или потоки данных.

Операция без ошибок, предупреждений или подсказок

При резервном копировании DB2 выполняет для *каждого* сеанса следующую последовательность вызовов.

```
sqluvint, action = SQLUV_WRITE
```

далее следует от 1 до n вызовов

```
sqluvput
```

далее следует 1 вызов

```
sqluvend, action = SQLUV_COMMIT
```

Когда DB2 выполняет вызов **sqluvend** (action = SQLUV_COMMIT), она ожидает, что продукт другого поставщика должным образом сохранит выведенные данные. Код возврата SQLUV_OK сообщает DB2 об успешном завершении.

Структура DB2-INFO, используемая при вызове **sqluvint**, содержит необходимую информацию для идентификации резервной копии (смотрите раздел “DB2-INFO” на стр. 506). В частности, передается порядковый номер. Продукт другого поставщика имеет возможность, если нужно, сохранить эту информацию. DB2 использует ее при восстановлении для идентификации резервной копии, выбранной для восстановления.

При восстановлении для каждого сеанса выполняется такая последовательность вызовов:

```
sqluvint, action = SQLUV_READ
```

далее следует от 1 до n вызовов

```
sqluvget
```

далее следует 1 вызов

```
sqluvend, action = SQLUV_COMMIT
```

Структура DB2-INFO, используемая при вызове **sqluvint**, будет содержать необходимую информацию для идентификации резервной копии. Порядковый номер не передается. DB2 ожидает, что будут возвращены все объекты резервного копирования (выведенные и принятые при резервном копировании данные сеансов). Первым возвращаемым объектом резервного копирования является объект, сгенерированный с порядковым номером 1; остальные объекты восстанавливаются в произвольном порядке. DB2 проверяет концевой маркер носителя, чтобы убедиться, что были обработаны все объекты.

Примечание: Не все продукты других поставщиков записывают имена объектов резервного копирования. Это особенно вероятно при использовании для резервного копирования лент и иных носителей ограниченной емкости. Во время инициализации сеансов восстановления эта информация об идентификации поможет вовремя находить нужные объекты резервного копирования, особенно когда для хранения резервных копий используются CD-ченджеры или другие автоматизированные системы. DB2 обязательно проверяет заголовок носителя (первую сделанную сеансом запись), чтобы убедиться, что восстанавливаются правильные данные.

Режим PROMPTING

После инициализации операции резервного копирования или восстановления возможны два режима работы:

- WITHOUT PROMPTING или NOINTERRUPT, при котором продукт другого поставщика не имеет возможности записывать сообщения для пользователя и получать его ответы.
- PROMPTING или INTERRUPT, при котором пользователь может получать сообщения от этого продукта и отвечать на них.

В режиме PROMPTING для резервного копирования и восстановления определены три возможных ответа пользователя:

- Continue (Продолжить)

  Операция чтения или записи данных на устройстве будет возобновлена.
- Device terminate (Прервать на устройстве)

  Передача данных устройству прекращается и сеанс прерывается.
- Terminate (Прервать)

  Прерывается вся операция резервного копирования или восстановления.

Использование режимов PROMPTING и WITHOUT PROMPTING обсуждается в последующих разделах.

Характеристики устройства

Для API поддержки продуктов других поставщиков определены два общих типа устройств:

- Устройства ограниченной емкости, которые требуют вмешательства пользователя для смены носителя; например, ленточные накопители, дисководы для дискет и компакт-дисков.
- Устройства очень большой емкости, при нормальной работе которых пользователю не приходится работать с носителями; например, CD-ченджеры и устройства с автоматизированной сменой носителей.

Устройство ограниченной емкости может потребовать, чтобы пользователь во время операции резервного копирования или восстановления получал приглашения загрузить дополнительный носитель. Обычно DB2 допускает изменение порядка загрузки носителей при операциях резервного копирования и восстановления. Кроме того, она обеспечивает возможность передавать пользователю сообщения продукта другого поставщика о работе с носителями. Для этого нужно, чтобы операция резервного копирования или восстановления была инициализирована в режиме PROMPTING. Текст сообщения о работе с носителями задается в поле описания в структуре кодов возврата.

В режиме PROMPTING, когда DB2 получает код возврата SQLUV_ENDOFMEDIA или SQLUV_ENDOFMEDIA_NO_DATA от вызова **sqluvput** (при записи) или **sqluvget** (при чтении), она:

- Помечает последний буфер, посланный сеансу, как подлежащий повторной посылке, если вызванная функция - **sqluvput**. Он будет передан сеансу позже.
- Вызывает сеанс при помощи **sqluvend** (action = SQLUV_COMMIT). При успешном вызове (код возврата SQLUV_OK) DB2:
  - Посылает пользователю взятое из структуры кодов возврата сообщение продукта другого поставщика о конце носителя.
  - Предлагает пользователю ответы: continue (продолжить), device terminate (прервать на устройстве) или terminate (прервать).
- При ответе *continue* (продолжить) DB2 инициализирует еще один сеанс при помощи вызова **sqluvint** и, если сеанс успешно открыт, начинает запись или чтение данных в этом сеансе. Для уникальности идентификации сеансов записи DB2 увеличивает последовательный номер. Этот последовательный номер доступен в структуре DB2-INFO, используемой с функцией **sqluvint**, и помещается в заголовок носителя, то есть первую посылаемую сеансу запись данных.

  DB2 не запустит больше сеансов, чем будет затребовано при запуске операции резервного копирования или восстановления и чем задано продуктом другого поставщика в предупреждении SQLUV_MAX_LINK_GRANT вызова **sqluvint**.
- При ответе *device terminate* (прервать на устройстве) DB2 не пытается инициализировать еще один сеанс, и число активных сеансов уменьшается на

один. DB2 не допускает прерывания всех сеансов при помощи ответов о прерывании на устройстве; хотя бы один сеанс должен оставаться активным, пока не завершится операция резервного копирования или восстановления.

- При ответе *terminate* (прервать) DB2 прерывает операцию резервного копирования или восстановления. Дополнительную информацию о том, как DB2 прерывает сеансы, смотрите в разделе “Если DB2 возвращаются состояния ошибки” на стр. 487.

Поскольку производительность резервного копирования и восстановления часто зависит от числа используемых устройств, поддержание параллелизма имеет большое значение. При операциях резервного копирования пользователям рекомендуется отвечать `continue` (продолжить), если только не известно, что остающиеся активные сеансы вместят данные, которые еще остается записать. При операциях восстановления пользователям также рекомендуется отвечать `continue` (продолжить), пока не будут обработаны все носители.

Если резервное копирование или восстановление выполняется в режиме WITHOUT PROMPTING и DB2 получает от сеанса код возврата SQLUV_ENDOFMEDIA или SQLUV_ENDOFMEDIA_NO_DATA, она прервет сеанс и не будет пытаться открыть еще один сеанс. Если все сеансы сообщат DB2 о конце носителя до завершения операции резервного копирования или восстановления, операция завершается с ошибкой. Из-за этого следует проявлять осторожность при использовании режима WITHOUT PROMPTING для устройств ограниченной емкости; с другой стороны, при работе с устройствами очень большой емкости имеет смысл использовать этот режим.

Продукт другого поставщика может скрывать от DB2 действия по монтированию и переключению носителей, создавая впечатление устройства бесконечной емкости. В таком режиме работают некоторые устройства очень большой емкости. В таких случаях важно, чтобы во время операции восстановления все данные резервных копий возвращались DB2 в том же порядке. Невыполнение этого условия может привести к потере данных, причем у DB2 не будет никаких средств обнаружить такую потерю, и она будет считать восстановление успешным.

DB2 записывает данные в независимый продукт в предположении, что каждый буфер будет храниться на одном и только на одном носителе (например, на ленте). Продукт другого поставщика может распределять буферы DB2 по нескольким носителям, не ставя в известность DB2. В этом случае порядок обработки носителей при операции восстановления будет важен, поскольку независимый продукт будет нести ответственность за возврат DB2 буферов, воссозданных с нескольких носителей. Невыполнение этого условия приведет к невыполнению операции восстановления.

Если DB2 возвращаются состояния ошибки

При выполнении операции резервного копирования или восстановления DB2 ожидает, что все сеансы завершатся успешно; в противном случае вся операция резервного копирования или восстановления завершается с ошибкой. Сеанс сообщает DB2 об успешном завершении кодом возврата SQLUV_OK в вызове **sqluvend**, action = SQLUV_COMMIT.

Если возникают неустранимые ошибки, DB2 прерывает сеанс. Это могут быть ошибки DB2 и ошибки, возвращенные DB2 продуктом другого поставщика. Поскольку для завершения операции резервного копирования или восстановления должны успешно завершиться все сеансы, ошибка в одном из них заставляет DB2 прервать все остальные сеансы, связанные с данной операцией.

Когда независимый продукт отвечает на вызов из DB2 кодом возврата неустранимой ошибки, он также может передать дополнительную информацию в виде текста сообщения в поле описания структуры RETURN-CODE. Этот текст сообщения вместе с информацией DB2 предъявляется пользователю для исправления ситуации.

Возможен сценарий резервного копирования, когда один сеанс завершается успешно, а в другом сеансе, связанном с той же операцией резервного копирования, возникает неустранимая ошибка. Поскольку для того, чтобы операция резервного копирования считалась успешной, требовалось успешное завершение всех сеансов, DB2 вынуждена удалить данные, выведенные в принятые сеансы: DB2 выполняет вызов **sqluvdel**, чтобы потребовать удалить объект. Этот вызов не считается сеансом ввода-вывода; он отвечает за инициализацию и прерывание всех соединений, необходимых для удаления объекта резервного копирования.

Последовательного номера в структуре DB2-INFO не будет; **sqluvdel** удалит все объекты резервного копирования, которые отвечают остальным параметрам в структуре DB2-INFO.

Состояния предупреждения

DB2 может получать предупреждения от продукта другого поставщика, например, когда устройство не готово или при возникновении других исправимых ситуаций. Это верно как для операций чтения, так и для операций записи.

При вызовах **sqluvput** и **sqluvget** независимый продукт может задать код возврата SQLUV_WARNING и, если нужно, передать любую дополнительную информацию в виде текста сообщения в поле описания структуры RETURN-CODE. Этот текст сообщения предъявляется пользователю для

исправления ситуации. Пользователь может выбрать один из трех ответов: continue (продолжить), device terminate (прервать на устройстве) или terminate (прервать):

- При ответе *continue* (продолжить) во время операции резервного копирования DB2 пытается повторно записать буфер при помощи **sqluvput**. При операции восстановления DB2 выполняет вызов **sqluvget**, чтобы прочитать следующий буфер.
- При ответах *device terminate* (прервать на устройстве) и или *terminate* (прервать) DB2 прерывает всю операцию резервного копирования или восстановления, так же, как после возникновения неустранимой ошибки (в частности, прерывает активные сеансы и удаляет принятые сеансы).

Советы и рекомендации

В этом разделе даются советы и рекомендации по созданию продуктов других поставщиков.

Файл хронологии восстановлений

Файл хронологии восстановлений может использоваться при операциях восстановления базы данных как вспомогательное средство. Он связан с каждой базой данных и автоматически обновляется при каждой операции резервного копирования и восстановления. Дополнительную информацию о файле хронологии восстановлений смотрите в разделе “Что такое файл хронологии восстановления” на стр. 53 . Информацию в файле можно просматривать, изменять и сокращать, для чего можно использовать:

- Центр управления
- командную строку (CLP)
  - команду LIST HISTORY
  - команду UPDATE HISTORY FILE
  - команду PRUNE HISTORY
- API
  - db2HistoryOpenScan
  - db2HistoryGetEntry
  - db2HistoryCloseScan
  - db2HistoryUpdate
  - db2Prune

Информацию о структуре этого файла смотрите в разделе “Структура данных: db2HistData” на стр. 381.

По завершении операции резервного копирования в этот файл заносится одна или несколько записей. Если данные резервного копирования направлялись в

устройства под управлением продукта другого поставщика, в записи хронологии в поле DEVICE помещается 0, а в поле LOCATION - одно из двух:

- Имя файла независимого продукта, заданное при вызове операции резервного копирования.
- Имя совместно используемой библиотеки, если имя файла независимого продукта не задано.

Дополнительную информацию о задании этой опции смотрите в разделе “Вызов операции резервного копирования или восстановления при помощи продуктов других поставщиков” на стр. 517.

Поле LOCATION можно обновлять при помощи Центра управления, командной строки и API. Информация о положении резервной копии может изменяться, если для хранения используются устройства ограниченной емкости (например, сменные носители), и носители физически перемещаются в другое (возможно, удаленное) место хранения. В таких ситуациях файл хронологии восстановлений можно использовать для поиска резервной копии при необходимости восстановления.

Функции и структуры данных

В следующих разделах описаны функции общего характера и структуры данных, доступные для продуктов других поставщиков.

API для продуктов других поставщиков:

Структуры данных, используемые API для продуктов других поставщиков:

“DATA” на стр. 515
Содержит данные, которые были переданы между DB2 и устройством под управлением продукта другого поставщика.

“RETURN-CODE” на стр. 516
Содержит код возврата и описание ошибки.

sqluvint - Инициализировать устройство и связаться с ним

Вызов этой функции предоставляет информацию для инициализации и установки логической связи между DB2 и устройством поставщика.

Полномочия

Одни из следующих:

- *sysadm*
- *dbadm*

Необходимое соединение

База данных

Файл включения API

sql.h

Синтаксис API на языке C

```
/* Файл: sqluvend.h */
/* API: Инициализировать устройство и связаться с ним */
/* ... */
int sqluvint (
  struct Init_input   *,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

Параметры API

Init_input
Входной. Структура, содержащая информацию, предоставляемую DB2 для установки логической связи с устройством поставщика.

Init_output
Выходной. Структура, содержащая вывод, возвращенный устройством поставщика.

Return_code
Выходной. Структура, содержащая код возврата для передачи DB2 и краткое текстовое объяснение.

Замечания по использованию

При каждом сеансе ввода-вывода DB2 будет вызывать эту функцию для получения хэндла устройства. Если по какой-либо причине при инициализации эта функция обнаружит ошибку, она сообщит о ней через код возврата. Если код возврата указывает на ошибку, DB2 может принять решение о завершении операции путем вызова функции **sqluvend**. Подробности о возможных кодах возврата, а также реакция DB2 на каждый из них приводятся в таблице кодов возврата (смотрите Табл. 25 на стр. 493).

В структуре INIT-INPUT содержатся элементы, которые могут быть использованы установленным продуктом для определения возможности выполнения резервного копирования или восстановления из резервной копии:

- size_HI_order и size_LOW_order

  Это оцениваемый размер резервной копии. Их можно использовать для определения того, могут ли устройства поставщика обработать образ резервной копии данного размера. Можно использовать их также для оценки количества сменных носителей, необходимых для размещения резервной копии. Если ожидаются проблемы с носителями, имеет смысл прекратить работу на первом вызове **sqluvint**.

- req_sessions

  Для определения возможности операции резервного копирования или восстановления из резервной копии можно использовать число запрошенных пользователем сеансов в сочетании с оцениваемым размером и уровнем подсказок.

- prompt_lvl

  Уровень подсказок сообщает функции устройства другого поставщика, можно ли запрашивать такие действия, как замена сменного носителя (например, установка новой ленты в ленточное устройство). Если к пользователю нельзя обратиться, может быть принято решение о невозможности выполнения операции.

  Если уровень подсказок - WITHOUT PROMPTING, а количество сменных носителей больше запрошенного числа сеансов, DB2 не сможет успешно завершить операцию (дополнительную информацию смотрите в разделах “Режим PROMPTING” на стр. 484 и “Характеристики устройства” на стр. 485).

DB2 задает имя записываемой или читаемой резервной копии посредством полей в структуре DB2-INFO. Если действие - SQLUV_READ, устройство поставщика должно проверить существование объекта с таким именем. Если его не удается найти, для кода возврата должно быть установлено SQLUV_OBJ_NOT_FOUND, чтобы DB2 предприняла соответствующее действие.

После успешного завершения инициализации DB2 может продолжить работу, вызвав другие функции передачи данных, или же завершить сеанс в любое время посредством вызова **sqluvend**.

Коды возврата

Таблица 25. Допустимые коды возврата для sqluvint и ответные действия DB2

Литерал в файле заголовка	Описание	Вероятный следующий вызов	Прочие комментарии
SQLUV_OK	Операция успешно завершена.	sqluvput, sqluvget (смотрите комментарии)	Если действие = SQLUV_WRITE, далее будет вызвана sqluvput (для резервного копирования данных). Если действие = SQLUV_READ, перед тем, как возвращать SQLUV_OK проверьте существование названного объекта; далее будет вызвана sqluvget для восстановления данных.
SQLUV_LINK_EXIST	Сеанс уже был активирован.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.
SQLUV_COMM_ ERROR	Ошибка связи с устройством.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.
SQLUV_INV_VERSION	DB2 и установленные продукты поставщика несовместимы.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.
SQLUV_INV_ACTION	Запрошено недопустимое действие. Может также использоваться для указания того, что данное сочетание параметров приведет к невозможной операции.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.
SQLUV_NO_DEV_ AVAIL	В настоящее время нет доступных для использования устройств.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.
SQLUV_OBJ_NOT_ FOUND	Указанный объект не найден. Этот код необходимо использовать для действия 'R' (чтение) при вызове sqluvint, когда требуемый объект нельзя найти на основе критериев, указанных в структуре DB2-INFO.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.
SQLUV_OBJS_FOUND	Указанным критериям соответствует несколько объектов. Это сообщение появляется для действия при вызове sqluvint 'R' (чтение), когда критериям в структуре DB2-INFO удовлетворяют несколько объектов.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.
SQLUV_INV_USERID	Указан неверный ID пользователя.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.

Таблица 25. Допустимые коды возврата для sqluvint и ответные действия DB2 (продолжение)

Литерал в файле заголовка	Описание	Вероятный следующий вызов	Прочие комментарии
SQLUV_INV_ PASSWORD	Введен неверный пароль.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.
SQLUV_INV_OPTIONS	В поле опций оборудования поставщика обнаружены недопустимые опции.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.
SQLUV_INIT_FAILED	Инициализация не удалась, и сеанс будет закрыт.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.
SQLUV_DEV_ERROR	Ошибка устройства.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.
SQLUV_MAX_LINK_ GRANT	Установлено максимальное число связей.	sqluvput, sqluvget (смотрите комментарии)	Рассматривается DB2 как предупреждение. Это сообщение заставляет DB2 прекращать открывать дополнительные сеансы с устройством поставщика, поскольку достигнуто максимальное число сеансов, которое он поддерживает (примечание: это может быть следствием доступности устройства). Если действие - SQLUV_WRITE (BACKUP), следующим вызовом будет sqluvput. Если действие - SQLUV_READ, перед тем, как возвращать SQLUV_MAX_LINK_GRANT, проверьте существование названного объекта; следующим вызовом будет sqluvget для восстановления данных.
SQLUV_IO_ERROR	Ошибка ввода-вывода.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.
SQLUV_NOT_ ENOUGH_SPACE	Недостаточно места для сохранения всего образа резервной копии; оценка размера приводится как 64-битное значение в байтах.	последующих вызовов нет	Не удалось инициализировать сеанс. Освободите память, выделенную для этого сеанса, и завершите его. Вызов sqluvend не будет получен, поскольку сеанс не был начат.

sqluvget - Чтение данных с устройства

После инициализации эту функцию можно вызывать для чтения данных с устройства.

Полномочия

Одни из следующих:

- *sysadm*
- *dbadm*

Необходимое соединение

База данных

Файл включения API

sqluvend.h

Синтаксис API на языке C

```
/* Файл: sqluvend.h */
/* API: Чтение данных с устройства */
/* ... */
int sqluvget (
  void * pVendorCB,
  struct Data         *,
  struct Return_code  *);
/* ... */

typedef struct Data
}
   sqlint32  obj_num;
   sqlint32  buff_size;
   sqlint32  actual_buff_size;
   void      *dataptr;
   void      *reserve;
{ Data;
```

Параметры API

pVendorCB
Входной. Указатель на область, выделенную для структур DATA (включая буфер данных) и Return_code.

Data Входной/выходной. Указатель на структуру *data*.

Return_code
Выходной. Код возврата вызова API.

obj_num
Указывает, какой объект резервной копии следует получить.

buff_size
Указывает, какой использовать размер буфера.

actual_buff_size

Указывает, сколько байт в действительности прочитано или записано. Это значение надо задать при выходе, чтобы показать, сколько было прочитано байт.

dataptr

Указатель на буфер данных.

reserve

Зарезервирован для будущего использования.

Замечания по использованию

Эта функция используется утилитой восстановления.

Коды возврата

Таблица 26. Допустимые коды возврата sqluvget и ответные действия DB2

Литерал в файле заголовка	Описание	Вероятный следующий вызов	Другие комментарии
SQLUV_OK	Операция выполнена успешно.	sqluvget	DB2 обрабатывает данные
SQLUV_COMM_ERROR	Ошибка связи с устройством.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
SQLUV_INV_ACTION	Затребовано недопустимое действие.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
SQLUV_INV_DEV_HANDLE	Неверный хэндл устройства.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
SQLUV_INV_BUFF_SIZE	Задан неверный размер буфера.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
SQLUV_DEV_ERROR	Ошибка устройства.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
SQLUV_WARNING	Предупреждение. Его не надо использовать для сообщения DB2 о конце носителя, для этого используйте SQLUV_ENDOFMEDIA или SQLUV_ENDOFMEDIA_NO_DATA. Однако состояние неготовности устройства можно показывать с помощью этого кода возврата.	sqluvget или sqluvend, action = SQLU_ABORT	Объяснение, как DB2 обрабатывает предупреждения, смотрите в разделе “Состояния предупреждения” на стр. 487.
SQLUV_LINK_NOT_EXIST	Связь в данный момент не существует.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
SQLUV_MORE_DATA	Операция выполнена успешно; есть еще данные.	sqluvget	
SQLUV_ENDOFMEDIA_NO_DATA	Конец носителя, считано 0 байт (например, конец ленты).	sqluvend	Объяснение, как DB2 обрабатывает состояние конца носителя, смотрите в разделах “Режим PROMPTING” на стр. 484 и “Характеристики устройства” на стр. 485.
SQLUV_ENDOFMEDIA	Конец носителя, считано > 0 байт (например, конец ленты).	sqluvend	DB2 обрабатывает данные, а потом обрабатывает состояние конца носителя как описано в разделах “Режим PROMPTING” на стр. 484 и “Характеристики устройства” на стр. 485.
SQLUV_IO_ERROR	Ошибка ввода/вывода.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.

Таблица 26. Допустимые коды возврата sqluvget и ответные действия DB2 (продолжение)

Литерал в файле заголовка	Описание	Вероятный следующий вызов	Другие комментарии
Следующий вызов: [a] Если следующий вызов - sqluvend, action = SQLU_ABORT, этот сеанс и все другие активные сеансы будут прерваны.			

sqluvput - Запись данных на устройство

После инициализации эту функцию можно использовать, чтобы записывать данные на устройство.

Полномочия

Одни из следующих:

- *sysadm*
- *dbadm*

Необходимое соединение

База данных

Файл включения API

sqluvend.h

Синтаксис API на языке C

```
/* Файл: sqluvend.h */
/* API: Запись данных на устройство */
/* ... */
int sqluvput (
  void * pVendorCB,
  struct Data  *,
  struct Return_code  *);
/* ... */

typedef struct Data
}
   sqlint32  obj_num;
   sqlint32  buff_size;
   sqlint32  actual_buff_size;
   void      *dataptr;
   void      *reserve;
{ Data;
```

Параметры API

pVendorCB
Входной. Указатель на область, выделенную для структур DATA (включая буфер данных) и Return_code.

Data Выходной. Буфер данных, заполненный данными для записи на устройство.

Return_code
Выходной. Код возврата вызова этого API.

obj_num
Указывает, какой объект резервной копии следует получить.

buff_size

Указывает, какой использовать размер буфера.

actual_buff_size

Указывает, сколько байт в действительности прочитано или записано. Это значение надо установить, чтобы показать, сколько было прочитано байт.

dataptr

Указатель на буфер данных.

reserve

Зарезервирован для будущего использования.

Замечания по использованию

Эта функция используется утилитой резервного копирования.

Коды возврата

Таблица 27. Допустимые коды возврата sqluvput и ответные действия DB2

Литерал в файле заголовка	Описание	Вероятный следующий вызов	Другие комментарии
SQLUV_OK	Операция выполнена успешно.	sqluvput или sqluvend, если запись завершена (например, у DB2 больше нет данных)	Сообщает другим процессам о успешном выполнение операции.
SQLUV_COMM_ERROR	Ошибка связи с устройством.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
SQLUV_INV_ACTION	Затребовано недопустимое действие.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
SQLUV_INV_DEV_HANDLE	Неверный хэндл устройства.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
SQLUV_INV_BUFF_SIZE	Задан неверный размер буфера.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
SQLUV_ENDOFMEDIA	Достигнут конец носителя, например конец ленты.	sqluvend	Объяснение, как DB2 обрабатывает состояние конца носителя, смотрите в разделах “Режим PROMPTING” на стр. 484 и “Характеристики устройства” на стр. 485.
SQLUV_DATA_RESEND	Устройство затребовало повторной посылки буфера.	sqluvput	DB2 передаст последний буфер повторно. Это будет сделано только один раз.
SQLUV_DEV_ERROR	Ошибка устройства.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
SQLUV_WARNING	Предупреждение. Его не надо использовать для сообщения DB2 о конце носителя, для этого используйте SQLUV_ENDOFMEDIA. Однако состояние неготовности устройства можно показывать с помощью этого кода возврата.	sqluvput	Объяснение, как DB2 обрабатывает предупреждения, смотрите в разделе “Состояния предупреждения” на стр. 487.
SQLUV_LINK_NOT_EXIST	Связь в данный момент не существует.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
SQLUV_IO_ERROR	Ошибка ввода/вывода.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.

Таблица 27. Допустимые коды возврата sqluvput и ответные действия DB2 (продолжение)

Литерал в файле заголовка	Описание	Вероятный следующий вызов	Другие комментарии
Следующий вызов: [a] Если следующий вызов - sqluvend, action = SQLU_ABORT, этот сеанс и все другие активные сеансы будут прерваны. Принятые сеансы удаляются последовательным вызовом sqluvint, sqluvdel и sqluvend (смотрите раздел “Если DB2 возвращаются состояния ошибки” на стр. 487).			

sqluvend - Отсоединить устройство и освободить его ресурсы

Завершает работу с устройством или отсоединяет его и освобождает все связанные с ним ресурсы. Перед возвратом в DB2 поставщик должен освободить неиспользуемые ресурсы (например, выделенную память и хэндлы файлов).

Полномочия

Одни из следующих:

- *sysadm*
- *dbadm*

Необходимое соединение

База данных

Файл включения API

sql.h

Синтаксис API на языке C

```
/* Файл: sqluvend.h */
/* API: Отсоединить устройство и освободить его ресурсы */
/* ... */
int sqluvend (
  sqlint32 action,
  void * pVendorCB,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

Параметры API

action Входной. Используется для принятия или прекращения сеанса:

- SQLUV_COMMIT (0 = принятие)
- SQLUV_ABORT (1 = прекращение)

pVendorCB

Входной. Указатель на структуру Init_output.

Init_output

Выходной. Место Init_output освобождено. Для резервного копирования если действие - принятие, то данные были приняты и записаны на постоянное хранение. Если действие - прекращение сеанса, то данные были удалены.

Код возврата

Выходной. Код возврата вызова API.

Замечания по использованию

Эта функция вызывается для каждого открытого сеанса. Есть два возможных кода действия:

- Commit

  Вывод данных в этот сеанс или чтение данных из него завершено.

  Для сеанса записи (резервного копирования), если поставщик возвращает в DB2 кодSQLUV_OK, DB2 считает, что выходные данные правильно сохранены программой поставщика, и к ним можно получить обращаться в дальнейшем при помощи вызова **sqluvint**.

  Для сеанса чтения (восстановления из резервной копии), если поставщик возвращает в DB2 код SQLUV_OK, данные не следует удалять, поскольку они могут потребоваться еще раз.

  Если поставщик возвращает SQLUV_COMMIT_FAILED, DB2 считает, что операция резервного копирования или восстановления завершилась неудачно. Все активные сеансы прерываются вызовами **sqluvend** с параметром action = SQLUV_ABORT. Для операции резервного копирования принятые сеансы получают последовательность вызовов **sqluvint**, **sqluvdel** и **sqluvend** (смотрите “Если DB2 возвращаются состояния ошибки” на стр. 487).

- Abort

  DB2 столкнулась с ошибкой, и чтение данных из этого сеанса или запись в него прекращается.

  Для сеанса записи (резервного копирования) поставщик должен удалить незавершенный выходной набор данных, и возвратить код SQLUV_OK, если этот набор удален. DB2 считает, что с операция резервного копирования завершилась неудачно. Все активные сеансы прерываются при помощи вызовов **sqluvend** с action = SQLUV_ABORT, а принятые сеансы получают последовательность вызовов **sqluvint**, **sqluvdel** и **sqluvend** (смотрите “Если DB2 возвращаются состояния ошибки” на стр. 487).

  Для сеанса чтения (восстановления) поставщик не должен удалять данные (поскольку они могут потребоваться еще раз), но должен освободить используемые ресурсы и возвратить DB2 код SQLUV_OK. DB2 прерывает все сеансы восстановления при помощи вызовов **sqluvend** с параметром action = SQLUV_ABORT. Если поставщик возвращает в DB2 код SQLUV_ABORT_FAILED, вызывающая программа не получает сообщения об этой ошибке, так как DB2 возвращает только первую неисправимую ошибку, игнорируя все последующие. В этом случае, чтобы DB2 вызвала **sqluvend** с параметром action = SQLUV_ABORT, вначале должна произойти неисправимая ошибка.

Коды возврата

Таблица 28. Допустимые коды возврата sqluvend и соответствующие им действия DB2

Литерал в файле заголовка	Описание	Вероятный следующий вызов	Другие комментарии
SQLUV_OK	Операция завершена успешно.	последующих вызовов нет	Освобождение всей выделенной для этого сеанса памяти и завершение работы.
SQLUV_COMMIT_FAILED	Требование принятия завершилось неудачно.	последующих вызовов нет	Освобождение всей выделенной для этого сеанса памяти и завершение работы.
SQLUV_ABORT_FAILED	Требование прекращения сеанса завершилось неудачно.	последующих вызовов нет	

sqluvdel - Удалить сеанс после принятия

Удаляет сеансы после принятия.

Полномочия

Одни из следующих:

- *sysadm*
- *dbadm*

Необходимое соединение

База данных

Файл включения API

sqluvend.h

Синтаксис API на языке C

```
/* Файл: sqluvend.h */
/* API: Удалить сеанс после принятия */
/* ... */
int sqluvdel (
  struct Init_input   *,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

Параметры API

Init_input
Входной. Память, выделенная для Init_input и Return_code.

Return_code
Выходной. Код возврата из вызова API. Объект, на который указывает структура Init_input, удаляется.

Замечания по использованию

Если было открыто несколько сеансов и некоторые из них приняты, а один завершился неудачно, эта функция вызывается для удаления принятых сеансов. Порядковые номера указывать не надо; **sqluvdel** ответственна за поиск всех объектов, созданных в процессе конкретной операции резервного копирования, а также за их удаление. Для идентификации удаляемых выходных данных используется информация из структуры INIT-INPUT. Вызов **sqluvdel** несет ответственность за установление любых соединений или сеансов, необходимых для удаления объекта резервной копии с устройства поставщика. Если этот вызов возвращает SQLUV_DELETE_FAILED, DB2 не уведомляет об этом вызывающую программу, поскольку DB2 возвращает первую неисправимую ошибку, а все последующие игнорирует. То есть для того, чтобы DB2 вызвала **sqluvdel**, должна произойти первичная неисправимая ошибка.

Коды возврата

Таблица 29. Допустимые коды возврата для sqluvdel и ответные действия DB2

Литерал в файле заголовка	Описание	Вероятный следующий вызов	Прочие комментарии
SQLUV_OK	Операция успешно завершена.	последующих вызовов нет	
SQLUV_DELETE_FAILED	Запрос на удаление завершился неудачно.	последующих вызовов нет	

DB2-INFO

Эта структура содержит информацию, идентифицирующую DB2 для устройства поставщика.

Таблица 30. Поля структуры DB2-INFO. Все поля представляют собой символьные строки с нулевым символом-ограничителем.

Имя поля	Тип данных	Описание
DB2_id	char	Идентификатор продукта DB2. Максимальная длина строки, на которую он указывает, составляет 8 символов.
version	char	Текущая версия продукта DB2. Максимальная длина строки, на которую он указывает, составляет 8 символов.
release	char	Текущий выпуск продукта DB2. Если он несуществен, задайте значение NULL. Максимальная длина строки, на которую он указывает, составляет 8 символов.
level	char	Текущий уровень продукта DB2. Если он несуществен, задайте значение NULL. Максимальная длина строки, на которую он указывает, составляет 8 символов.
action	char	Задает действие, которое нужно выполнить. Максимальная длина строки, на которую он указывает - 1 символ.
filename	char	Имя файла, определяющее резервную копию. Если в этом поле задан NULL, резервная копия однозначно определяется полями *server_id, db2instance, dbname* и *timestamp*. Максимальная длина строки, на которую он указывает - 255 символов.
server_id	char	Уникальное имя, определяющее сервер, где находится база данных. Максимальная длина строки, на которую он указывает, составляет 8 символов.
db2instance	char	ID db2instance. Это ID пользователя, вызывающего команду. Максимальная длина строки, на которую он указывает, составляет 8 символов.

Таблица 30. Поля структуры DB2-INFO (продолжение). Все поля представляют собой символьные строки с нулевым символом-ограничителем.

Имя поля	Тип данных	Описание

type	char	Задает тип выполняемого резервного копирования или восстановления. Допустимые значения:
		
		Для action = SQLUV_WRITE:
		`0 - полное резервное копирование базы данных`
		`3 - резервное копирование уровня табличного пространства`
		
		При action = SQLUV_READ:
		`0 - полное восстановление`
		`3 - оперативное восстановление табличного пространства`
		`4 - восстановление табличного пространства`
		`5 - восстановление файла хронологии`

dbname	char	Имя базы данных, для которой выполняется резервное копирование или восстановление. Максимальная длина строки, на которую он указывает, составляет 8 символов.
alias	char	Алиас базы данных, для которой выполняется резервное копирование или восстановление. Максимальная длина строки, на которую он указывает, составляет 8 символов.
timestamp	char	Отметка времени, определяющая резервную копию. Максимальная длина строки, на которую указывает это поле - 26 символов.
sequence	char	Задает расширение файла для резервной копии. При операциях записи значение для первого сеанса равно 1; каждый раз, когда вызов sqluvint запускает очередной сеанс, оно увеличивается на 1. При операциях чтения значение всегда равно нулю. Максимальная длина строки, на которую указывает это поле - 3 символа.
obj_list	struct sqlu_gen_list	Перечисляет объекты, содержащиеся в резервной копии. Эта информация предназначается только для поставщиков.
max_bytes_per_txn	sqlint32	Задает для поставщика заданное пользователем число байтов в заданном буфере передачи.
image_filename	char	Зарезервирован для будущего использования.
reserve	void	Зарезервирован для будущего использования.
nodename	char	Имя узла, на котором была создана резервная копия.

Таблица 30. Поля структуры DB2-INFO (продолжение). Все поля представляют собой символьные строки с нулевым символом-ограничителем.

Имя поля	Тип данных	Описание
password	char	Пароль для узла, на котором была создана резервная копия.
owner	char	ID пользователя, запустившего резервное копирование.
mcNameP	char	Класс управления.
nodeNum	SQL_PDB_NODE_TYPE	Номер узла. Номера, превышающие 255, поддерживаются интерфейсом поставщика.

Резервная копия однозначно определяется либо полем *filename*, либо полями *server_id*, *db2instance*, *type*, *dbname* и *timestamp*. Последовательное число, заданное в поле *sequence*, определяет расширение файла. При восстановлении из резервной копии для получения нужной резервной копии нужно задать те же значения. В зависимости от продукта поставщика, если задано поле *filename*, остальные параметры могут быть равны NULL, и наоборот.

Синтаксис языка

Структура на языке C

```
/* Файл: sqluvend.h */
/* ... */
typedef struct DB2_info
{
  char                  *DB2_id;
  char                  *version;
  char                  *release;
  char                  *level;
  char                  *action;
  char                  *filename;
  char                  *server_id;
  char                  *db2instance;
  char                  *type;
  char                  *dbname;
  char                  *alias;
  char                  *timestamp;
  char                  *sequence;
  struct sqlu_gen_list  *obj_list;
  long                  max_bytes_per_txn;
  char                  *image_filename;
  void                  *reserve;
  char                  *nodename;
  char                  *password;
  char                  *owner;
  char                  *mcNameP;
  SQL_PDB_NODE_TYPE     nodeNum;
} DB2_info;
/* ... */
```

VENDOR-INFO

Эта структура содержит информацию, идентифицирующую поставщика и версию устройства.

Таблица 31. Поля структуры VENDOR-INFO. Все поля представляют собой символьные строки с нулевым символом-ограничителем.

Имя поля	Тип данных	Описание
vendor_id	char	Идентификатор поставщика. Максимальная длина строки, на которую он указывает, составляет 64 символа.
version	char	Текущая версия продукта поставщика. Максимальная длина строки, на которую он указывает, составляет 8 символов.
release	char	Текущий выпуск продукта поставщика. Если он несуществен, задайте значение NULL. Максимальная длина строки, на которую он указывает, составляет 8 символов.
level	char	Текущий уровень продукта поставщика. Если он несуществен, задайте значение NULL. Максимальная длина строки, на которую он указывает, составляет 8 символов.
server_id	char	Уникальное имя, идентифицирующее сервер, на котором находится база данных. Максимальная длина строки, на которую он указывает, составляет 8 символов.
max_bytes_per_txn	sqlint32	Максимальный поддерживаемый размер буфера передачи в байтах. Задается поставщиком. Используется только в том случае, когда функция инициализации поставщика возвращает код SQLUV_BUFF_SIZE, что означает неверный размер буфера.
num_objects_in_backup	sqlint32	Число сеансов, использованных для изготовления полной резервной копии. Используется во время операции восстановления из резервной копии для определения завершения обработки всех образов резервных копий.
reserve	void	Зарезервирован для будущего использования.

Синтаксис языка

Структура на языке C

```
typedef struct Vendor_info
{
  char       *vendor_id;
  char       *version;
  char       *release;
  char       *level;
  char       *server_id;
  sqlint32   max_bytes_per_txn;
  sqlint32   num_objects_in_backup;
  void       *reserve;
} Vendor_info;
```

INIT-INPUT

Эта структура содержит информацию, предоставляемую DB2 для установки логической связи с устройством поставщика.

Таблица 32. Поля структуры INIT-INPUT. Все поля представляют собой символьные строки с нулевым символом-ограничителем.

Имя поля	Тип данных	Описание
DB2_session	struct DB2_info	Описание сеанса с точки зрения DB2.
size_options	unsigned short	Длина поля опций. При использовании функции резервного копирования или восстановления из резервной копии DB2 данные в это поле передаются непосредственно из параметра *VendorOptionsSize*.
size_HI_order	sqluint32	Старшие 32 бита оценки размера базы данных в байтах; общий размер - 64 бита.
size_LOW_order	sqluint32	Младшие 32 бита оценки размера базы данных в байтах; общий размер - 64 бита.
options	void	Эта информация передается из прикладной программы, когда вызывается функция резервного копирования или восстановления из резервной копии. Эта структура должна быть плоской; другими словами, уровни косвенности не поддерживаются. Обращение байтов не производится, и кодовая страница для этих данных не проверяется. При использовании функции резервного копирования или восстановления из резервной копии DB2 данные в это поле передаются непосредственно из параметра *pVendorOptions*.
reserve	void	Зарезервирован для будущего использования.
prompt_lvl	char	Уровень подсказок пользователю при вызове операции резервного копирования или восстановления из резервной копии. Максимальная длина строки, на которую он указывает - 1 символ.

Таблица 32. Поля структуры INIT-INPUT (продолжение). Все поля представляют собой символьные строки с нулевым символом-ограничителем.

Имя поля	Тип данных	Описание
num_sessions	unsigned short	Число сеансов, запрошенное пользователем при вызове операции резервного копирования или восстановления из резервной копии.

Синтаксис языка

Структура на языке C

```
typedef struct Init_input
{
  struct DB2_info  *DB2_session;
  unsigned short   size_options;
  sqluint32        size_HI_order;
  sqluint32        size_LOW_order;
  void             *options;
  void                *reserve;
  char             *prompt_lvl;
  unsigned short   num_sessions;
} Init_input;
```

INIT-OUTPUT

Эта структура содержит выходную информацию, передаваемую устройством поставщика.

Таблица 33. Поля в структуре INIT-OUTPUT

Имя поля	Тип данных	Описание
vendor_session	struct Vendor_info	Содержит информацию, описывающую поставщика для DB2.
pVendorCB	void	Блок управления поставщика.
reserve	void	Зарезервирован для будущего использования.

Синтаксис языка

Структура на языке C

```
typedef struct Init_output
{
  struct Vendor_info  *vendor_session;
  void                *pVendorCB;
  void                *reserve;
} Init_output;
```

DATA

Эта структура содержит данные, переданные между DB2 и устройством поставщика.

Таблица 34. Поля структуры DATA

Имя поля	Тип данных	Описание
obj_num	sqlint32	Последовательный номер, назначенный DB2 во время операции резервного копирования.
buff_size	sqlint32	Размер буфера.
actual_buf_size	sqlint32	Реальное число посланных или полученных байтов. Не должно превышать *buff_size*.
dataptr	void	Указатель на буфер данных. DB2 выделяет место для буфера.
reserve	void	Зарезервирован для будущего использования.

Синтаксис языка

Структура на языке C

```
typedef struct Data
{
  sqlint32  obj_num;
  sqlint32  buff_size;
  sqlint32  actual_buff_size;
  void      *dataptr;
  void      *reserve;
} Data;
```

RETURN-CODE

Эта структура содержит код возврата и краткое описание ошибки, возвращенной DB2.

Таблица 35. Поля структуры RETURN-CODE

Имя поля	Тип данных	Описание
return_code[a]	sqlint32	Код возврата из функции поставщика.
description	char	Краткое описание кода возврата.
reserve	void	Зарезервирован для будущего использования.
[a] Это специфичный для поставщика код возврата, который не совпадает со значениями, возвращаемыми различными API DB2. Коды возврата, принимаемые от продуктов поставщика, смотрите в описаниях отдельных API.		

Синтаксис языка

Структура на языке C

```
typedef struct Return_code
{
  sqlint32  return_code,
  char      description[60],
  void      *reserve,
} Return_code;
```

Вызов операции резервного копирования или восстановления при помощи продуктов других поставщиков

Продукты других поставщиков можно задавать, когда для вызова утилит резервного копирования и восстановления DB2 используются:

- Центр управления
- командная строка (CLP)
- API (application programming interface - интерфейс прикладного программирования).

Центр управления

Центр управления - это графический пользовательский интерфейс для управления базами данных, поставляемый с DB2.

Задаваемый параметр	**Входная переменная Центра управления для операций резервного копирования и восстановления**
Использование устройства и библиотеки другого поставщика	*Use Library*. Задает имя библиотеки (в системах на основе UNIX) или DLL (в операционной системе Windows или OS/2).
Число сеансов	*Sessions*
Опции другого поставщика	Не поддерживается
Имя файла другого поставщика	Не поддерживается
Размер буфера передачи	При резервном копировании *Size of each Buffer*, при восстановлении не задается.

Командная строка (CLP)

В командной строке (CLP) можно ввести команды DB2 BACKUP DATABASE и RESTORE DATABASE.

Задаваемый параметр	**Параметр командной строки**	
	при резервном копировании	**при восстановлении**
Использование устройства и библиотеки другого поставщика	*library-name*	*shared-library*
Число сеансов	*num-sessions*	*num-sessions*
Опции другого поставщика	не поддерживается	не поддерживается
Имя файла другого поставщика	не поддерживается	не поддерживается
Размер буфера передачи	*buffer-size*	*buffer-size*

Вызов операции резервного копирования или восстановления при помощи продуктов других поставщиков

API (Application Programming Interface, интерфейс прикладного программирования)

Операции резервного копирования и восстановления поддерживаются двумя вызовами функций API: **sqlubkp** для резервного копирования (смотрите разделы, “API резервного копирования базы данных” на стр. 99) и **sqlurestore** для восстановления (смотрите разделы, “API восстановления базы данных” на стр. 129.)

Задаваемый параметр	**Параметр API (для sqlubkp и sqlurestore)**
Использование устройства и библиотеки другого поставщика	В структуре *sqlu_media_list* задайте тип носителя SQLU_OTHER_MEDIA, затем в структуре *sqlu_vendor* задайте совместно используемую библиотеку или DLL в *shr_lib*.
Число сеансов	В структуре *sqlu_media_list* задайте *sessions*.
Опции другого поставщика	*PVendorOptions*
Имя файла другого поставщика	В структуре *sqlu_media_list* задайте тип носителя SQLU_OTHER_MEDIA, затем в структуре *sqlu_vendor* задайте имя файла в *filename*.
Размер буфера передачи	*BufferSize*

Приложение J. 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. 534. Вы можете просматривать сведения о задачах, книги DB2, информацию по устранению неисправностей, программы примеров и информацию по DB2 в Web.

Файлы PDF и печатные книги DB2

Информация DB2

В следующей таблице книги DB2 разделены на 4 категории:

Руководства и справочники по DB2
В этих книгах содержится информация по DB2, общая для всех платформ.

Информация по установке и конфигурированию DB2
Эти книги применимы к DB2 для конкретной платформы. Например, есть отдельные книги *Быстрый старт* для DB2 на OS/2, Windows и на платформах на основе UNIX.

Кроссплатформенные программы примеров в формате HTML
Эти примеры - HTML-версии программ примеров, которые устанавливаются с клиентом разработки программ. Примеры используются для справок и не заменяют самих программ.

Замечания по выпуску
Эти файлы содержат самую свежую информацию, которую не успели включить в книги по DB2.

Руководства по установке, замечания по выпуску и обучающие книги в формате HTML можно просматривать прямо на компакт-диске. Большинство книг доступны в формате HTML на компакт-диске данного продукта (для просмотра) и в формате Adobe Acrobat (PDF) на компакт-диске публикаций DB2 (для просмотра и печати). Можно также заказать печатные копии в IBM; смотрите раздел “Заказ печатных копий” на стр. 530. Ниже в таблице перечислены книги, которые можно заказать.

© Copyright IBM Corp. 2001

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. 533
- “Поиск электронной информации” на стр. 538
- “Заказ печатных копий” на стр. 530
- “Печать книг PDF” на стр. 529

Таблица 36. Информация DB2

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
	Руководства и справочники по DB2		
Administration Guide	*Administration Guide: Planning* содержит обзор понятий баз данных, информацию по вопросам разработки (в частности, по логическому и физическому проектированию баз данных) и обсуждение доступности баз данных. *Administration Guide: Implementation* содержит информацию о реализации ваших проектов, доступе к базам данных, аудите, резервном копировании и восстановлении. *Administration Guide: Performance* содержит информацию о среде баз данных, оценке и настройке производительности программ. Эти три тома *Administration Guide* можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8934.	SH43-0146 db2d1x70 SH43-0144 db2d2x70 SH43-0145 db2d3x70	db2d0

Таблица 36. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
Administrative API Reference	Описывает интерфейсы прикладного программирования (API) DB2 и структуры данных, которые можно использовать при работе с вашими базами данных. Эта книга также объясняет, как вызывать API из ваших программ.	SC09-2947 db2b0x70	db2b0
Application Building Guide	Содержит информацию о настройке среды и пошаговые инструкции для компиляции, компоновки и запуска программ DB2 в системах Windows, OS/2 и на платформах на базе UNIX.	SC09-2948 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	Содержит общие сведения о смысловых кодах APPC, CPI-C и SNA, которые могут встретиться вам при работе с продуктами DB2 Universal Database. Существует только в формате HTML.	Номера формы нет db2apx70	db2ap
Application Development Guide	Объясняет, как разрабатывать программы, обращающиеся к базам данных DB2 с использованием встроенного SQL или Java (JDBC и SQLJ). Эта книга содержит обсуждение программирования хранимых процедур, пользовательских функций, создания пользовательских типов, использования триггеров и разработки прикладных программ для работы в многораздельной среде и в системах объединения.	SC09-2949 db2a0x70	db2a0
CLI Guide and Reference	Объясняет, как разрабатывать программы, обращающиеся к базам данных DB2 при помощи интерфейса уровня вызовов (CLI) DB2 - интерфейса SQL, совместимого со спецификациями Microsoft ODBC.	SC09-2950 db2l0x70	db2l0
Command Reference	Объясняет, как использовать процессор командной строки, и описывает команды DB2, которые можно использовать для управления вашей базой данных.	SC09-2951 db2n0x70	db2n0

Таблица 36. Информация DB2 (продолжение)

Название	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 хранилищ данных.	SC26-9993 db2ddx70	db2dd
Data Warehouse Center Application Integration Guide	Содержит информацию, которая поможет программистам интегрировать прикладные программы с Центром хранилищ данных и Менеджером каталогов 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
Глоссарий	Содержит определения терминов, используемых в DB2 и его компонентах. Доступен в формате HTML, а также в книге *SQL Reference*.	Номера формы нет db2t0x70	db2t0

Таблица 36. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
Image, Audio, and Video Extenders Administration and Programming	Содержит общую информацию о модулях расширения DB2, о конфигурировании модулей расширения для работы с изображениями, звуком и видео (IAV), об управлении ими и о программировании с использованием модулей расширения IAV. Включает в себя справочную информацию, диагностическую информацию (с сообщениями) и примеры.	SC26-9929 dmbu7x70	dmbu7
Information Catalog Manager Administration Guide	Руководство по управлению каталогами данных.	SC26-9995 db2dix70	db2di
Information Catalog Manager Programming Guide and Reference	Содержит определения для проектирования интерфейсов менеджера каталогов данных.	SC26-9997 db2bix70	db2bi
Information Catalog Manager User's Guide	Содержит информацию об использовании пользовательского интерфейса менеджера каталога данных.	SC26-9996 db2aix70	db2ai
Дополнение по установке и конфигурированию	Помогает планировать, устанавливать и конфигурировать клиенты DB2 для конкретных платформ. Это дополнение содержит также информацию по связыванию, конфигурированию связей клиента и сервера, инструментам DB2 с графическим интерфейсом, DRDA AS, распределенной установке, конфигурации распределенных запросов и доступу к неоднородным источникам данных.	GH43-0126 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 Справочник по сообщениям можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8932.	Том 1 SC09-2978 db2m1x70 Том 2 SC09-2979 db2m2x70	db2m0
OLAP Integration Server Administration Guide	Объясняет, как использовать менеджер управления сервером OLAP Integration Server.	SC27-0782 db2dpx70	нет

Таблица 36. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
OLAP Integration Server Metaoutline User's Guide	Объясняет, как создавать и заполнять метамакеты OLAP при помощи стандартного интерфейса метамакетов OLAP (а не при помощи Metaoutline Assistant).	SC27-0784 db2upx70	нет
OLAP Integration Server Model User's Guide	Объясняет, как создавать и заполнять метамакеты OLAP при помощи стандартного интерфейса моделей OLAP (а не при помощи Model Assistant).	SC27-0783 db2lpx70	нет
Руководство по установке и использованию OLAP	Содержит информацию о конфигурировании и установке для Начального комплекта OLAP.	SC27-0702 db2ipx70	db2ip
Руководство пользователя надстройки электронных таблиц для Excel	Описывает, как использовать программу электронных таблиц Excel для анализа данных OLAP.	SC27-0786 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.	SC27-0785 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 36. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
SQL Reference, Том 1 и Том 2	Описывает синтаксис SQL, его семантику и правила языка. Эта книга включает также информацию о совместимости версий, ограничения продукта и обзор каталогов. Оба тома *SQL Reference* можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8933.	Том 1 SC09-2974 db2s1x70 Том 2 SC09-2975 db2s2x70	db2s0
System Monitor Guide and Reference	Описывает сбор различной информации о базах данных и менеджере баз данных. Эта книга объясняет, как использовать информацию, чтобы понять работу с базой данных, улучшить производительность и найти причины ошибок.	SC09-2956 db2f0x70	db2f0
Text Extender Administration and Programming	Содержит общую информацию о модулях расширения DB2, о конфигурировании модуля расширения для работы с текстом, об управлении им и о программировании с использованием модулей расширения для работы с текстом. Включает в себя справочную информацию, диагностическую информацию (с сообщениями) и примеры.	SC26-9930 desu9x70	desu9
Troubleshooting Guide	Помогает определить причины ошибок, выполнить восстановительные операции, и использовать средства диагностики, консультируясь со Службой заказчиков DB2.	GC09-2850 db2p0x70	db2p0
Что нового	Описывает новые возможности, функции и усовершенствования в DB2 Universal Database Версии 7.	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 36. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
DB2 Connect Enterprise Edition for UNIX Quick Beginnings	Содержит информацию по планированию, установке, конфигурированию и выполнению заданий для DB2 Connect Enterprise Edition на платформах на основе UNIX. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2952 db2cyx70	db2cy
DB2 Connect Personal Edition 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 36. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
DB2 for OS/2 Быстрый старт	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database Personal Edition в операционной системе OS/2. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2968 db2i2x70	db2i2
DB2 for UNIX Быстрый старт	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database на платформах на основе UNIX. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2970 db2ixx70	db2ix
DB2 for Windows Быстрый старт	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database в 32-битных системах Windows. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GH43-0160 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 каталогов данных.	GC26-9998 db2idx70	db2id
Кроссплатформенные программы примеров в формате HTML			

Таблица 36. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
Программы примеров в виде HTML	Содержит для справок программы примеров в виде HTML для языков программирования на всех платформах, поддерживаемых DB2. Эти программы примеров приводятся только в информационных целях. Не все из них доступны на всех языках программирования. Примеры HTML доступны, только если установлен клиент разработки программ DB2. Дополнительную информацию об этих программах смотрите в книге *Application Building Guide*.	Номера формы нет	db2hs
	Замечания по выпуску		
Замечания по выпуску DB2 Connect	Содержит самую свежую информацию, которую не успели включить в книги по DB2 Connect.	Смотрите примечание 2.	db2cr
Замечания по установке DB2	Содержит самую свежую информацию по установке, которую не успели включить в книги по DB2.	Доступна только на компакт-диске продукта.	
Замечания по выпуску DB2	Содержит самую свежую информацию о всех продуктах DB2 и их возможностях, которую не успели включить в книги по DB2.	Смотрите примечание 2.	db2ir

Примечания:

1. Символ *x* на шестой позиции в имени файла указывает язык книги. Например, имя файла `db2d0e70` говорит о том, что это английская версия книги *Administration Guide*, а имя файла `db2d0f70` соответствует французской версии этой же книги. Для обозначений языков используются на шестой позиции имени файла используются следующие буквы:

Язык	Обозначение
Английский	e
Болгарский	u
Бразильский португальский	b
Венгерский	h
Голландский	q
Греческий	a
Датский	d

Испанский	z
Итальянский	i
Корейский	k
Немецкий	g
Норвежский	n
Польский	p
Португальский	v
Русский	r
Словенский	l
Традиционный китайский	t
Турецкий	m
Упрощенный китайский	c
Финский	y
Французский	f
Чешский	x
Шведский	s
Японский	j

2. Последнюю информацию, которую не успели включить в книги по DB2, смотрите в Замечаниях по выпуску в формате HTML и в виде ASCII-файла. HTML-версию можно вызвать через Информационный центр или с компакт-диска продукта. Чтобы посмотреть ASCII-файл:
   - На платформах на базе UNIX смотрите файл `Release.Notes`. Он расположен в каталоге `DB2DIR/Readme/%L`, где `%L` - национальная версия, а `DB2DIR`:
     - `/usr/lpp/db2_07_01` в AIX
     - `/opt/IBMdb2/V7.1` в HP-UX, PTX, Solaris, и Silicon Graphics IRIX
     - `/usr/IBMdb2/V7.1` в Linux.
   - На других платформах смотрите файл `RELEASE.TXT`. Он находится в каталоге, где установлен продукт. На платформах OS/2 можно также дважды щелкнуть по папке **IBM DB2**, а затем дважды щелкнуть по значку **Release Notes**.

Печать книг PDF

Если вы предпочитаете использовать печатные версии книг, можно напечатать файлы .pdf с компакт-диска публикаций по DB2. При помощи Adobe Acrobat Reader можно напечатать книгу целиком или же определенный диапазон страниц. Имена файлов для каждой книги в библиотеке приводятся в Табл. 36 на стр. 520.

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 дисковод компакт-диск с публикациями DB2. На платформах на основе UNIX смонтируйте компакт-диск с публикациями DB2. Процедуру монтирования посмотрите в книге *Быстрый старт*.
2. Запустите Acrobat Reader.
3. Откройте требуемый файл PDF из одного из следующих мест:
   - На платформах OS/2 и Windows:

     Из каталога *x*:\doc*язык*, где *x* - буква компакт-диска, а *язык* двухсимвольный код страны, соответствующий вашему языку (например, RU для русского).
   - На платформах на основе UNIX:

     Из каталога /*cdrom*/doc/*%L* на компакт-диске, где */cdrom* - точка установки компакт-диска, а *%L* - имя требуемой национальной версии.

Можно также скопировать файлы PDF с компакт-диска на локальный или сетевой диск и читать их оттуда.

Заказ печатных копий

Печатные копии книг DB2 можно заказать по отдельности или в комплекте (только в Северной Америке) по номеру SBOF. Чтобы заказать книги, обратитесь к вашему авторизованному дилеру или торговому представителю IBM, или позвоните по телефону 1-800-879-2755 в Соединенных Штатах или 1-800-IBM-4YOU в Канаде. Можно также заказать книги на Web-странице Publications по адресу http://www.elink.ibmlink.ibm.com/pbl/pbl.

Есть два комплекта книг. SBOF-8935 содержит справочную и пользовательскую информацию для DB2 Warehouse Manager. SBOF-8931 содержит справочную и пользовательскую информацию для всех остальных продуктов и возможностей DB2 Universal Database. Содержимое каждого комплекта SBOF приводится в следующей таблице:

Таблица 37. Заказ печатных книг

Номер SBOF	Содержит книги	
SBOF-8931	• Руководство администратора: Планирование • Руководство администратора: Реализация • Руководство администратора: Производительность • 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 • Дополнение по установке и настройке • Image, Audio, and Video Extenders Administration and Programming • Справочник по сообщениям, Тома 1 и 2	• OLAP Integration Server Administration Guide • OLAP Integration Server Metaoutline User's Guide • OLAP Integration Server Model User's Guide • OLAP Integration Server User's Guide • OLAP Setup and User's Guide • OLAP Spreadsheet Add-in User's Guide for Excel • OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3 • Replication Guide and Reference • Spatial Extender Administration and Programming Guide • SQL Getting Started • SQL Reference, Volumes 1 and 2 • System Monitor Guide and Reference • Text Extender Administration and Programming • Troubleshooting Guide • Что нового
SBOF-8935	• Information Catalog Manager Administration Guide • Information Catalog Manager User's Guide • Information Catalog Manager Programming Guide and Reference	• Query Patroller Administration Guide • Query Patroller User's Guide

Электронная документация DB2

Обращение к электронной справке

Для всех компонентов DB2 доступна электронная справка. Различные типы справки перечислены в следующей таблице.

Тип справки	Содержание	Как вызвать...
Справка по командам	Объясняет синтаксис команд процессора командной строки.	В процессоре командной строки в интерактивном режиме введите: `? команда` где *команда* - ключевое слово для команды целиком. Например, `? catalog` выводит справку по всем командам CATALOG, а `? catalog database` выводит справку по команде CATALOG DATABASE.
Справка по Ассистенту конфигурирования клиента *Справка по Командному центру* *Справка по Центр управления* *Справка по Data Warehouse Center* *Справка по анализатору событий* *Справка по менеджеру каталогов 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

    Этот путь расположен 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 дисковод компакт-диск Netscape.
2. На платформах на основе UNIX смонтируйте компакт-диск. Процедуру монтирования посмотрите в книге *Быстрый старт*.
3. Прочтите инструкции по установке в файле CDNAV*nn*.txt, где *nn* - двухсимвольный идентификатор языка. Этот файл находится в корневом каталоге компакт-диска.

Доступ к информации через Информационный центр

Информационный центр обеспечивает быстрый доступ к информации о продуктах DB2. Информационный центр доступен на всех платформах, где есть инструменты администратора DB2.

Чтобы открыть Информационный центр, щелкните дважды по значку Информационный центр. В зависимости от того, в какой системе вы работаете, этот значок может находиться в основной папке продукта или в меню **Пуск**.

На платформах Windows можно также вызвать Информационный центр через панель задач и через меню **Справка** DB2.

Информационный центр дает шесть типов информации. Для обращения к информации одного из этих типов выберите соответствующую закладку.

Задания Основные задания, которые вы можете выполнить в DB2.

Справочник Справочная информацию по таким элементам DB2, как ключевые слова, команды и API.

Книги Книги DB2.

Устранение неисправностей
Список сообщений об ошибках и рекомендуемых действий по категориям.

Программы примеров
Программы примеров, поставляемые с клиентом разработки программ DB2. Если вы не установили клиент разработки программ DB2, эта закладка не выводится.

Web Информация по DB2 в WWW. Чтобы посмотреть эту информацию, ваша система должна быть подключена к Web.

Когда вы выбираете пункт в одном из списков, информационный центр запускает программу просмотра для вывода информации. Этой программой может быть программа просмотра системной справки, редактор или браузер Web в зависимости от того, какую информацию вы выбрали.

Информационный центр поддерживает возможность поиска, и вы можете искать определенную тему, не просматривая книги целиком.

Для полнотекстового поиска выберите гипертекстовую ссылку в Информационном центре и откройте поисковую форму **Поиск электронной информации DB2**.

Обычно сервер поиска HTML запускается автоматически. Если поиск информации HTML не работает, вам, возможно надо запустить сервер поиска одним из следующих способов:

В Windows
Выберите **Пуск**, затем **Программы—> IBM DB2 —> Информация —> Запустить сервер поиска HTML**.

В OS/2 Щелкните дважды по папке **DB2 for OS/2**, а затем щелкните дважды по значку **Запустить сервер поиска HTML**.

Если у вас есть проблемы с использованием поиска информации HTML, посмотрите замечания по выпуску.

Примечание: Функция поиска недоступна в средах Linux, PTX и Silicon Graphics IRIX.

Использование мастеров DB2

Мастера помогают вам выполнять конкретные задачи управления, ведя последовательно по шагам необходимых действий. Мастера доступны в Центре управления и в Ассистенте конфигурирования клиента. Список мастеров с соответствующими задачами приведен в следующей таблице.

Примечание: Мастера по созданию баз данных, индексов, конфигурированию многоузлового изменения и производительности доступны в среде многораздельных баз данных.

Мастер	Помогает вам...	Как вызвать...
по добавлению баз данных	Каталогизировать базу данных на клиентской рабочей станции	В Ассистенте конфигурирования клиента нажмите кнопку **Добавить**.
по резервному копированию базы данных	Определить, создать и включить в расписание резервное копирование.	В Центре управления щелкните правой кнопкой мыши по базе данных, для которой вам нужна резервная копия, и выберите **Резервное копирование —> Базы данных при помощи мастера**.
по конфигурированию многоузлового изменения	Конфигурировать многоузловые изменения, распределенные транзакции или двухфазное принятие.	В Центре управления щелкните правой кнопкой мыши по папке **Базы данных** и выберите **Многоузловое изменение**.
Создать базу данных	Создать базу данных и выполнить основные задачи конфигурирования.	В Центре управления щелкните правой кнопкой мыши по папке **Базы данных** и выберите **Создать —> Базу данных при помощи мастера**.
по созданию таблиц	Выбрать типы основных данных и создать первичные ключи для таблицы.	В Центре управления щелкните правой кнопкой мыши по значку **Таблицы** и выберите **Создать —> Таблицу при помощи мастера**.
по созданию табличных пространств	Создать новое табличное пространство.	В Центре управления щелкните правой кнопкой мыши по значку **Табличные пространства** и выберите **Создать —> Табличное пространство при помощи мастера**.
Создать индекс	Выбрать, какие индексы создать или отбросить для всех ваших запросов.	В Центре управления щелкните правой кнопкой мыши по значку **Индекс** и выберите **Создать —> Индекс при помощи мастера**.

Мастер	Помогает вам...	Как вызвать...
по настройке производительности	Настроить производительность базы данных, изменив параметры конфигурации в соответствии с вашими требованиями.	В Центре управления щелкните правой кнопкой мыши по базе данных, которую вы хотите настроить, и выберите **Конфигурировать производительность при помощи мастера**. Для многораздельной среды баз данных в окне Разделы баз данных щелкните правой кнопкой мыши по первому разделу баз данных, который вы хотите настроить, и выберите **Конфигурировать производительность при помощи мастера**.
по восстановлению баз данных	Восстановить базу данных после сбоя. Он поможет понять, какую резервную копию использовать, и какие журналы использовать при повторе.	В Центре управления щелкните правой кнопкой мыши по базе данных, которую вы хотите восстановить, и выберите **Восстановить —> Базу данных при помощи мастера**.

Установка сервера документации

По умолчанию информация по DB2 устанавливается в вашей локальной системе. Это значит, что каждый, кому требуется доступ к информации по DB2, должен устанавливать одни и те же файлы. Чтобы держать информацию по DB2 в едином месте, выполните следующие действия:

1. Скопируйте все файлы и подкаталоги каталога `\sqllib\doc\html` вашей локальной системы на сервер Web. Каждая книга находится в своем собственном подкаталоге, где записаны все необходимые для нее файлы HTML и GIF. Структура подкаталогов должна остаться без изменений.
2. Сконфигурируйте сервер Web на поиск файлов на новом месте. Дополнительную информацию смотрите в приложении NetQuestion руководства *Дополнение по установке и конфигурированию*.
3. Если вы используете Java-версию Информационного центра, можно задать базовый URL для всех файлов HTML. Этот URL надо использовать для списка книг.
4. Когда вы сможете просматривать файлы книг, можно пометить закладками часто используемые темы. Вероятно, вы захотите пометить закладками следующие страницы:
   - Список книг
   - Содержания часто используемых книг

- Часто требуемые статьи, например, тему ALTER TABLE
- Форму поиска

Информацию о том, как работать с файлами электронной документации на центральном компьютере, смотрите в приложении NetQuestion руководства *Дополнение по установке и конфигурированию*.

Поиск электронной информации

Для поиска информации в файлах HTML используйте один из следующих способов:

- Нажмите кнопку **Поиск** в верхнем фрейме. При помощи формы поиска найдите нужную тему. Эта функция недоступна в средах Linux, PTX и Silicon Graphics IRIX.
- Нажмите кнопку **Индекс** в верхнем фрейме. При помощи индекса найдите в книге нужную тему.
- Выведите содержание или индекс справки или книги HTML, затем при помощи функции поиска браузера Web найдите в книге нужную тему.
- При помощи функции закладок браузера Web можно быстро вернуться к определенной теме.
- Используйте для поиска определенных тем функцию поиска информационного центра. Подробности смотрите в разделе “Доступ к информации через Информационный центр” на стр. 534.

Приложение K. Замечания

IBM может предлагать описанные продукты, услуги и возможности не во всех странах. Сведения о продуктах и услугах, доступных в настоящее время в вашей стране, можно получить в местном представительстве IBM. Любые ссылки на продукты, программы или услуги IBM не означают явным или неявным образом, что можно использовать только продукты, программы или услуги IBM. Разрешается использовать любые функционально эквивалентные продукты, программы или услуги, если при этом не нарушаются права IBM на интеллектуальную собственность. Однако ответственность за оценку и проверку работы любых продуктов, программ и услуг других фирм лежит на пользователе.

Фирма IBM может располагать патентами или рассматриваемыми заявками на патенты, относящимися к предмету данного документа. Получение этого документа не означает предоставления каких-либо лицензий на эти патенты. Запросы по поводу лицензий следует направлять в письменной форме по адресу:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

По поводу лицензий, связанных с использованием наборов двухбайтных символов (DBCS), обращайтесь в отдел интеллектуальной собственности IBM в вашей стране или направьте запрос в письменной форме по адресу:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

Следующий абзац не применяется в Великобритании или в любой другой стране, где подобные заявления противоречат местным законам: КОРПОРАЦИЯ INTERNATIONAL BUSINESS MACHINES ПРЕДСТАВЛЯЕТ ДАННУЮ ПУБЛИКАЦИЮ “КАК ЕСТЬ” БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ПРЕДПОЛАГАЕМЫЕ ГАРАНТИИ СОВМЕСТИМОСТИ, РЫНОЧНОЙ ПРИГОДНОСТИ И СООТВЕТСТВИЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. В некоторых странах для определенных сделок подобные оговорки не допускаются, таким образом, это утверждение может не относиться к вам.

© Copyright IBM Corp. 2001

Данная информация может содержать технические неточности и типографские опечатки. Периодически в информацию вносятся изменения, они будут включены в новые издания этой публикации. Фирма IBM может в любое время без уведомления вносить изменения и усовершенствования в продукты и программы, описанные в этой публикации.

Любые ссылки в данной информации на Web-сайты, не принадлежащие IBM, приводятся только для удобства и никоим образом не означают поддержки IBM этих Web-сайтов. Материалы этих Web-сайтов не являются частью данного продукта IBM и вы можете использовать их только на собственную ответственность.

IBM может использовать или распространять присланную вами информацию любым способом, как фирма сочтет нужным, без каких-либо обязательств перед вами.

Если обладателю лицензии на данную программу понадобятся сведения о возможности: (i) обмена данными между независимо разработанными программами и другими программами (включая данную) и (ii) совместного использования таких данных, он может обратиться по адресу:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

Такая информация может быть предоставлена на определенных условиях (в некоторых случаях к таким условиям может относиться оплата).

Лицензированная программа, описанная в данной публикации, и все лицензированные материалы, доступные с ней, предоставляются IBM на условиях IBM Customer Agreement (Соглашения IBM с заказчиком), Международного соглашения о лицензиях на программы IBM или эквивалентного соглашения.

Приведенные данные о производительности измерены в контролируемой среде. Таким образом, результаты, полученные в других операционных средах, могут существенно отличаться от них. Некоторые показатели измерены в системах разработки и нет никаких гарантий, что в общедоступных системах эти показатели будут теми же. Более того, некоторые результаты могут быть получены путем экстраполяции. Реальные результаты могут отличаться от них. Пользователи должны проверить данные для своих конкретных сред.

Информация о продуктах других фирм получена от поставщиков этих продуктов, из их опубликованных объявлений или из других общедоступных

источников. Фирма IBM не проверяла эти продукты и не может подтвердить точность измерений, совместимость или прочие утверждения о продуктах других фирм. Вопросы о возможностях продуктов других фирм следует направлять поставщикам этих продуктов.

Все утверждения о будущих планах и намерениях IBM могут быть изменены или отменены без уведомлений, и описывают исключительно цели фирмы.

Эта информация может содержать примеры данных и отчетов, иллюстрирующие типичные деловые операции. Чтобы эти примеры были правдоподобны, в них включены имена лиц, названия компаний и товаров. Все эти имена и названия вымышлены и любое их сходство с реальными именами и адресами полностью случайно.

ЛИЦЕНЗИЯ НА КОПИРОВАНИЕ:

Эта информация может содержать примеры прикладных программ на языках программирования, иллюстрирующих приемы программирования для различных операционных платформ. Разрешается копировать, изменять и распространять эти примеры программ в любой форме без оплаты фирме IBM для целей разработки, использования, сбыта или распространения прикладных программ, соответствующих интерфейсу прикладного программирования операционных платформ, для которых эти примера программ написаны. Эти примеры не были всесторонне проверены во всех возможных условиях. Поэтому IBM не может гарантировать их надежность, пригодность и функционирование.

Каждая копия программ примеров или программ, созданных на их основе, должна содержать следующее замечание об авторских правах:

© (название вашей фирмы) (год). Части этого кода построены на основе примеров программ IBM Corp. © Copyright IBM Corp. _введите год или годы_. Все права защищены.

Товарные знаки

Следующие термины (они могут быть помечены звездочкой - *) являются товарными знаками корпорации International Business Machines в Соединенных Штатах и/или в других странах:

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

Следующие термины являются товарными знаками или зарегистрированными товарными знаками других компаний:

Microsoft, Windows и Windows NT - товарные знаки или зарегистрированные товарные знаки Microsoft Corporation.

Java, все товарные знаки и логотипы на основе Java и Solaris - товарные знаки Sun Microsystems, Inc. в Соединенных Штатах и/или в других странах.

Tivoli и NetView - товарные знаки Tivoli Systems Inc. в Соединенных Штатах и/или других странах.

UNIX - зарегистрированный товарный знак в Соединенных Штатах и в других странах, его использование лицензируется исключительно фирмой X/Open Company Limited.

Названия других компаний, продуктов и услуг (они могут быть отмечены двойной звездочкой - **) могут быть товарными знаками или марками сервиса других фирм.

Индекс

© Copyright IBM Corp. 2001

U

W

X

A

Б

В

Г

Д

Ж

З

И

К

Л

М

Н

О

П

Р

C

T

У

Ф

Х

Ц

Ч

Э

Как связаться с IBM

Если у вас имеется техническая проблема, пожалуйста, перед обращением к службе поддержки пользователей DB2 просмотрите еще раз и выполните действия, рекомендуемые в руководстве *Troubleshooting Guide*. В этом руководстве описано, какую информацию надо собрать, чтобы служба поддержки пользователей DB2 могла лучше помочь вам.

Чтобы получить информацию или заказать любой из продуктов DB2 Universal Database, обратитесь к представителю IBM в местном отделении или к авторизованному продавцу программных продуктов IBM.

Если вы находитесь в США, позвоните по одному из следующих номеров:

- 1-800-237-5511, чтобы обратиться в службу поддержки
- 1-888-426-4343, чтобы узнать о доступных формах обслуживания.

Информация о продукте

Если вы находитесь в США, позвоните по одному из следующих номеров:

- 1-800-IBM-CALL (1-800-426-2255) или 1-800-3IBM-OS2 (1-800-342-6672), чтобы заказать продукты или получить общую информацию.
- 1-800-879-2755, чтобы заказать публикации.

http://www.ibm.com/software/data/
На страницах DB2 в WWW содержится текущая информация DB2: новости, описания продуктов, учебные планы и т.д.

http://www.ibm.com/software/data/db2/library/
DB2 Product and Service Technical Library содержит ответы на часто задаваемые вопросы, исправления, книги и свежую техническую информацию по DB2.

Примечание: Эта информация может быть только в английском варианте.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
На сайте заказов International Publications приводится информация о том, как заказывать книги.

http://www.ibm.com/education/certify/
На этом сайте представлена программа Professional Certification Program IBM и приводится информация о сертификационных испытаниях для многих продуктов IBM, в том числе DB2.

© Copyright IBM Corp. 2001

ftp.software.ibm.com

Зарегистрируйтесь как аноним. В каталоге `/ps/products/db2` можно найти демо-версии, исправления, информацию и инструменты для DB2 и многих других продуктов.

comp.databases.ibm-db2, bit.listserv.db2-l

В этих группах новостей пользователи обмениваются опытом работы с продуктами DB2.

В Compuserve: GO IBMDB2

Введите эту команду, чтобы попасть на форумы IBM DB2 Family. Через эти форумы поддерживаются все продукты DB2.

Информацию о том, как связаться с IBM из других стран, смотрите в Приложении A книги *IBM Software Support Handbook*. Этот документ можно найти в Web, обратившись по адресу: http://www.ibm.com/support/ и выбрав ссылку на IBM Software Support Handbook у нижнего края страницы.

Примечание: В некоторых странах авторизованные дилеры IBM должны обращаться не в центр поддержки IBM, а в структуры поддержки дилеров.

Напечатано в Дании

Spine information:

IBM® DB2® Universal Database

Справочное руководство по восстановлению данных и высокой доступности

Версия 7