IBM® DB2
Universal Database™

Справочное руководство по восстановлению данных и высокой доступности

Версия 8

SH43-0187-00

IBM® DB2
Universal Database™

Справочное руководство по восстановлению данных и высокой доступности

Версия 8

SH43-0187-00

Перед тем как использовать данный документ и продукт, описанный в нем, прочтите общие сведения под заголовком *Замечания*.

Заказать публикации IBM можно через Интернет или у местного представителя IBM.

- Чтобы заказать публикации через Интернет, перейдите на Web-страницу Центра публикаций IBM (IBM Publications Center): www.ibm.com/shop/publications/order
- Чтобы найти местное представительство IBM, перейдите на страницу IBM Directory of Worldwide Contacts по адресу www.ibm.com/planetwide

Чтобы заказать публикации DB2 через отдел DB2 Marketing and Sales в Соединенных Штатах или Канаде, позвоните по телефону 1-800-IBM-4YOU (426-4968).

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

Содержание

Об этой книге

В 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. В этом руководстве не даются инструкции по установке DB2, так как они зависят от вашей операционной системой.

Структура книги

В этом приложении 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.

Глава 1, “Разработка правильной стратегии резервного копирования и восстановления”
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 высокодоступного восстановления при отказах продуктом DB2 в системах Windows, поддерживающих 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 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 операций”
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, “Пример программы восстановления”
Содержит код примера программы, использующей API DB2 и вызовы встроенного SQL, связанные с восстановлением, а также информацию об использовании этих API и вызовов.

Приложение F, “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 G, “Обработчик пользователя для восстановления баз данных”
Обсуждаются возможности использования программ обработчиков пользователей с файлами журналов баз данных и описывается несколько программ примеров.

Приложение H, “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

Глава 1. Разработка правильной стратегии резервного копирования и восстановления

В этом разделе обсуждаются вопросы, которые необходимо учесть при выборе способов резервного копирования и восстановления базы данных и табличного пространства, а также восстановления с повтором транзакций.

В этом приложении рассматриваются следующие темы:

Разработка 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 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.

Файлы журналов восстановления и файл хронологии восстановления создаются автоматически при создании базы данных (рис. 1 на стр. 5). Эти файлы журналов применяются при восстановлении утерянных и поврежденных данных. Файл журнала восстановления и файл хронологии восстановления нельзя изменять вручную, однако вы можете удалить записи из файла хронологии восстановления с помощью команды PRUNE HISTORY. Кроме того, в параметре конфигурации базы данных *rec_his_retentn* можно указать число дней, в течение которых должен храниться файл хронологии восстановления.

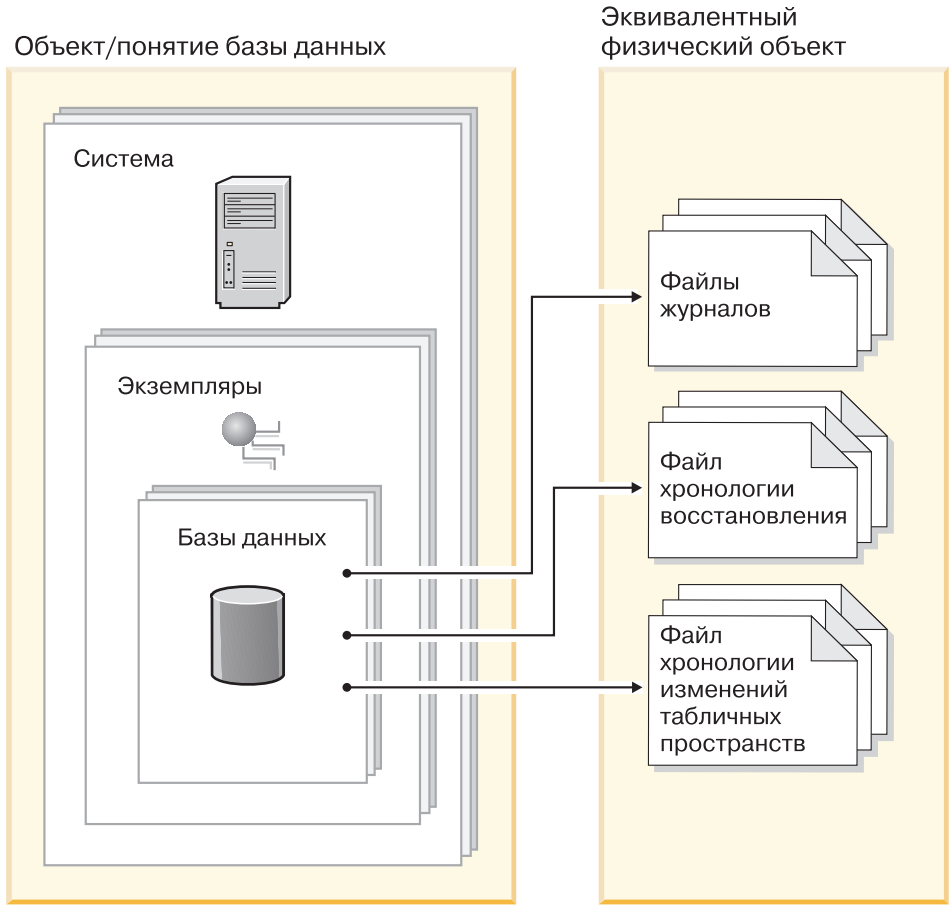


Рисунок 1. Файлы журнала восстановления и файл хронологии восстановления

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, с помощью которой можно определить опции восстановления, 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. Этот файл расположен в каталоге базы данных.

Файл хронологии изменения табличного пространства содержит информацию, с помощью которой можно определить, какие файлы журнала необходимы для восстановления заданного табличного пространства. Этот файл также расположен в каталоге базы данных.

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 резервной копии. Восстановление путем повтора транзакций для такой базы данных не поддерживается.

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.

Понятия, связанные с данным:

- “Восстановление после отказа” на стр. 11
- “Восстановление версии” на стр. 27
- “Восстановление с повтором транзакций” на стр. 28
- “Data Links server file backups” в *Справка: Центр управления*
- “Failure and recovery overview” в *DB2 Data Links Manager Administration Guide and Reference*

Ссылки, связанные с данной темой:

- “Recovery History Retention Period configuration parameter - rec_his_retentn” в *Руководство администратора: Производительность*
- “DB2 Data Links Manager system setup and backup recommendations” в *DB2 Data Links Manager Administration Guide and Reference*

Как часто выполнять резервное копирование

План восстановления должен предполагать регулярное создание резервных копий, поскольку резервное копирование базы данных требует времени и системных ресурсов. В плане можно предусмотреть как операции создания полной резервной копии, так и операции создания инкрементных резервных копий.

Следует регулярно создавать полные резервные копии базы данных, даже если вы архивируете журналы (нужные для восстановления с повтором транзакций). Восстановление базы данных из набора образов резервных копий табличных пространств занимает больше времени, чем аналогичное восстановление 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. Во время автономного резервного копирования другие программы *не могут* подключаться к базе 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, так как не возникают конфликты при доступе к файлам данных.

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. В этом случае при сбое диска с большой вероятностью будет затронуто только одно табличное пространство. Восстановление или повтор транзакций для одного из этих табличных пространств займет меньше времени, чем восстановление табличного пространства, хранящего все данные.

Для экономии времени резервные копии различных табличных пространств можно создавать в разное время, так как они изменяются с разной частотой. Например, если длинные поля и данные LOB изменяются реже, чем остальные данные, то для соответствующих табличных пространств можно реже создавать резервную копию. Если длинные поля и данные LOB не требуются для восстановления, то резервную копию соответствующего табличного пространства можно не создавать. Если данные больших объектов воспроизводятся из отдельного источника, при создании или изменении таблицы, содержащей столбцы больших объектов, выберите опцию NOT LOGGED.

Примечание: Если данные длинных полей, данные LOB и индексы хранятся в разных табличных пространствах и не копируются одновременно, то учтите следующее: Если будет создана резервная копия табличного пространства, содержащего не все табличные данные, то вы не сможете выполнить восстановление с повтором транзакций до указанного момента времени для этого табличного пространства. 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 составлении плана восстановления следует учесть эти затраты и их влияние на ваши деловые операции. Проверка полного плана восстановления поможет определить, отвечает ли время, необходимое на восстановление базы данных, вашим требованиям. После каждой проверки может потребоваться увеличить частоту создания резервных копий. Если в состав стратегии входит восстановление с повтором транзакций, это уменьшит число журналов, архивируемых между резервными копированиями и, как результат, сократит время, затрачиваемое на повтор транзакций базы данных после операции восстановления.

Понятия, связанные с данным:

- “Инкрементное резервное копирование и восстановление” на стр. 31

Ссылки, связанные с данной темой:

- Приложение G, “Обработчик пользователя для восстановления баз данных” на стр. 345
- “Параметры конфигурации для записи в журнал базы данных” на стр. 44

Требования к памяти

После выбора метода восстановления следует рассмотреть требуемый объем памяти.

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

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

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.

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:

- Воспользоваться обработчиком пользователя, чтобы скопировать эти журналы на другое устройство хранения в вашей среде.
- Вручную копировать журналы на устройство хранения или в каталог, отличный от каталога пути журналов базы данных, после того как они перестают быть активными.

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 операционными системами схожей архитектуры, например, AIX® и Sun Solaris, и между 32-разрядными и 64-разрядными операционными системами возможны операции межплатформенного копирования и восстановления. Передача резервной копии между системами должна происходить в двоичном режиме. Версия DB2®, установленная в системе назначения, должна быть не младше той версии, которая установлена в исходной системе. Восстановление системы младшей версии не поддерживается.

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.

Ссылки, связанные с данной темой:

- “db2move - Database Movement Tool Command” в *Command Reference*
- “EXPORT Command” в *Command Reference*
- “IMPORT Command” в *Command Reference*
- “LOAD Command” в *Command 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 на стр. 12

на стр. 12). Когда база данных находится в согласованном и пригодном к использованию состоянии, говорят, что она достигла так называемой ″точки согласованности″.

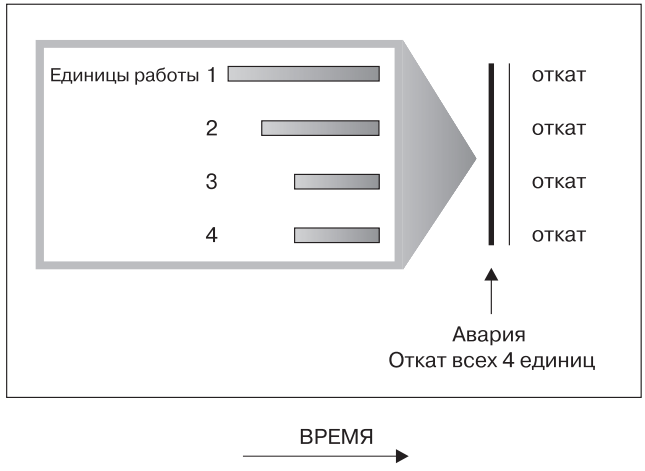


Рисунок 2. Откат единиц работы (восстановление после отказа)

Ошибка транзакции происходит в результате серьезной ошибки или условия, 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 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®.
- Сбой аппаратного обеспечения, например, повреждение памяти, либо сбой диска, CPU или сети.

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 применяется по умолчанию.) Если вы не хотите использовать автоматический перезапуск, присвойте параметру конфигурации базы данных *autorestart* значение `OFF`. В этом случае после сбоя базы данных вам потребуется вручную вызывать команду RESTART DATABASE. Если перед возникновением сбоя в базе данных было приостановлено выполнение операций ввода-вывода, то для продолжения восстановления после сбоя в команде RESTART DATABASE нужно указать опцию WRITE RESUME. Когда операция

перезапуска базы данных начнет выполняться, в журнал оповещений администратора будет занесена соответствующая запись.

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 будут доступны, и с базой данных могут быть установлены соединения. Однако если в автономный режим было переведено табличное пространство, содержащее системные каталоги, то соединения с базой данных можно будет установить только после восстановления этого табличного пространства.

Ссылки, связанные с данной темой:

- "Auto Restart Enable configuration parameter - autorestart" в *Руководство администратора: Производительность*

Восстановление после сбоя - Подробная информация

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

Если необходимо выполнить восстановление после сбоя, то поврежденное табличное пространство будет переведено в автономный режим и не будет доступно. Оно будет находиться в состоянии отложенного повтора транзакций. Если других ошибок нет, то операция перезапуска будет выполнена, а поврежденное табличное пространство станет доступно, если вы:

- Исправите поврежденные контейнеры, не потеряв исходные данные, а затем выполните для табличного пространства повтор транзакций до конца журналов. (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 выполните повтор транзакций до конца журналов или до определенного момента времени.

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 журнала

уведомлений администратора 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.

Уменьшение воздействия ошибок носителя

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

- Создайте зеркальные копии или дубликаты дисков, где хранятся данные и журналы для важных баз данных.
- Используйте конфигурацию RAID (Redundant Array of Independent Disks - дублированный массив независимых дисков), например, RAID уровня 5.
- 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 для этого используется *массив дисков*, представляющий собой набор дисков.

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 только RAID 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. Производительность 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; это обеспечит возможность восстановления до момента возникновения 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.

Более подробную информацию об уровнях RAID можно найти на следующем Web-сайте: http://www.acnc.com/04_01_00.html

Программные массивы дисков: Программный массив дисков выполняет те же функции, что и аппаратный, 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.

Стандартный программный массив дисков поддерживает зеркальную копию дисков. 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 запретить доступ к диску. Загрузочный диск нужно отделить от массива дисков.

Уменьшение воздействия ошибок транзакций

Для уменьшения воздействия ошибок транзакций попытайтесь гарантировать:

- Непрерывное электропитание
- Достаточное дисковое пространство для журналов базы данных
- Надежные каналы связи по серверам разделов базы данных в среде многораздельных баз данных
- Синхронизацию системного времени в среде многораздельной базы данных.

Понятия, связанные с данным:

- “Синхронизация системного времени в системе многораздельной базы данных” на стр. 144

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.

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).
- Если прикладная программа была (до возникновения ошибки) связана с сервером раздела базы данных, где произошла ошибка, но ни локальный сервер раздела базы данных, ни сервер раздела базы данных, где произошла ошибка, 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 аварийному завершению работы менеджера баз данных, вы можете перезапустить менеджер баз данных после перезапуска раздела базы данных с помощью команды **db2start** с опцией RESTART. Если вам не удалось перезапустить раздел базы данных, то с помощью команды **db2start** менеджер баз данных можно перезапустить в другом разделе базы данных.

Если работа менеджера баз данных завершается ненормально, разделы базы данных на этом сервере могут остаться в несогласованном состоянии. Чтобы они стали 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 существует 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.

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).

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

журнал уведомлений администратора 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.

Понятия, связанные с данным:

- “Двухфазное принятие (Two-phase commit)” в *Руководство администратора: Планирование*
- “Восстановление после ошибок при двухфазном принятии” в *Руководство администратора: Планирование*

Задачи, связанные с данной темой:

- “Разрешение неоднозначных транзакций вручную” в *Руководство администратора: Планирование*

Ссылки, связанные с данной темой:

- “db2start - Start DB2 Command” в *Command Reference*
- “LIST INDOUBT TRANSACTIONS Command” в *Command Reference*

Восстановление после ошибки на сервере раздела базы данных

Процедура:

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.

Понятия, связанные с данным:

- “Восстановление при ошибках транзакций в среде многораздельных баз данных” на стр. 18

Ссылки, связанные с данной темой:

- “db2start - Start DB2 Command” в *Command Reference*
- “RESTART DATABASE Command” в *Command Reference*

Восстановление неоднозначных транзакций на хосте, когда на DB2 Connect настроен менеджер точек синхронизации

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.

Процедуры:

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 в 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 сервер 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.

```
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.

Задачи, связанные с данной темой:

- “Восстановление неоднозначных транзакций на хосте, когда DB2 Connect не применяет менеджер точек синхронизации DB2” на стр. 24

Ссылки, связанные с данной темой:

- “db2start - Start DB2 Command” в *Command Reference*
- “LIST INDOUBT TRANSACTIONS Command” в *Command Reference*
- “RESTART DATABASE Command” в *Command Reference*

Восстановление неоднозначных транзакций на хосте, когда DB2 Connect не применяет менеджер точек синхронизации DB2

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.

Процедура:

В этом разделе описана процедура обновления DB2 for OS/390 по соединению TCP/IP при обновлении нескольких узлов в DB2 Connect Personal Edition или DB2 Connect Enterprise Server Edition, когда менеджер точек синхронизации не применяется. 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`.

Повторная синхронизация любой неоднозначной транзакции происходит автоматически, как только TM, а также принимающие участие в транзакции базы данных и их соединения становятся снова доступными. 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*.

Задачи, связанные с данной темой:

- “Восстановление неоднозначных транзакций на хосте, когда на DB2 Connect настроен менеджер точек синхронизации” на стр. 23

Ссылки, связанные с данной темой:

- “LIST INDOUBT TRANSACTIONS Command” в *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:

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

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

В любом плане аварийного восстановления могут играть роль как резервные копии табличных пространств, так и полные резервные копии базы данных. План аварийного восстановления может базироваться на функциях резервного копирования, восстановления и повтора транзакций DB2®. Для защиты результатов вашей работы следует убедиться в наличии проверенных процедур восстановления.

Понятия, связанные с данным:

- “Переопределение контейнеров табличных пространств при операции восстановления (перенаправленное восстановление)” на стр. 103

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).

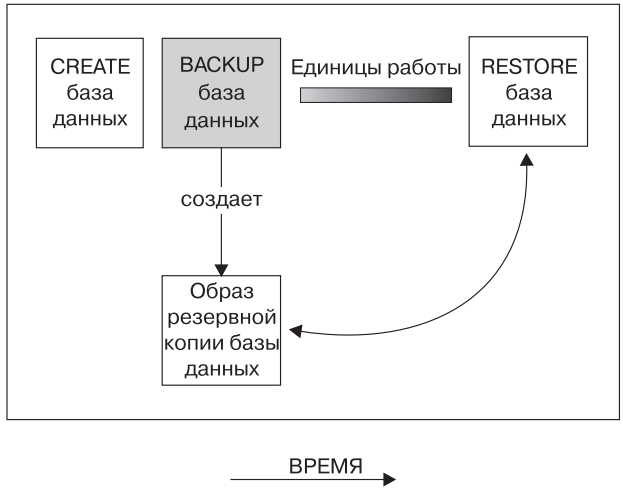


Рисунок 3. Восстановление версии. База данных восстанавливается из последней резервной копии, однако все единицы работы между моментом создания этой копии и моментом отказа теряются.

Для восстановления версии необходимо запланировать и регулярно создавать полные резервные копии 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 путем повтора транзакций* необходима резервная копия базы данных и архивированные журналы (для этого нужно задать параметры конфигурации базы данных *logretain* и (или) *userexit*). 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.

Примечание: Опцию WITHOUT ROLLING FORWARD нельзя использовать, если резервная копия базы данных была создана в оперативном режиме.

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 на стр. 29). 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.

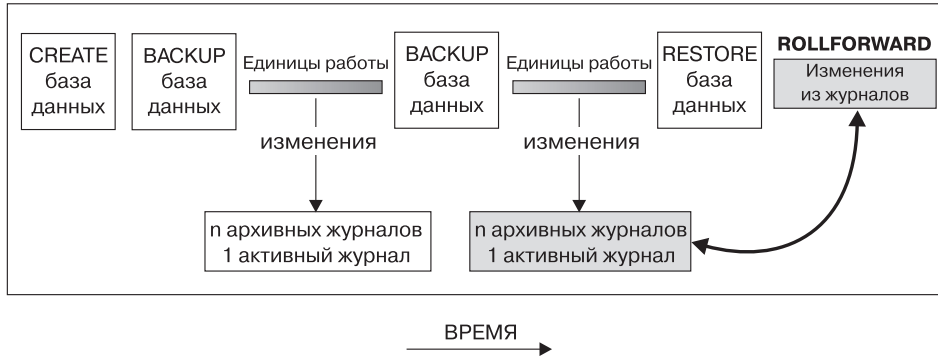


Рисунок 4. Восстановление базы данных с повтором транзакций. Для длительных транзакций может существовать несколько активных журналов.

- *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. 30). 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. Вызовите команду ROLLFORWARD DATABASE, чтобы выполнить повтор транзакций для табличных пространств с использованием журналов либо до указанного момента времени, либо до конца журналов.
- 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.

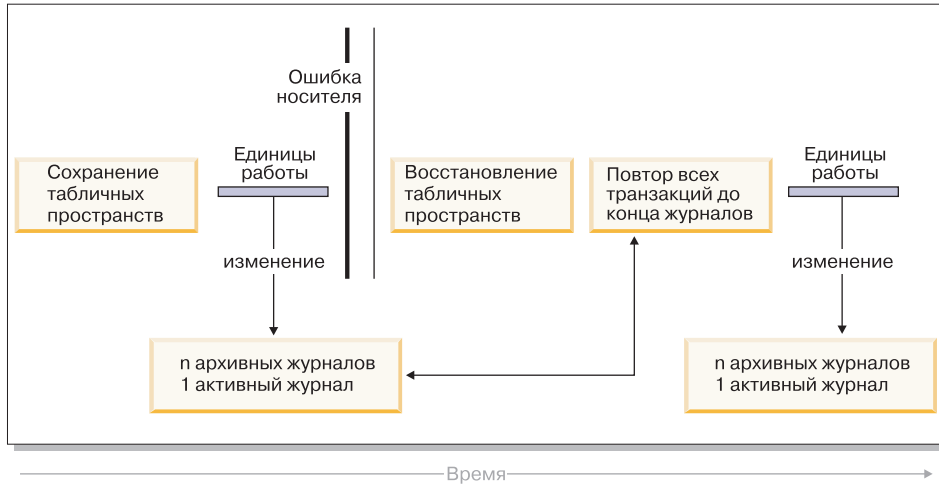


Рисунок 5. Восстановление с повтором транзакций табличного пространства. Для длительных транзакций может существовать несколько активных журналов.

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 запрос на повтор 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.

Понятия, связанные с данным:

- “Что такое журналы восстановления” на стр. 38

Ссылки, связанные с данной темой:

- “ROLLFORWARD DATABASE” на стр. 146

Примеры, связанные с данной темой:

- “dbrecov.out -- HOW TO RECOVER A DATABASE (C)”
- “dbrecov.sqc -- How to recover a database (C)”
- “dbrecov.out -- HOW TO RECOVER A DATABASE (C++)”
- “dbrecov.sqC -- How to recover a database (C++)”

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 течение какого-то времени. Каждая следующая инкрементная копия включает все содержимое предыдущей инкрементной копии плюс все новые или изменившиеся с момента создания предыдущей полной копии данные. Разностная резервная копия содержит только те страницы, которые изменились с момента создания предыдущей копии любого типа.

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.

Разрешить отслеживание изменений базы данных можно с помощью нового параметра конфигурации базы данных DB2, *trackmod*. Для этого параметра допустимы два значения:

- NO. Инкрементное резервное копирование в этой конфигурации не разрешено. Изменения страниц баз данных не отслеживаются и никак не регистрируются. Это значение применяется по умолчанию.
- YES. Инкрементное резервное копирование в этой конфигурации разрешено. Когда разрешается отслеживание изменений, новая конфигурация вступает в силу при первом успешном соединении с базой данных. Перед созданием

инкрементной резервной копии отдельного табличного пространства необходимо создать полную резервную копию этого табличного пространства.

Для табличных пространств SMS и DMS отслеживание изменений выполняется на уровне табличного пространства. На этом уровне для каждого табличного пространства предусмотрен флаг, указывающий, содержит ли табличное пространство страницы, для которых нужно создать резервную копию. Если ни для одной страницы табличного пространства копию создавать не нужно, то из операции резервного копирования можно исключить все табличное пространство.

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

Задачи, связанные с данной темой:

- “Восстановление из инкрементных резервных копий” на стр. 33

Инкрементное резервное копирование и восстановление - Подробная информация

Восстановление из инкрементных резервных копий

Процедура:

Операция восстановления из инкрементных резервных копий всегда состоит из следующих шагов:

1. Определение инкрементной копии назначения.

   Определите последнюю копию, подлежащую восстановлению, и запросите операцию инкрементного восстановления в утилите восстановления DB2. Этот образ при инкрементном восстановлении называется целевым образом, поскольку он будет последним из восстановленных образов. 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.

Существует два способа восстановить инкрементные резервные копии.

- Инкрементное восстановления можно выполнить вручную. Для этого для каждой резервной копии, которую требуется восстановить (как было определено ранее), нужно вызвать команду RESTORE.
- Для автоматического инкрементного восстановления команду RESTORE нужно вызвать один раз, указав образ назначения. DB2 с помощью хронологии базы данных определит остальные образы резервных копий и восстановит их.

Пример инкрементного восстановления вручную

Для того чтобы восстановить набор инкрементных резервных копий вручную, укажите образ резервной копии назначения в параметре TAKEN AT *отметка_времени* команды RESTORE DATABASE и выполните описанные выше действия. Например:

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

   где:
     <отметка> указывает на последнюю инкрементную
     резервную копию (резервную копию назначения),
      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 выполняется инкрементное восстановление базы данных, и существуют резервные образы табличного пространства, то эти образы должны восстанавливаться в хронологическом порядке в соответствии с их отметками времени резервного копирования.

Если вы планируете вручную выполнить инкрементное восстановление, то с помощью утилиты **db2ckrst** можно получить хронологию базы данных и создать список отметок времени резервных образов, необходимых для инкрементного восстановления. Кроме того, будет создан упрощенный синтаксис восстановления для инкрементного восстановления вручную. Рекомендуется хранить полную информацию об операциях резервного копирования и использовать эту утилиту только в качестве руководства.

Пример автоматического инкрементного восстановления

Для того чтобы автоматически восстановить набор инкрементных резервных копий, задайте опцию TAKEN AT в команде RESTORE DATABASE. Укажите в ней отметку времени последней резервной копии, которую нужно восстановить. Например:

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

В результате утилита восстановления DB2 автоматически выполнит действия, описанные в начале раздела. 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 этом случае автоматическое инкрементное восстановление выполнить нельзя, и вам потребуется вызвать команду RESTORE DATABASE с опцией INCREMENTAL ABORT. В результате будут очищены оставшиеся ресурсы, для того чтобы можно было выполнить инкрементное восстановление вручную.

Примечание: Настоятельно рекомендуется не применять 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.

На третьей фазе обработки DB2 восстановит все оставшиеся резервные копии в созданной цепочке. Если на этой фазе произойдет ошибка, вам потребуется

вызвать команду RESTORE DATABASE с опцией INCREMENTAL ABORT для очистки остальных ресурсов. После этого нужно определить, можно ли исправить ошибку перед повторным вызовом команды RESTORE или повторной попыткой выполнить инкрементное восстановление вручную.

Понятия, связанные с данным:

- “Инкрементное резервное копирование и восстановление” на стр. 31

Ссылки, связанные с данной темой:

- “RESTORE DATABASE” на стр. 105
- “db2ckrst - Проверить последовательность инкрементного образа восстановления” на стр. 234

Ограничения автоматического инкрементного восстановления

1. Если табличное пространство было переименовано после создания резервной копии, которую нужно восстановить, и в операции восстановления на уровне таблиц было задано новое имя, то на основании хронологии базы данных не будет создана правильная цепочка резервных копий и возникнет ошибка (SQL2571N).

   Например:

   ```
   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>

   SQL2571N  Невозможно выполнить автоматическое инкрементное восстановление.
   Код причины: "3".
   ```

   Рекомендуемое исправление: Выполните инкрементное восстановление вручную.

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 затем будет выполнено автоматическое инкрементное восстановление, для которого потребуются утерянные записи хронологии, то утилита RESTORE создаст неправильную цепочку резервных копий и вернет ошибку ″out of sequence″ (SQL2572N).

   Например:

   ```
   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>
   ```

Рекомендуемое исправление:

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

3. Если вы восстановили образ резервной копии одной базы данных в другой базе данных, а затем создали инкрементную (разностную) резервную копию, то для данной резервной копии нельзя будет выполнить автоматическое инкрементное восстановление.

   Например:

```
db2 create db a
db2 create db b

db2 update db cfg for a using trackmod on

db2 backup db a -> ts1
db2 restore db a taken at ts1 into b

db2 backup db b incremental -> ts2

db2 restore db b incremental automatic taken at ts2

SQL2542N Для заданных алиаса базы данных "B" и временной отметки "ts1"
соответствующий файл резервной копии не найден.
```

   Рекомендуемое исправление:

   - Выполните инкрементное восстановление вручную, как указано ниже:

```
db2 restore db b incremental taken at ts2
db2 restore db a incremental taken at ts1 into b
db2 restore db b incremental taken at ts2
```

   - После того как вы выполните операцию восстановления в базе данных B, создайте полную резервную копию базы данных для начала новой цепочки инкрементных копий

Понятия, связанные с данным:

• “Инкрементное резервное копирование и восстановление” на стр. 31

Задачи, связанные с данной темой:

• “Восстановление из инкрементных резервных копий” на стр. 33

Ссылки, связанные с данной темой:

• “RESTORE DATABASE” на стр. 105

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 два типа регистрации в журналах: *циклическая* и *архивная*. Каждая из них обеспечивает свой уровень возможностей восстановления:

- *Циклическая* запись выбирается по умолчанию, когда создается новая база данных. (Параметрам конфигурации базы данных *logretain* и *userexit присвоено значение* `NO`.) При этом типе регистрации в журнале допустимы только полные автономные резервные копии базы данных. При выполнении резервного копирования база данных должна находиться в автономном состоянии (недоступном для пользователей). 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. Так как этот тип восстановления восстанавливает данные до конкретной точки времени создания полной резервной копии, он называется *восстановлением версии*.

  На рис. 6 на стр. 39 показано, что когда активна циклическая запись, активный журнал использует файлы журналов в циклическом порядке.

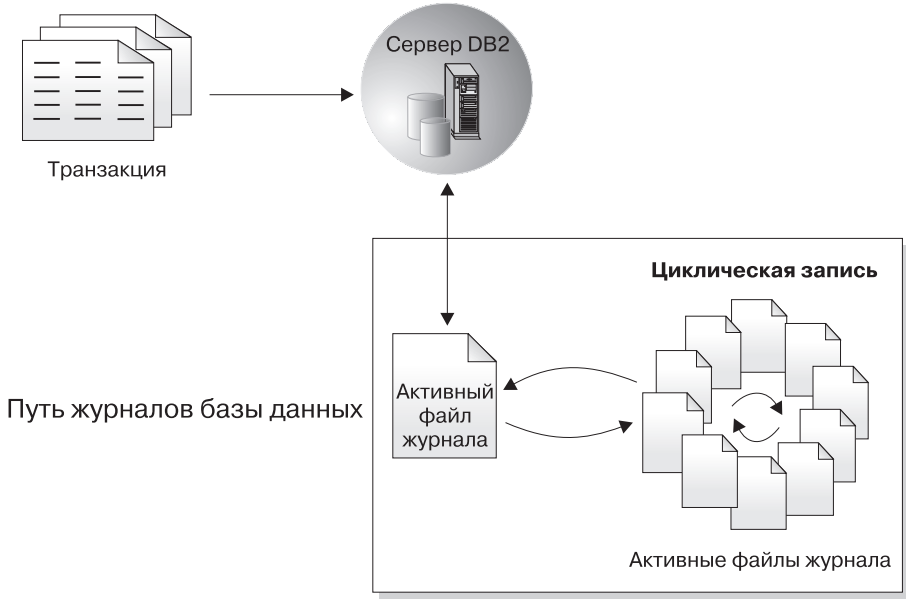


Рисунок 6. Циклическая запись журналов

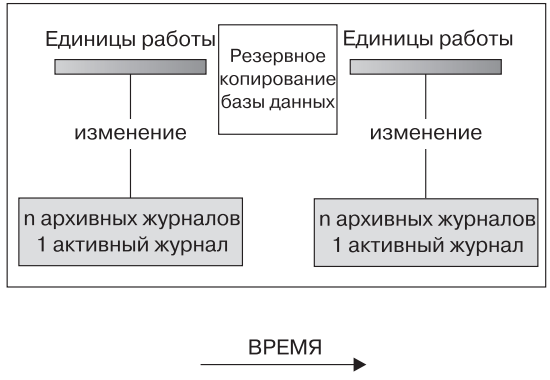
Активные журналы используются во время восстановления после сбоя, чтобы не допустить ситуации, когда база данных после сбоя останется в несогласованном состоянии. Команда RESTART DATABASE использует активные журналы, когда базу данных нужно перевести в согласованное и пригодное к использованию состояние. 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.

- *Архивная* регистрация используется специально для восстановления с повтором транзакций. Такая регистрация применяется в том случае, если включен хотя бы один из параметров конфигурации базы данных *logretain* и *userexit*. Вы можете настроить DB2 таким образом, чтобы файлы архивных журналов сохранялись в активном каталоге, и вы архивировали их вручную, либо установить программу обработчика для автоматического архивирования этих файлов. Архивными называются те журналы, которые были активны, однако больше не нужны для восстановления после сбоя.

Преимуществом архивной регистрации является то, что при восстановлении базы данных с повтором транзакций до конца журналов или до заданного момента времени могут применяться как архивные, так и активные журналы. Файлы архивных журналов могут применяться для восстановления изменений, внесенных после создания резервной копии. В этом состоит отличие архивной

регистрации от циклической регистрации, при которой восстановление можно выполнить только до момента создания резервной копии, а все изменения, внесенные после этого момента, теряются.

Резервное копирование без отключения допустимо лишь в том случае, если для базы данных настроена архивная регистрация. 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 базах данных используются журналы.

Рисунок 7. 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.

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 *Руководство администратора: Производительность*.

Понятия, связанные с данным:

- “Создание зеркальной копии журнала” на стр. 41

Ссылки, связанные с данной темой:

- Приложение G, “Обработчик пользователя для восстановления баз данных” на стр. 345
- “First Active Log File configuration parameter - loghead” в *Руководство администратора: Производительность*

Подробная информация о журнале восстановления

Создание зеркальной копии журнала

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 активные журналы могут быть повреждены (в результате сбоя диска), то с помощью нового параметра конфигурации DB2, MIRRORLOGPATH, можно задать путь к копии активного журнала, для того чтобы база данных создала зеркальную копию томов журнала.

Параметр конфигурации MIRRORLOGPATH позволяет базе данных сохранить вторую копию файлов журнала в другом расположении. Рекомендуется разместить такую копию на другом физическом диске (желательно с другим контроллером диска). Тогда контроллер диска не будет уязвимой точкой.

После первого включения параметра MIRRORLOGPATH он начинает применяться не сразу же, а после следующего запуска базы данных. В этом смысле он аналогичен параметру конфигурации NEWLOGPATH.

Если при записи в путь активного журнала или путь зеркальной копии журнала возникает ошибка, база данных помечает этот путь как “плохой”, заносит сообщение в журнал уведомлений администратора и продолжает помещать последующие записи журнала 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 ошибка произойдет при записи данных в оставшийся "хороший" путь, то работа базы данных будет завершена.

Ссылки, связанные с данной темой:

- "Mirror Log Path configuration parameter - mirrorlogpath" в *Руководство администратора: Производительность*

Сокращение объема регистрируемой в журнале информации с помощью параметра NOT LOGGED INITIALLY

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

Примечания:

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

Поскольку при использовании атрибута NOT LOGGED INITIALLY в журнале не регистрируются изменения, внесенные в таблицу, следует учесть следующее:

- *Все* изменения, внесенные в таблицу, будут сохраняться на диске во время принятия. Это означает, что принятие может занять больше времени.
- Если задан атрибут NOT LOGGED INITIALLY, и была выполнена незарегистрированная операция, то в случае ошибки при выполнении оператора или вызова операции ROLLBACK TO SAVEPOINT (SQL1476N) будет выполнен откат всей единицы работы.
- При повторе транзакций такую таблицу восстановить невозможно. Если при повторе транзакций будет обнаружена таблица, при создании или изменении которой был задан атрибут NOT LOGGED INITIALLY, то таблица будет

помечена 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.

Сокращение объема регистрируемой в журнале информации для объявленных временных таблиц

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

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

Понятия, связанные с данным:

- “Application processes, concurrency, and recovery” в *SQL Reference, Том 1*

Задачи, связанные с данной темой:

- “Создание табличного пространства” в *Руководство администратора: Реализация*

Ссылки, связанные с данной темой:

- “DECLARE GLOBAL TEMPORARY TABLE statement” в *SQL Reference, Том 2*

Параметры конфигурации для записи в журнал базы данных

Первичные файлы журнала (logprimary)

Этот параметр задает число создаваемых первичных файлов журнала, размер которых равен *logfilsz*.

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. Общее ограничение на размер файлов журнала в пространстве активного журнала составляет 256 Гб.

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.

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 этот параметр равен -1, то ограничение на размер пространства активного журнала базы данных не устанавливается. Размер и число начатых транзакций, выполняемых в базе данных, не ограничены.

Примечания:

1. Параметру *logsecond* можно присвоить значение -1 только в том случае, если включен параметр конфигурации базы данных *userexit*.
2. Если этот параметр равен -1, то время восстановления после сбоя может возрасти, поскольку DB2 может потребоваться восстановить файлы архивных журналов.

Размер файла журнала (logfilsiz)

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

Для общего размера пространства активного журнала установлено логическое ограничение в 256 Гб. Такое ограничение связано с тем, что максимальное значение *logfilsiz* составляет 262144, а максимальное значение (*logprimary* + *logsecond*) составляет 256.

Размер файлов журнала непосредственно влияет на производительность. Переключение с одного журнала на другой приводит к определенному снижению производительности. Следовательно, с точки зрения производительности чем больше размер файла журнала, тем лучше. Этот параметр также задает размер файла журнала для архивирования. С этой точки зрения слишком большой размер файла журнала повышает вероятность возникновения сбоя или задержки в сценариях распределения журналов. Если рассматривать пространство активного журнала, то иногда выгоднее создать несколько более мелких файлов журнала. Например, если существует два очень больших файла журнала, и к моменту начала транзакции один из файлов журнала был почти заполнен, то половина пространства журнала останется доступной.

При деактивации базы данных (завершении всех соединений с базой данных) усекается тот файл журнала, в который в данным момент записываются данные. Следовательно, если база данных часто деактивируется, то лучше не устанавливать большой размер файла журнала, поскольку практически сразу после создания большого файла он будет усечен. Для того чтобы избежать этих издержек, можно вызвать команду ACTIVATE DATABASE. Производительность будет выше и в том случае, если пул буферов будет заполнен.

Предположим, что у вас есть прикладная программа, поддерживающая базу данных в открытом состоянии для сокращения расходов процессорного времени на ее открытие; тогда размер файла журнала должен определяться количеством времени, необходимого для автономного создания копий архивированных файлов 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 задает размер буфера, в который помещаются записи журнала перед их сохранением на диске. 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. Для того чтобы изменить расположение активных журналов и будущих архивных журналов, укажите в этом параметре конфигурации другой каталог или устройство. Если база данных настроена для восстановления с повтором транзакций, то активные журналы, хранящиеся в исходном каталоге, не перемещаются в новое расположение.

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 *database_consistent*.

Путь зеркальной копии журнала (mirrorlogpath)

Для того чтобы обеспечить защиту журналов от сбоя диска и случайного удаления, можно указать путь зеркальной копии, в котором

будет храниться идентичный набор журналов. Для этого укажите в этом параметре конфигурации другой каталог. Активные журналы, хранящиеся в каталоге, для которого создается зеркальная копия, не перемещаются в новое расположение, если база данных настроена для восстановления с повтором транзакций.

Поскольку расположение файлов журнала можно изменять, те файлы, которые необходимы для восстановления с повтором транзакций, могут находиться в разных каталогах. 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 *database_consistent*.

Для выключения этого параметра конфигурации присвойте ему значение DEFAULT.

Примечания:

1. Этот параметр конфигурации не поддерживается, если в качестве основного пути журнала задано устройство с линейным доступом.
2. В качестве значения этого параметра нельзя указать устройство с линейным доступом.

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.

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.

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 скорость передачи меньше скорости генерирования новых данных журнала, то при достаточно продолжительном времени записи в журнал свободное место на диске может закончиться. Время записи в журнал, по истечении которого

закончится место на диске, зависит от объема свободной памяти на диске. В этом случае обработка базы данных останавливается.

Особенно критична скорость передачи данных при использовании магнитной ленты или оптических носителей. 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.

Примечание: Для того чтобы размер пространства активного журнала не был ограничен, этому параметру необходимо присвоить значение `ON`.

Примечание: По умолчанию параметры конфигурации базы данных *logretain* и *userexit* настроены таким образом, что восстановление путем повтора транзакций не поддерживается. Если вы планируете использовать эту функцию, измените эти параметры.

Альтернативный путь журналов (overflowlogpath)

Этот параметр может применяться в нескольких целях, в зависимости от требований, предъявляемых к ведению журнала. Вы можете указать расположение, применяемое DB2 для поиска файлов журнала, необходимых для повтора транзакций. Такое расположение аналогично значению опции OVERFLOW LOG PATH команды ROLLFORWARD. Вместо того чтобы задавать эту опцию в каждой команде ROLLFORWARD, можно один раз настроить этот параметр конфигурации. Если задан и параметр конфигурации *overflowlogpath*, и опция OVERFLOW LOG PATH, то значение опции переопределит значение параметра для одной операции повтора транзакций.

Если параметр *logsecond* равен -1, то вы можете указать каталог, в котором DB2 будет хранить файлы активного журнала, восстановленные из архива. (Файлы архивного журнала восстанавливаются для выполнения операций отката, если они уже не хранятся в пути активного журнала).

Если параметр *overflowlogpath* не задан, DB2 будет восстанавливать файлы журнала в пути активного журнала. Этот параметр позволяет задать дополнительный ресурс, в котором DB2 может хранить восстановленные файлы журнала. Это позволяет распределить операции

ввода-вывода по нескольким дискам и освободить место для дополнительных файлов журнала в пути активного журнала.

Например, если для копирования применяется API **db2ReadLog**, то с помощью параметра *overflowlogpath* можно задать расположение, в котором DB2 будет выполнять поиск файлов журнала, необходимых для этого API. Если файл журнала не будет найден ни в пути активного журнала, ни в альтернативном пути журнала, и для базы данных включен параметр конфигурации *userexit*, то DB2 восстановит файл журнала. Кроме того, с помощью этого параметра можно задать каталог, в котором DB2 будет хранить восстановленные файлы журнала. Это дает возможность сократить число операций ввода-вывода, выполняемых для пути активного журнала, и освободить место для дополнительных файлов журнала в пути активного журнала.

Если в качестве пути активного журнала задано устройство с линейным доступом, и вы хотите присвоить параметру *logsecond* значение -1, либо планируете использовать API **db2ReadLog**, то необходимо настроить параметр *overflowlogpath*.

В качестве значения *overflowlogpath* можно задать строку размером не более 242 байт. Эта строка должна представлять собой полное, а не относительное имя. Можно указать только имя каталога, но не имя устройства с линейным доступом.

Примечание: В среде многораздельных баз данных к пути автоматически добавляется номер узла. Это делается для сохранения единого пути в нескольких конфигурациях логических узлов.

Блокировать сообщения о переполнении диска журнала (blk_log_dsk_ful)
Этот параметр конфигурации 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 создать файл журнала каждые 5 минут, пока это не удастся сделать. После каждой попытки DB2 будет заносить сообщение в журнал уведомлений администратора. Единственный способ убедиться в том, что программа зависает из-за переполнения диска журнала - отслеживать сообщения, заносимые в журнал уведомлений администратора. Пока файл журнала не будет создан успешно, ни одна пользовательская программа, пытающаяся изменить табличные данные, не сможет осуществить принятие транзакций. При этом запросы на чтение данных не будут затронуты напрямую, однако если запрос попытается обратиться к данным, которые заблокированы запросом на обновление, или странице данных, фиксированной в пуле буферов программой, выполняющей обновление, то такой запрос тоже зависнет.

Если параметру *blk_log_dsk_ful* будет присвоено значение YES, и DB2 обнаружит, что диск журнала переполнен, то прикладная программа может зависнуть. После исправления этой ошибки выполнение транзакции будет продолжено. Для того чтобы разгрузить диск можно переместить старые файлы журналов в другую файловую систему или увеличить размер текущей файловой системы. Это позволит завершить выполнение зависшей программы.

Если параметру blk_log_dsk_ful присвоено значение NO, то в случае переполнения диска журнала в транзакции возникает ошибка, и выполняется откат транзакции. В некоторых случаях при переполнении диска журнала во время выполнения транзакции работа базы данных может быть завершена.

Понятия, связанные с данным:

- “Управление файлами журнала” на стр. 50
- “Повышение производительности восстановления” на стр. 68

Ссылки, связанные с данной темой:

- Приложение G, “Обработчик пользователя для восстановления баз данных” на стр. 345
- “Number of Secondary Log Files configuration parameter - logsecond” в *Руководство администратора: Производительность*
- “Change the Database Log Path configuration parameter - newlogpath” в *Руководство администратора: Производительность*

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 расположение.)

- 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. 8 на стр. 52). 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. Обычно такая ситуация распознается во время восстановления, однако если восстановление базы данных выполняется сразу после восстановления табличного пространства, то утилита восстановления не сможет распознать резервную копию табличного пространства в старой последовательности. 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 этого табличного пространства завершиться неудачно, и табличное пространство останется в состоянии отложенного восстановления.

Предположим, например, что операция снятия резервной копии табличного пространства, Backup 3, завершена между `S0000013.LOG` и `S0000014.LOG` верхней последовательности журналов (смотрите рис. 8). 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.

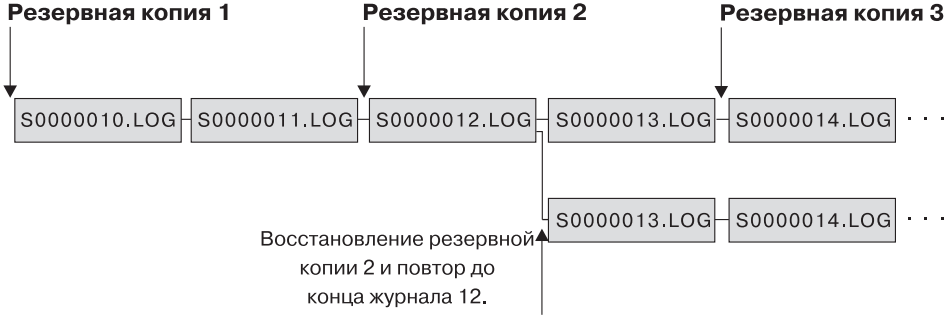


Рисунок 8. Повторное использование имен файлов журналов

Ссылки, связанные с данной темой:

- Приложение G, “Обработчик пользователя для восстановления баз данных” на стр. 345

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 операционных системах Windows® нельзя использовать обработчик пользователя REXX для архивирования файлов журнала.
- 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.
- При запуске обработчика пользователя для архивирования файла журнала DB2® 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 эффективнее использовать повторно, чем освобождать и снова запрашивать страницы на диске.
- DB2 *не* вызывает программу обработчика пользователя во время восстановления после сбоя и отката, если параметр конфигурации базы данных *logsecond* не равен -1.
- 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. При последнем отсоединении от базы данных активный файл журнала может оказаться неполным. 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.
- Если активированы программы обработчика пользователя, и в качестве устройства для хранения журналов и образов резервной копии применяется лентопротяжное устройство, убедитесь, что в качестве расположения назначения образов резервных копий и архивных журналов не задано то же лентопротяжное устройство. Поскольку во время резервного копирования могут выполняться некоторые операции архивирования журналов, то может возникнуть ошибка, связанная с тем, что два процесса попытаются одновременно записать данные на одно и то же лентопротяжное устройство.
- 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.
- Если программа обработчика пользователя включена для нескольких баз данных, применяющих одно и то же лентопротяжное устройство для архивирования файлов журнала, то при повторе транзакций для одной из этих баз данных другие базы данных не должны быть активны. Если другая база данных попытается заархивировать файл журнала во время выполнения операции повтора транзакций, то могут быть не найдены журналы, необходимые для повтора транзакций, либо новый архивированный файл журнала может заменить те файлы журнала, которые были ранее сохранены на лентопротяжном устройстве.

  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.

Понятия, связанные с данным:

- “Управление файлами журнала” на стр. 50

Создание и удаление файлов журналов

Файлы журналов из каталога журналов базы данных не удаляются, если они могут потребоваться для восстановления после сбоя. Если включен параметр конфигурации базы данных *userexit*, то весь файл журнала может быть удален только в том случае, если он не нужен для восстановления после сбоя. Файл журнала, необходимый для восстановления после сбоя, называется активным журналом. Файл журнала, ненужный для восстановления после сбоя, называется архивным журналом.

Способ создания новых файлов журнала и удаления старых файлов журнала зависит от значений параметров конфигурации базы данных *userexit* и *logretain*:

Оба параметра *logretain* **и** *userexit* **равны** `OFF`

Будет применяться циклическая регистрация в журнале. При таком способе регистрации восстановление путем повтора транзакций не поддерживается, а восстановление после сбоя поддерживается.

При циклической регистрации новые файлы журналов, отличные от дополнительных журналов, не создаются, а старые файлы не удаляются.

Данные записываются в файлы журнала циклическим образом. Это означает, что при заполнении последнего файла журнала DB2® снова начинает записывать данные в первый файл журнала.

Переполнение журнала может возникнуть в том случае, если все файлы журнала активны, и процесс циклической регистрации не может перейти к первому файлу журнала. Если все основные файлы журнала заполнены и активны, то создаются дополнительные файлы журнала. Дополнительный журнал удаляется только во время перезапуска базы данных.

Logretain **равен ON, а** *userexit* **равен OFF**
Допустимо как восстановление путем повтора транзакций, так и восстановление после сбоя. База данных является восстановимой. Если параметр *userexit* равен OFF, то DB2 не удаляет файлы журнала из каталога журналов базы данных. При заполнении очередного файла журнала DB2 начинает записывать данные в новый файл, если максимальное число основных и дополнительных файлов журнала еще не достигнуто.

Userexit **равен ON**
Если оба параметра *logretain* и *userexit* равны on, то допустимо как восстановление путем повтора транзакций, так и восстановление после сбоя. После заполнения файла журнала он автоматически архивируется с помощью программы обработчика пользователя.

Файлы журнала обычно не удаляются. Если потребовался новый файл журнала, а свободного файла нет, то переименовывается и заново используется один из архивных файлов журнала. Архивный файл журнала не удаляется и не переименовывается после того, как он был закрыт и скопирован в каталог архивных журналов. Когда DB2 требуется новый файл журнала, то переименовывается самый старый архивный журнал. Файл журнала, перемещенный в каталог базы данных во время восстановления, удаляется, как только он становится ненужным. Если в DB2 есть свободное место в пространстве журналов, то в каталоге базы данных будут находиться старые файлы журнала.

Если во время архивации файла журнала возникнет ошибка, то процедура архивации будет приостановлена на пять минут, после чего будет сделана еще одна попытка выполнить операцию. После этого DB2 продолжит архивировать файлы журнала по мере их заполнения. Если какие-либо файлы заполнятся в течение пятиминутного периода ожидания, они не будут заархивированы сразу после истечения этого периода. DB2 распределит архивацию этих файлов во времени.

Самый простой способ удалить старые файлы журнала - перезапустить базу данных. После перезапуска базы данных в ее каталоге останутся только новые файлы журнала и файлы журнала, которые не удалось заархивировать программе обработчика пользователя.

Минимальное число журналов, которые останутся в каталоге журналов после перезапуска базы данных, равно числу основных журналов. Число основных журналов можно задать в параметре конфигурации базы данных *logprimary*. Возможно, что число журналов в каталоге журналов будет превышать число основных журналов. Это может произойти в том случае, если число пустых журналов в каталоге на момент завершения работы базы данных будет больше значения параметра конфигурации *logprimary* на момент перезапуска базы данных. Такая ситуация возможна, если в промежутке между завершением работы базы данных и ее запуском было изменено значение параметра конфигурации *logprimary*, либо если были созданы, но не были использованы дополнительные журналы.

Если во время перезапуска базы данных число пустых журналов меньше числа основных журналов, заданного в параметре конфигурации *logprimary*, то будут созданы недостающие файлы журнала. Если число пустых журналов в каталоге базы данных превосходит число основных журналов, то при перезапуске базы данных будет оставлено столько пустых журналов, сколько было в каталоге на момент завершения работы базы данных. Созданные дополнительные файлы журналов останутся в каталоге активных журналов после перезапуска базы данных.

Запрет транзакций при переполнении каталога журнала

Параметр конфигурации базы данных *blk_log_dsk_ful* позволяет предотвратить генерирование сообщений об ошибках, связанных с переполнением диска, когда DB2® не удается создать новый файл журнала по действующему пути журнала.

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 файл архивного журнала успешно архивирован, DB2 можете переименовать этот файл журнала и продолжить работу. После каждой попытки DB2 записывает сообщение в журнал уведомлений администратора. Единственный способ убедиться в том, что программа зависает из-за переполнения диска журнала - отслеживать поступление сообщений в журнал уведомлений администратора.

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.

Понятия, связанные с данным:

- “Что такое журналы восстановления” на стр. 38
- “Управление файлами журнала с программой обработчика пользователя” на стр. 52

Архивирование журнала по требованию

Теперь DB2® поддерживает закрытие (и, если включена опция обработчика пользователя, архивирование) активного журнала базы данных с возможностью восстановления 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 запросу, введя команду ARCHIVE LOG или вызвав API **db2ArchiveLog**.

Ссылки, связанные с данной темой:

- “ARCHIVE LOG” на стр. 243
- “db2ArchiveLog - Archive Active Log” в *Administrative API Reference*

Применение 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 нагрузки.
- 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*. 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. С помощью системного вызова DB2® определит размер 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 включен параметр конфигурации *userexit*, убедитесь, что число журналов, которые можно разместить на непосредственном устройстве, превосходит значение параметра конфигурации *logprimary*. Это компенсирует задержку, возникающую при архивировании файла журнала с помощью программы обработчика пользователя.

- 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.
- Если для хранения журналов было выбрано непосредственное устройство, и применяется программа DataPropagator™ (DPROP), либо другая программа, вызывающая API **db2ReadLog**, то необходимо задать параметр конфигурации базы данных *overflowlogpath*. Для предоставления данных журнала, запрошенных API **db2ReadLog**, DB2 может потребоваться восстановить файл журнала. Этот файл будет помещен в каталог, заданный в параметре конфигурации базы данных *overflowlogpath*.

Задачи, связанные с данной темой:

- “Использование прямого ввода-вывода” в *Руководство администратора: Реализация*

Ссылки, связанные с данной темой:

- “db2ReadLog - Асинхронное чтение журнала” на стр. 283
- Приложение F, “Tivoli Storage Manager” на стр. 341

Как избежать потери файлов журнала

При отбрасывании базы данных и восстановлении путем повтора транзакций до указанного момента времени могут быть утеряны некоторые файлы журналов. Эти файлы могут потребоваться в будущем при выполнении операций восстановления. В связи с этим необходимо хранить копии всех журналов, расположенных в текущем каталоге журналов базы данных. Рассмотрим следующие ситуации:

- Если вы планируете отбросить базу данных перед выполнением операции восстановления, то сохраните все файлы журнала из пути активного журнала перед вызовом команды DROP DATABASE. После восстановления базы данных эти файлы журнала могут потребоваться для восстановления путем повтора транзакций, так как некоторые из них могли быть не заархивированы на момент отбрасывания базы данных. Обычно базу данных не нужно отбрасывать перед вызовом команды RESTORE. Базу данных требуется отбросить целиком (или только в одном разделе, указав параметр AT NODE в команде DROP DATABASE), если она повреждена настолько, что команду RESTORE выполнить не удается. Кроме того, базу данных можно отбросить перед ее восстановлением для получения новой точки отсчета.
- Если для базы данных выполняется повтор транзакций до заданного момента времени, то все данные журнала, занесенные после указанного момента времени, будут заменены. Если после завершения операции повтора транзакций и установления соединения с базой данных будет обнаружено, что требовалось повторить транзакции до более позднего момента времени, то вы не сможете исправить ошибку, поскольку необходимые данные в журналах будут уже заменены. Возможно, что исходный набор файлов журнала сохранится в архиве. Однако нельзя исключать вероятность того, что DB2® вызовет программу обработчика пользователей для автоматического архивирования новых файлов журнала. В зависимости от алгоритма работы программы обработчика пользователя, она может заменить исходный набор файлов журнала в каталоге архивного журнала. Даже если в каталоге архивного журнала присутствует исходный и новый набор файлов журнала (в виде разных версий одних и тех же файлов), вам потребуется определить, какой набор журналов следует использовать для последующих операций восстановления.

Понятия, связанные с данным:

- “Что такое журналы восстановления” на стр. 38

Что такое файл хронологии восстановления

Файл хронологии восстановления создается для каждой базы данных и автоматически обновляется в следующих случаях:

- 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

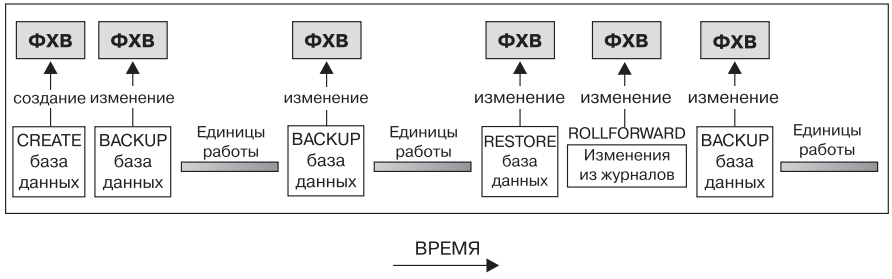


Рисунок 9. Создание и обновление файла хронологии восстановления

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.

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. При выполнении операции восстановления содержимое в файле хронологии восстановления не заменяется, за исключением случая, когда в файле на диске нет записей. В этом случае хронологии базы данных будет восстановлена из резервной копии.

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. Указав значение -1, можно задать неограниченный период хранения. В этом случае сокращение файла надо выполнять явно.

Ссылки, связанные с данной темой:

- “Recovery History Retention Period configuration parameter - rec_his_retentn” в *Руководство администратора: Производительность*
- “LIST HISTORY” на стр. 247

Файл хронологии восстановления - Сбор мусора

Чистка мусора

Хотя с помощью команды PRUNE HISTORY из файла хронологии можно в любой момент удалить записи, рекомендуется, чтобы очистка файла выполнялась DB2. Число резервных копий баз данных DB2®, записанных в файле хронологии восстановления, автоматически отслеживается функцией *чистки мусора* DB2. Эта функция вызывается в следующих случаях:

- После успешного создания полной (а не инкрементной) резервной копии базы данных.

- После успешного восстановления базы данных, в ходе которого не выполнялся повтор транзакций.
- После успешного повтора транзакций для базы данных.

Число сохраняемых активных полных (не инкрементных) резервных копий базы данных определяет параметр конфигурации *num_db_backups*. Значение этого параметра конфигурации используется для просмотра файла хронологии, начиная с последней записи.

После создания очередной полной (не инкрементной) резервной копии базы данных из файла хронологии удаляются записи с истекшим сроком хранения, используя значение параметра *rec_his_retentn*.

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.

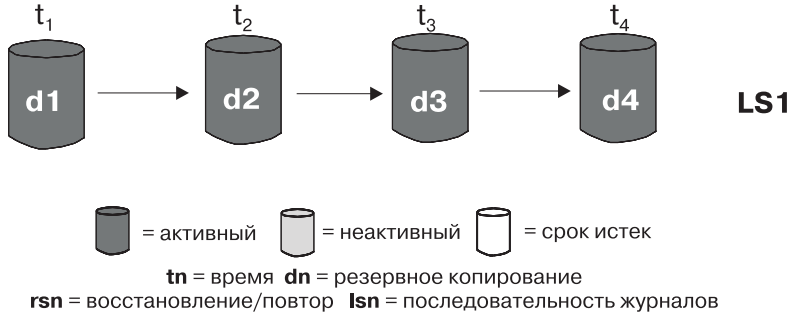


Рисунок 10. Активные резервные копии базы данных. Значение *num_db_backups* равно 4.

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”.

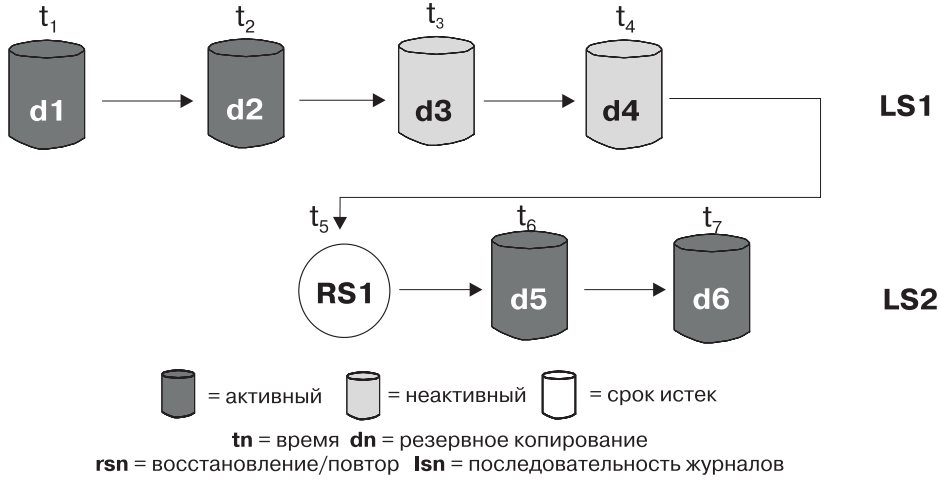


Рисунок 11. Неактивные резервные копии базы данных

Если восстановлена активная, но не самая последняя резервная копия базы 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”.

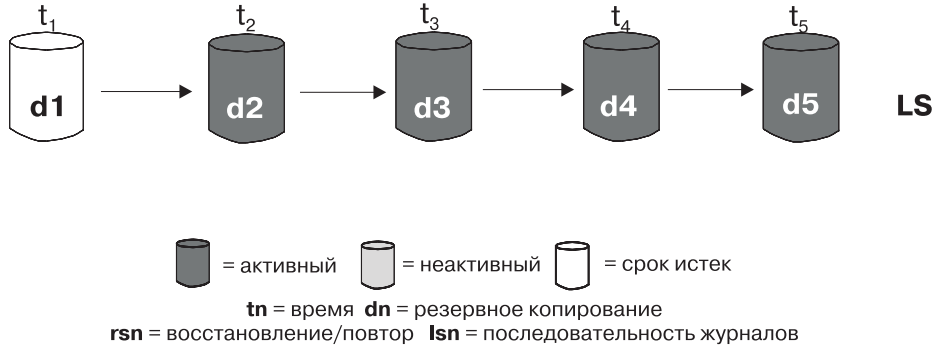


Рисунок 12. Резервные копии базы данных с истекшим сроком годности

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.

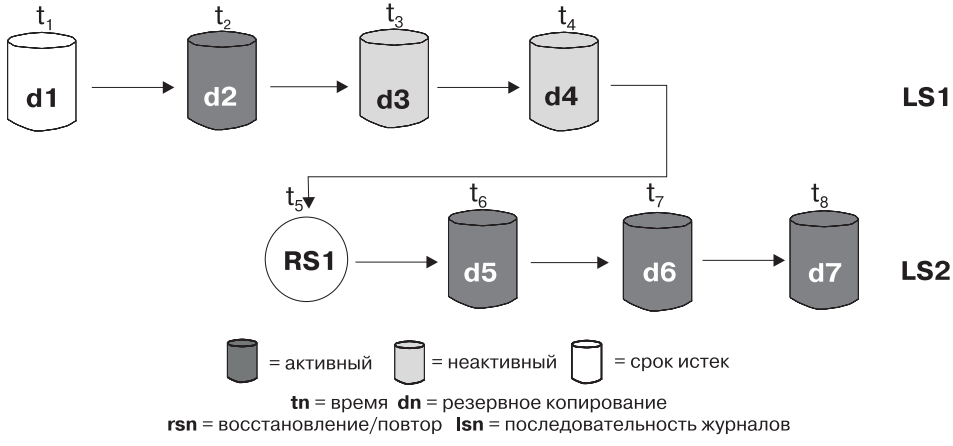


Рисунок 13. Резервные копии базы данных - активные, неактивные и с истекшим сроком

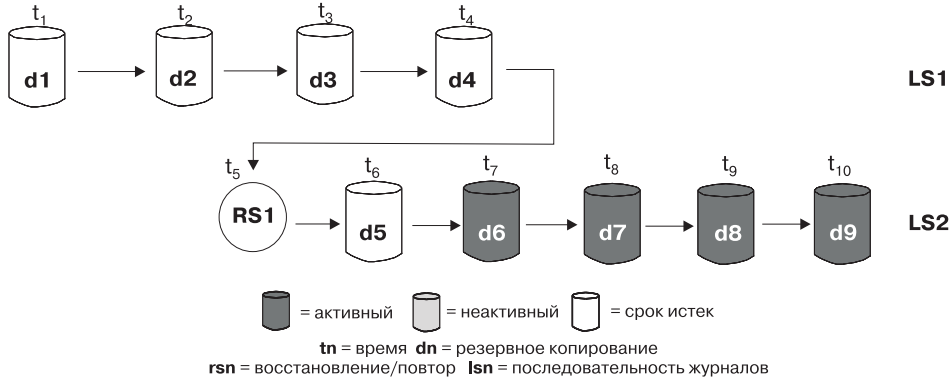


Рисунок 14. Последовательность файлов журнала с истекшим сроком годности

Понятия, связанные с данным:

- “Что такое файл хронологии восстановления” на стр. 61

Ссылки, связанные с данной темой:

- “PRUNE HISTORY/LOGFILE” на стр. 249

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

Исследуя вопрос повышения производительности восстановления, следует учитывать следующее:

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

  Следует также учитывать, какие еще файлы находятся на этом диске. Например, перемещение журналов на диск, используемый для подкачки страниц в системе, где не хватает реальной памяти, сведет на нет усилия по настройке.
- Чтобы сократить продолжительность операции восстановления:
  - Скорректируйте размер буфера восстановления. Размер этого буфера должен быть кратен размеру буфера, использовавшегося при резервном копировании.
  - Увеличьте число буферов.

    Если применяется несколько буферов и накопителей ввода-вывода, то число буферов должно по крайней мере вдвое превышать число накопителей, чтобы накопителям не приходилось ждать данных. 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 сумме числа накопителей и значения параметра 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.
  - Задавайте опцию PARALLELISM

Повышение производительности восстановления - Параллельное восстановление

Параллельное восстановление

Для восстановления после сбоя и восстановления путем повтора транзакций DB2® применяет несколько агентов. 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.

Для поддержки параллельного восстановления был добавлен новый тип агента - db2agnsc. DB2 выбирает число агентов, используемых при восстановлении базы данных, на основе числа процессоров в компьютере.

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.

Понятия, связанные с данным:

• “Повышение производительности восстановления” на стр. 68

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.

В этом приложении рассматриваются следующие темы:

Обзор резервного копирования

В самом простом вызове 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 операционных системах Windows® эта команда (вызванная из корневого каталога) создает образ резервной копии, который будет выглядеть в списке файлов каталога следующим образом:

```
 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
```

Примечание: Если сервер и клиент DB2 расположены в разных системах, то по умолчанию образ резервной копии создается в текущем рабочем

каталоге системы клиента, в которой была вызвана команда. Такой каталог или устройство назначения должны существовать в системе сервера.

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)
- Сервером другого поставщика

При создании резервной копии базы данных в файле хронологии восстановления автоматически обновляется сводка информации. Этот файл создается в том же каталоге, что и файл конфигурации базы данных.

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

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

Например:

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

В операционных системах Windows создается четырехуровневое дерево каталогов:

```
алиас_б_д.тип\имя_экземп\NODEnnnn\CATNnnnn\ггггммдд\ччммсс.пор_номер
```

Например:

```
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.
- 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.

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.

Просмотр 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.
- Вывести информацию, хранящуюся в заголовке резервной копии.
- Просмотреть информацию об объектах, а также заголовок файла журнала из резервной копии.

Понятия, связанные с данным:

- “Что такое файл хронологии восстановления” на стр. 61

Ссылки, связанные с данной темой:

- “db2ckbkp - Проверка резервной копии” на стр. 231
- Приложение F, “Tivoli Storage Manager” на стр. 341

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

Предварительные требования для установки:

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 как локальной, так и 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.) 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.

Ограничения:

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 была 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.

Процедура:

Утилиту резервного копирования можно вызвать с помощью процессора командной строки (CLP), окна Резервное копирование базы данных или мастера Центра управления, либо с помощью API **db2Backup**.

Ниже приведен пример команды BACKUP DATABASE, вызванной с помощью процессора командной строки:

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

Для того чтобы открыть записную книжку Резервное копирование базы данных 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.

Подробную информацию можно найти в электронной справке Центра управления.

Понятия, связанные с данным:

- "Administrative APIs in Embedded SQL or DB2 CLI Programs" в *Application Development Guide: Programming Client Applications*
- "Встраиваемые модули Центра управления - Введение" в *Руководство администратора: Реализация*

Задачи, связанные с данной темой:

- "Преобразование баз данных" в *Quick Beginnings for DB2 Servers*

Ссылки, связанные с данной темой:

- "LIST DBPARTITIONNUMS Command" в *Command Reference*
- "db2Backup - Резервное копирование базы данных" на стр. 86

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 размера 512, 1024, 2048 или 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 лентопротяжных устройств. Максимальную производительность дает размер буфера, равный максимальному размеру блока для используемого устройства.

Перед применением лентопротяжного устройства в операционной системе Windows® нужно вызвать следующую команду:

```
db2 initialize tape on <устройство> using <размер_блока>
```

Где:

<устройство>
: допустимое имя лентопротяжного устройства. По умолчанию в операционных системах Windows применяется имя `\\.\TAPE0`.

<размер-блока>
: размер блока для магнитной ленты. Это значение должно быть кратно 4096 или являться множителем 4096. Значение по умолчанию - это размер блока по умолчанию для 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. Ниже приведен 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 `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 лентопротяжного устройства на другое. Это возможно, если образ занимает не более одной магнитной ленты. При использовании двух лентопротяжных устройств команда **dd** выглядит следующим образом:

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

Если использование двух лентопротяжных 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 лентопротяжных 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.

Предварительные требования:

Процесс, записывающий данные в именованный конвейер, и процесс, читающий из него данные, должны находиться на одном компьютере. Конвейер должен существовать и располагаться в локальной файловой системе. Поскольку именованный конвейер воспринимается как локальное устройство, не нужно специально указывать, что данные записываются в конвейер.

Процедура:

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
   ```

Задачи, связанные с данной темой:

- “Использование резервного копирования” на стр. 75

Ссылки, связанные с данной темой:

- “BACKUP DATABASE” на стр. 80
- “RESTORE DATABASE” на стр. 105

BACKUP DATABASE

Создает резервную копию базы данных или табличного пространства.

Область:

Эта команда применяется только к разделу базы данных, на котором она выполняется.

Авторизация:

Один из следующих вариантов:

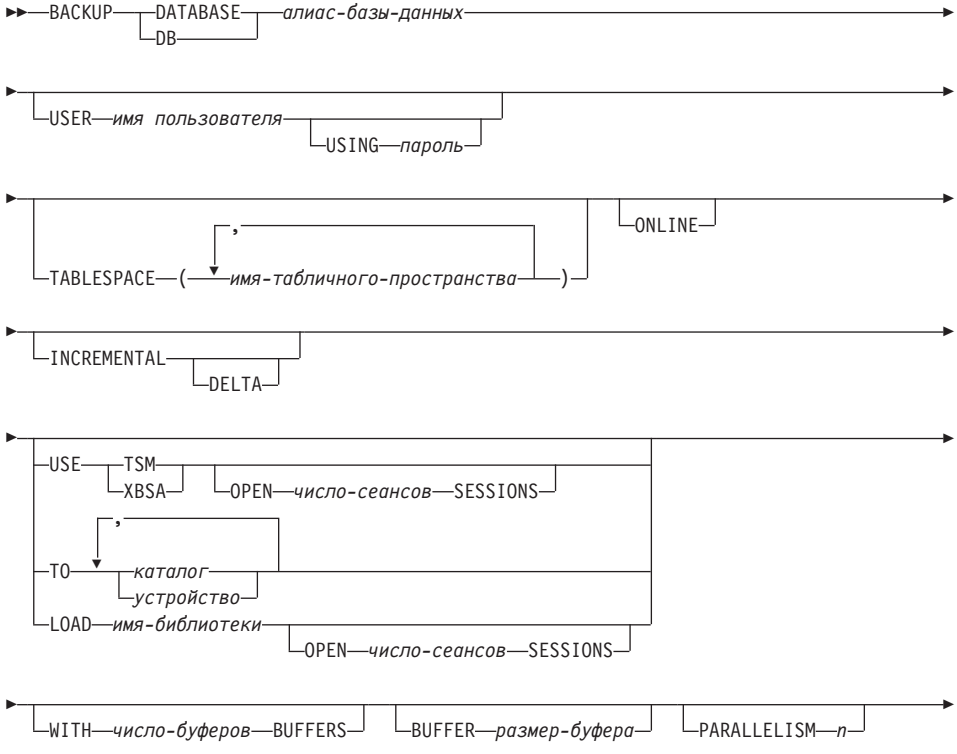
- *sysadm*
- *sysctrl*
- *sysmaint*

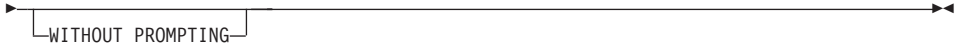
Необходимое соединение:

База данных. Эта команда автоматически устанавливает соединение с указанной базой данных.

Примечание: Если соединение с базой данных уже существует, то оно будет прервано, а для резервного копирования будет создано новое соединение. После завершения резервного копирования соединение будет прервано.

Синтаксис команды:





Параметры команды:

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

USER имя пользователя
Указывает имя пользователя, под именем которого будет выполняться резервное копирование базы данных.

USING пароль
Пароль 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
Указывает, что необходимо создать кумулятивную (инкрементальную) резервную копию. Инкрементальная резервная копия - это копия всех данных базы данных, которые изменились со времени последней успешной операции полного резервного копирования.

DELTA
Указывает, что необходимо создать некумулятивную (разностную) резервную копию. Разностная резервная копия - это копия всех данных базы данных, измененных с момента 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 должен применяться 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 - это менеджер памяти, поддерживающий в настоящее время интерфейс XBSA.

TO каталог/устройство

Список имен каталогов или ленточных устройств хранения. Необходимо указывать полный путь к каталогу. Если опции USE TSM, TO и LOAD не указаны, то по умолчанию резервная копия помещается в текущий рабочий каталог компьютера-клиента. Каталог или устройство назначения должны существовать 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 операционные системы Windows не содержат стандартных средств поддержки лентопротяжных устройств, для каждого такого устройства требуется собственный драйвер. Для резервного копирования файловой системы FAT в операционных системах Windows необходимо следовать ограничению на именование 8.3.

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

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

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

t Прервать. Прекратить операцию 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 библиотеки (в Windows - DLL), содержащей разработанные независимым поставщиком функции ввода-вывода для резервного копирования и восстановления. Можно вводить полное имя. Если не вводить полное имя, по умолчанию используется путь, по которому расположена программа обработчика пользователя.

WITH число-буферов BUFFERS

Число применяемых буферов. 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 этого параметра - 8 страниц; значение по умолчанию - 1024 страницы. Если указан нулевой размер буфера, для размера буфера используется значение параметра конфигурации менеджера баз данных *backbufsz*.

При использовании переменного размера блока сократите размер буфера до значения, поддерживаемого лентопротяжным устройством. В противном случае операция резервного копирования может закончиться успешно, но полученная резервная копия не будет пригодна для восстановления.

При использовании лентопротяжных устройств в SCO UnixWare 7 укажите 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. Значение по умолчанию - 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.

Примеры:

В следующем примере база данных WSDB определена на всех 4 разделах с номерами от 0 до 3. У всех разделов есть доступ к каталогу /dev3/backup . Раздел 0 - это раздел каталога, для которого резервную копию следует создавать отдельно, в автономном режиме. Для создания в автономном режиме резервной копии всех разделов базы данных WSDB в /dev3/backup укажите в одном из разделов базы данных следующую команду:

```
db2_all '<<+0< db2 BACKUP DATABASE wsdb TO /dev3/backup'
db2_all '|<<-0< db2 BACKUP DATABASE wsdb TO /dev3/backup'
```

Во второй команде утилита db2_all выполнит команду резервного копирования для всех разделов базы данных по очереди (за исключением раздела 0). Все четыре резервных копии разделов будут помещены в каталог /dev3/backup

В следующем примере выполняется резервное копирование базы данных SAMPLE на сервер TSM при помощи двух одновременных сеансов клиентов TSM. Утилита резервного копирования применяет четыре буфера, имеющих размер по умолчанию (1024 x 4K страниц).

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

В следующем примере выполняется резервное копирование на ленты табличных пространств (syscatspace, userspace1) базы данных payroll.

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

Ссылки, связанные с данной темой:

- “RESTORE DATABASE” на стр. 105
- “ROLLFORWARD DATABASE” на стр. 146

db2Backup - Резервное копирование базы данных

Создает резервную копию базы данных или табличного пространства.

Область:

Этот API влияет только на разделы базы данных, в которых он вызван.

Авторизация:

Одни из следующих прав доступа:

- *sysadm*
- *sysctrl*
- *sysmaint*

Необходимое соединение:

База данных. Этот API автоматически устанавливает соединение с указанной базой данных.

После завершения резервного копирования соединение будет прервано.

Включаемый файл API:

db2ApiDf.h

Синтаксис C API:

```
/* Файл: db2ApiDf.h */
/* API: db2Backup */
/* ... */
SQL_API_RC SQL_API_FN
db2Backup (
  db2Uint32 versionNumber,
  void         *pDB2BackupStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2BackupStruct
{
  char                        *piDBAlias;
  char                        oApplicationId[SQLU_APPLID_LEN+1];
  char                        oTimestamp[SQLU_TIME_STAMP_LEN+1];
  struct db2TablespaceStruct  *piTablespaceList;
  struct db2MediaListStruct   *piMediaList;
  char                        *piUsername;
  char                        *piPassword;
  void                        *piVendorOptions;
  db2Uint32                   iVendorOptionsSize;
  db2Uint32                   oBackupSize;
  db2Uint32                   iCallerAction;
  db2Uint32                   iBufferSize;
  db2Uint32                   iNumBuffers;
  db2Uint32                   iParallelism;
  db2Uint32 iOptions;
} db2BackupStruct;

typedef SQL_STRUCTURE db2TablespaceStruct
{
  char                        **tablespaces;
  db2Uint32                   numTablespaces;
} db2TablespaceStruct;

typedef SQL_STRUCTURE db2MediaListStruct
{
  char                        **locations;
  db2Uint32                   numLocations;
  char                        locationType;
} db2MediaListStruct;
/* ... */
```

Общий синтаксис API:

```
/* Файл: db2ApiDf.h */
/* API: db2Backup */
/* ... */
SQL_API_RC SQL_API_FN
db2gBackup (
  db2Uint32 versionNumber,
  void         *pDB2gBackupStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2gBackupStruct
{
  char                         *piDBAlias;
  db2Uint32                    iDBAliasLen;
  char                         *poApplicationId;
  db2Uint32                    iApplicationIdLen;
  char                         *poTimestamp;
  db2Uint32                    iTimestampLen;
  struct db2gTablespaceStruct *piTablespaceList;
  struct db2gMediaListStruct  *piMediaList;
  char                         *piUsername;
  db2Uint32                    iUsernameLen;
  char                *piPassword;
  db2Uint32 iPasswordLen;
  void                         *piVendorOptions;
  db2Uint32                    iVendorOptionsSize;
  db2Uint32                    oBackupSize;
  db2Uint32                    iCallerAction;
  db2Uint32                    iBufferSize;
  db2Uint32                    iNumBuffers;
  db2Uint32                    iParallelism;
  db2Uint32 iOptions;
} db2gBackupStruct;

typedef SQL_STRUCTURE db2gTablespaceStruct
{
  struct db2Char              *tablespaces;
  db2Uint32                    numTablespaces;
} db2gTablespaceStruct;

typedef SQL_STRUCTURE db2gMediaListStruct
{
  struct db2Char              *locations;
  db2Uint32                    numLocations;
  char                         locationType;
} db2gMediaListStruct;

typedef SQL_STRUCTURE db2Char
{
   char            *pioData;
   db2Uint32       iLength;
   db2Uint32       oLength;
} db2Char;
/* ... */
```

Параметры API:

versionNumber
Входной. Задает уровень версии и выпуска для структуры, передаваемой во втором параметре - *pDB2BackupStruct*.

pDB2BackupStruct
Входной. Указатель на структуру *db2BackupStruct*.

pSqlca Выходной. Указатель на структуры *sqlca*.

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

iDBAliasLen
Входной. 4-байтное целое без знака, равное длине (в байтах) алиаса базы данных.

oApplicationId
Выходной. 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.

poApplicationId
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.

iApplicationIdLen
Входной. 4-байтное целое без знака, задающее длину (в байтах) буфера *poApplicationId*. Должно быть равно SQLU_APPLID_LEN+1 (определено в `sqlutil`).

oTimestamp
Выходной. В этом буфере API возвращает значение отметки времени для образа резервной копии.

poTimestamp
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.

iTimestampLen
Входной. 4-байтное целое без знака, задающее длину (в байтах) буфера *poTimestamp*. Должно быть равно SQLU_TIME_STAMP_LEN+1 (определено в `sqlutil`).

piTablespaceList
Входной. Список табличных пространств, для которых нужно сделать резервные копии. Требуется только для операций резервного копирования уровня табличного пространства. На уровне базы данных должен быть равен NULL. См. структуру DB2TablespaceStruct.

piMediaList
Входной. В этой структуре вызывающая программа может задать назначение для операции резервного копирования. Передаваемая информация зависит от значения поля locationType. Допустимые значения locationType (определены в `sqlutil.h`):

SQLU_LOCAL_MEDIA
Локальные устройства (сочетание лент, дисков или дискет).

SQLU_TSM_MEDIA
TSM. Если указатель равен NULL, то применяется общая библиотека TSM, поставляемая вместе с DB2. 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 поле Положение.

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

Более подробная информация приведена в описании структуры DB2MediaListStruct.

piUsername
Входной. Строка с именем пользователя, используемым при установке соединения. Может иметь пустое значение (NULL).

iUsernameLen
Входной. 4-байтное целое без знака, равное длине (в байтах) имени пользователя. Задайте значение ноль, если имя пользователя не указано.

piPassword
Входной. Строка с паролем, используемым с этим именем пользователя. Может иметь пустое значение (NULL).

iPasswordLen
Входной. 4-байтное целое без знака, равное длине (в байтах) пароля. Задайте значение ноль, если пароль не указан.

piVendorOptions
Входной. Используется для передачи информации от прикладной

программы к функции 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.

iVendorOptionsSize
Входной. Длина поля *piVendorOptions*; не должна превышать 65535 байт.

oBackupSize
Выходной. Размер образа резервной копии (в Мбайтах).

iCallerAction
Входной. Задает выполняемое действие. Допустимые значения (заданы в `db2ApiDf`):

DB2BACKUP_BACKUP
Запустить операцию резервного копирования.

DB2BACKUP_NOINTERRUPT
Запустить операцию резервного копирования. Указывает, что резервное копирование будет запущено автономно, а для сценариев, в которых обычно требуется вмешательство пользователя, будет предпринята попытка выполнения без возвращения к вызывающей программе, либо 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.

DB2BACKUP_CONTINUE
Продолжить операцию резервного копирования после того, как пользователь выполнил действия, запрошенные утилитой резервного копирования (например, смонтировать новую ленту).

DB2BACKUP_TERMINATE
Прервать операцию резервного копирования после того, как пользователь не смог выполнить действия, запрошенные утилитой резервного копирования.

DB2BACKUP_DEVICE_TERMINATE
Удалить конкретное устройство из списка устройств, используемых для 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.

DB2BACKUP_PARM_CHK
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.

DB2BACKUP_PARM_CHK_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.

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

iNumBuffers
Входной. Задает число используемых буферов резервного копирования. Минимальное значение - 2. Максимальное ограничено объемом памяти. 0 задает значение по умолчанию 2.

iParallelism
Входной. Степень параллелизма (число манипуляторов буферов). Минимальное значение - 1. Максимальное - 1024. Значение по умолчанию - 1.

iOptions
Входной. Способ объединения параметров резервного копирования. Для получения значения *iOptions* опции комбинируются с помощью побитового оператора OR. Допустимые значения (заданы в `db2ApiDf`):

DB2BACKUP_OFFLINE
В автономном режиме; только утилита резервного копирования может соединяться с базой данных.

DB2BACKUP_ONLINE
В оперативном режиме; при выполнении резервного копирования другим прикладным программам разрешено соединяться с базой данных.

Примечание: Если пользователи заблокировали данные SMS LOB, то резервное копирование может зависнуть.

DB2BACKUP_DB
Полное резервное копирование.

DB2BACKUP_TABLESPACE
Резервное копирование табличных пространств. Укажите список табличных пространств в параметре *piTablespaceList*.

DB2BACKUP_INCREMENTAL
Задает образ кумулятивной (инкрементной) резервной копии. Образ инкрементной резервной копии - это копия всей информации базы данных, которая изменилась с момента последней успешной операции полного резервного копирования.

DB2BACKUP_DELTA
Задает образ некумулятивной (разностной) резервной копии. Образ разностной резервной копии - это копия всей информации базы данных, которая изменилась с момента последней успешной операции резервного копирования любого типа.

tablespaces
Указатель на список табличных пространств, для которых нужно сделать резервные копии. В программах C список должен быть строкой, завершающейся символом NULL. В общем случае, это список структур *db2Char*.

numTablespaces
Число записей в списке *tablespaces*.

locations
Указатель на список положений носителей. В программах C список должен быть строкой, завершающейся символом NULL. В общем случае, это список структур db2Char.

numLocations
Число записей в списке *locations*.

locationType
Символ, обозначающий тип носителя. Допустимые значения (определены в `sqlutil.h`):

SQLU_LOCAL_MEDIA
Локальные устройства (ленты, диски, дискеты или именованные конвейеры).

SQLU_TSM_MEDIA
Tivoli Storage Manager.

SQLU_OTHER_MEDIA
Библиотека другого поставщика.

SQLU_USER_EXIT
Обработчик пользователя (доступен только в OS/2).

pioData
Указатель на буфер символьных данных.

iLength
Входной. Размер буфера *pioData*.

oLength
Выходной. Зарезервирован для будущего использования.

Примеры, связанные с данной темой:

- “dbrecov.sqc -- How to recover a database (C)”
- “dbrecov.sqC -- How to recover a database (C++)”

Сеансы резервного копирования - Примеры для CLP

Пример 1

В следующем примере база данных SAMPLE копируется на сервер TSM с помощью двух параллельных сеансов клиента TSM. Утилита резервного копирования будет применять четыре буфера размером по умолчанию (1024 страницы по 4 Кбайта).

```
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
```

Пример 2

Следующий пример иллюстрирует стратегию еженедельного инкрементного резервного копирования восстановимой базы данных. Она включает в себя операцию еженедельного полного резервного копирования базы данных, операцию ежедневного разностного резервного копирования и операцию дополняющего (инкрементного копирования), выполняемую 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 3

Для того чтобы создать резервную копию на лентопротяжном устройстве в среде Windows, вызовите следующую команду:

```
db2 backup database sample to \\.\tape0
```

Задачи, связанные с данной темой:

- “Использование резервного копирования” на стр. 75

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.

  Если вы используете несколько каналов ввода/вывода, число буферов должно по крайней мере вдвое превышать число каналов, чтобы каналам не приходилось ждать данных.
- Используйте несколько устройств назначения.

Понятия, связанные с данным:

- “Обзор резервного копирования” на стр. 71

Задачи, связанные с данной темой:

- “Использование резервного копирования” на стр. 75

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.

В этом приложении рассматриваются следующие темы:

Обзор процедуры восстановления

В простейшем варианте команды DB2® RESTORE DATABASE необходимо задать только алиас базы данных, которую нужно восстановить. Например:

```
db2 restore db sample
```

Так как база данных SAMPLE существует, то в этом примере 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 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.

Повышение производительности восстановления

Для того чтобы сократить продолжительность операции восстановления:

- Увеличьте размер буфера восстановления.

  Размер буфера восстановления должен быть положительным целым числом, кратным размеру буфера, заданному для резервного копирования. 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 равно 16.
- Увеличьте значение опции PARALLELISM.

  В результате будет увеличено число манипуляторов буферов (BM), которые будут применяться для записи информации в базу данных во время операции восстановления. Значение по умолчанию равно 1.

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

Предварительные требования для установки:

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 как локальной, так и удаленной.

Ограничения:

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.
- 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.

Процедура:

Утилиту восстановления можно вызвать с помощью процессора командной строки (CLP), окна Восстановление базы данных или мастера Центра управления, либо API **db2Restore**.

Пример команды RESTORE DATABASE, введенной в командной строке:

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

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. Все существующие базы данных будут показаны на панели содержимого в правой части окна.
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.

Подробную информацию можно найти в электронной справке Центра управления.

Понятия, связанные с данным:

- “Administrative APIs in Embedded SQL or DB2 CLI Programs” в *Application Development Guide: Programming Client Applications*
- “Встраиваемые модули Центра управления - Введение” в *Руководство администратора: Реализация*

Ссылки, связанные с данной темой:

- “db2Restore - Восстановление базы данных” на стр. 114

Применение инкрементного восстановления в тестовой и рабочей среде

Если для рабочей базы данных включена опция инкрементного резервного копирования и восстановления, то вы можете создать или обновить текстовую базу данных с помощью инкрементной или разностной резервной копии. Для этого нужно выполнить инкрементное восстановление автоматически или вручную. Для восстановления резервной копии рабочей базы данных в тестовой базе данных укажите опцию INTO *алиас_базы_данных_назначения* в команде RESTORE DATABASE. Например, если для рабочей базы данных созданы следующие резервные копии:

```
backup db prod
Резервное копирование выполнено. Отметка времени копии: <ts1>

backup db prod incremental
Резервное копирование выполнено. Отметка времени копии: <ts2>
```

то инкрементное восстановление можно выполнить вручную с помощью следующих команд:

```
restore db prod incremental taken at ts2 into test without prompting
DB20000I  Команда RESTORE DATABASE выполнена успешно.

restore db prod incremental taken at ts1 into test without prompting
DB20000I  Команда RESTORE DATABASE выполнена успешно.

restore db prod incremental taken at ts2 into test without prompting
DB20000I  Команда RESTORE DATABASE выполнена успешно.
```

Если база данных TEST уже существует, то во время восстановления будут заменены все содержащиеся в ней данные. Если база данных TEST не существует, то утилита восстановления создаст ее и заполнит данными из резервных копий.

Поскольку операции автоматического инкрементного восстановления зависят от содержимого хронологии базы данных, выполняемые действия по восстановлению зависят от того, существует ли тестовая база данных. Для того чтобы в базе данных TEST можно было выполнить автоматическое инкрементное восстановление, ее хронология должна содержать хронологию резервной копии базы данных PROD. Хронология базы данных из резервной копии заменит хронологию базы данных TEST, если:

- база данных TEST не существует на момент вызова команды RESTORE DATABASE или
- база данных TEST существует на момент вызова команды RESTORE DATABASE, но в ее хронологии нет записей.

Ниже приведен пример автоматического инкрементного восстановления для случая, когда база данных TEST не существует:

```
restore db prod incremental automatic taken at ts2 into test without prompting
DB20000I  Команда RESTORE DATABASE выполнена успешно.
```

Утилита восстановления создаст и заполнит базу данных TEST.

Если база данных TEST существует, и ее хронология содержит записи, то перед выполнением автоматического инкрементного восстановления эту базу данных нужно отбросить:

```
drop db test
DB20000I  Команда DROP DATABASE выполнена успешно.

restore db prod incremental automatic taken at ts2 into test without prompting
DB20000I  Команда RESTORE DATABASE выполнена успешно.
```

Если вы не хотите отбрасывать базу данных, то перед вызовом команды RESTORE DATABASE вызовите команду PRUNE HISTORY, указав еще не наступивший момент времени и параметр WITH FORCE OPTION:

```
connect to test
Информация о соединении с базой данных

Сервер базы данных    = <id_сервера>
ID авторизации SQL          = <id>
Алиас локальной базы данных= TEST

prune history 9999 with force option
DB20000I  Команда PRUNE выполнена успешно.

connect reset
DB20000I  Команда SQL выполнена успешно.
restore db prod incremental automatic taken at ts2 into test without prompting
SQL2540W  Restore is successful, however a warning "2539" was
encountered during Database Restore while processing in No
Interrupt mode.
```

В этом случае команда RESTORE DATABASE COMMAND будет выполняться так же, как если бы база данных TEST не существовала.

Если база данных TEST существует, и в ее хронологии нет записей, то перед автоматическим инкрементным восстановлением отбрасывать эту базу данных не нужно:

```
restore db prod incremental automatic taken at ts2 into test without prompting
SQL2540W  Restore is successful, however a warning "2539" was
encountered during Database Restore while processing in No
Interrupt mode.
```

Вы можете создать инкрементную или разностную резервную копию тестовой базы данных, предварительно не создавая полную резервную копию. Однако если вам потребуется восстановить инкрементную или разностную резервную копию, это нужно будет делать вручную. Это связано с тем, что все резервные копии, обрабатываемые во время автоматического инкрементного восстановления, должны быть созданы на основе базы данных с одним и тем же алиасом.

Если вы создадите полную резервную копию тестовой базы данных после ее восстановления из резервной копии рабочей базы данных, то вы сможете создавать инкрементные или разностные резервные копии и восстанавливать их как вручную, так и автоматически.

Понятия, связанные с данным:

- “Инкрементное резервное копирование и восстановление” на стр. 31

Ссылки, связанные с данной темой:

- “BACKUP DATABASE” на стр. 80
- “RESTORE DATABASE” на стр. 105
- “LIST HISTORY” на стр. 247

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® позволяет добавлять, изменять и удалять контейнеры табличного пространства.

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. Команда восстановления инкрементной резервной копии с перенаправлением аналогична команде восстановления неинкрементной резервной копии: Вызовите команду RESTORE DATABASE с параметром REDIRECT и укажите образ резервной копии для инкрементного восстановления базы данных.

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

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.

Ссылки, связанные с данной темой:

- “RESTORE DATABASE” на стр. 105
- “Сеансы восстановления - Примеры для CLP” на стр. 126

Примеры, связанные с данной темой:

- “dbrecov.out -- HOW TO RECOVER A DATABASE (C)”
- “dbrecov.sqc -- How to recover a database (C)”
- “dbrecov.out -- HOW TO RECOVER A DATABASE (C++)”
- “dbrecov.sqC -- How to recover a database (C++)”

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. 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 и в нем есть записи. 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. Если вы не создадите новую базу данных, это сделает утилита восстановления.

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.

RESTORE DATABASE

Заново создает поврежденную базу данных, для которой утилитами DB2 была создана резервная копия. Восстановленная база данных будет в том же состоянии, в котором она находилась на момент создания резервной копии. В дополнение к восстановлению в новую базу данных эта утилита также позволяет восстанавливать резервную копию в базу данных с другим именем.

Данная утилита может применяться для восстановления баз данных из резервных копий, созданных в двух предыдущих версиях DB2. Если необходима перенастройка, то она автоматически будет выполнена после восстановления.

Если во время резервного копирования для базы данных было разрешено восстановление с повтором транзакций, то базу данных можно перевести в состояние, в котором она находилась до момента сбоя или повреждения. Для этого после успешного восстановления необходимо запустить утилиту повтора транзакций.

Данная утилита позволяет также выполнить восстановление из резервной копии уровня табличного пространства.

Для восстановления базы данных, созданной на другой платформе, следует воспользоваться утилитой db2move. Нельзя непосредственно восстановить базу данных из резервной копии, созданной на другой платформе. При этом поддерживаемые вервии Microsoft Windows считаются эквивалентными.

Область:

Эта команда применяется только к узлу, на котором она выполняется.

Авторизация:

Для восстановления в существующей базе данных необходимы одни из следующих прав доступа:

- *sysadm*
- *sysctrl*
- *sysmaint*

Для восстановления в новой базе данных необходимы одни из следующих прав доступа:

- *sysadm*
- *sysctrl*

Необходимое соединение:

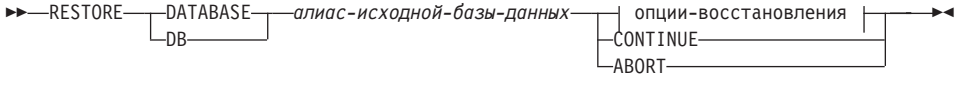
С базой данных - для восстановления в существующей базе данных. Эта команда автоматически устанавливает соединение с указанной базой данных.

С экземпляром и базой данных - для восстановления в новую базу данных. Для создания базы данных требуется подключение к экземпляру.

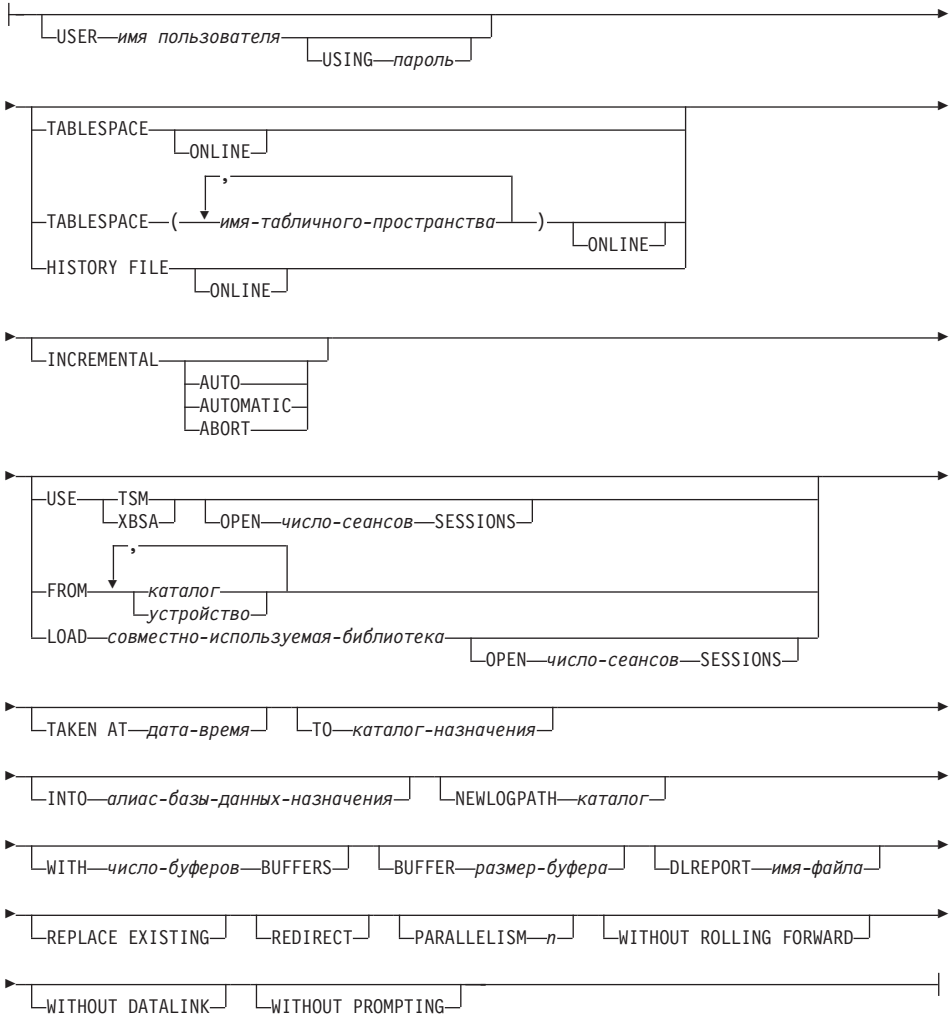
Для создания в новой базе данных экземпляра, отличающегося от текущего (в соответствии со значением переменной среды **DB2INSTANCE**) необходимо сначала подключиться к экземпляру, в котором будет находиться новая база данных.

Для восстановления в новой *удаленной* базе данных необходимо сначала подключиться к экземпляру, в котором будет находиться новая база данных.

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
Без дополнительных параметров INCREMENTAL задает операцию кумулятивного восстановления вручную. Во время восстановления вручную пользователь должен вручную вводить все команды восстановления для каждого образа. При этом следует соблюдать следующий порядок следования образов: последний, первый, второй, третий и так далее до последнего.

INCREMENTAL AUTOMATIC/AUTO
Задает операцию автоматического кумулятивного восстановления.

INCREMENTAL ABORT
Прерывает выполняющееся кумулятивное восстановление вручную.

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

Указывает, что должен применяться 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 - это менеджер памяти, поддерживающий в настоящее время интерфейс XBSA.

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 указанный каталог должен находиться вне дерева каталогов 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. Допустимые варианты ответа:

c Продолжить. Продолжит работу с устройством, выдающим предупреждающие сообщения (например, продолжить после монтирования новой ленты).

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

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

LOAD совместно-используемая-библиотека

Имя библиотеки (в Windows - DLL), содержащей разработанные независимым поставщиком функции ввода-вывода для 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

Метка 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 *ггггммддччммсс*. 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. Диск и каталог должны быть локальными.

Примечание: В операционных системах Windows при указании этого параметра необходимо указать букву устройства. Например, вы можете указать `x:\`*`каталог`* для восстановления в указанном каталоге, или `x:`, если каталог указывать не нужно. Если задано слишком длинное имя каталога, то будет возвращено сообщение об ошибке.

INTO алиас-базы-данных-назначения

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

При восстановлении в существующей базе данных восстанавливаемая база данных наследует алиас и имя существующей базы данных. При восстановлении в новой базе данных созданной базе данных присваиваются указанные вами алиас и имя. Имя новой базы данных должно быть уникальным в пределах системы, в которой выполняется восстановление.

NEWLOGPATH каталог

Абсолютный путь к каталогу, который будет применяться для активных файлов журнала после 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 применяемых 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 резервного копирования. Если задан неправильный размер буфера, будут использоваться буферы наименьшего приемлемого размера.

При использовании лентопротяжных устройств в SCO UnixWare 7 укажите размер буфера 16.

DLREPORT имя-файла

Имя файла, если оно указано, должно быть задано с абсолютным путем. В результате слишком быстрого согласования при восстановлении файлы может быть утрачена связь в файлами. Эта опция должна применяться только в том случае, если восстанавливаемая таблица имеет столбец 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 восстановления база данных находится в состоянии ожидания повтора транзакций, то перед началом работы с этой базой данных необходимо выполнить команду ROLLFORWARD.

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.

Примеры:

В следующем примере база данных WSDB определена на всех 4 разделах с номерами от 0 до 3. У всех разделов есть доступ к каталогу /dev3/backup . В /dev3/backup есть следующие образы автономной резервной копии:

```
wsdb.0.db2inst1.NODE0000.CATN0000.20020331234149.001
wsdb.0.db2inst1.NODE0001.CATN0000.20020331234427.001
wsdb.0.db2inst1.NODE0002.CATN0000.20020331234828.001
wsdb.0.db2inst1.NODE0003.CATN0000.20020331235235.001
```

Сначала необходимо восстановить раздел каталога, затем все остальные разделы базы данных WSDB из каталога /dev3/backup. Для этого введите следующие команды в одном из разделов базы данных:

```
db2_all '<<+0< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234149
  INTO wsdb REPLACE EXISTING'
db2_all '<<+1< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234427
  INTO wsdb REPLACE EXISTING'
db2_all '<<+2< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234828
```

```
  INTO wsdb REPLACE EXISTING'
db2_all '<<+3< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331235235
  INTO wsdb REPLACE EXISTING'
```

Утилита db2_all запустит команду restore для всех перечисленных разделов базы данных.

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:

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

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

```
restore db mydb incremental automatic taken at (Fri)
```

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

```
restore db mydb incremental taken at (Fri)
restore db mydb incremental taken at (Sun)
restore db mydb incremental taken at (Wed)
restore db mydb incremental taken at (Thu)
restore db mydb incremental taken at (Fri)
```

Примечания по использованию:

Команда RESTORE DATABASE в формате `db2 restore db <имя>` выполняет полное восстановление базы данных, независимо от того, является ли восстанавливаемый образ образом базы данных или табличного пространства. Команда RESTORE DATABASE в формате `db2 restore db <имя> tablespace` восстанавливает табличное пространство из образа. Любая команда RESTORE DATABASE, в которой перечислен список табличных пространств, восстанавливает перечисленные табличные пространства.

Ссылки, связанные с данной темой:

- “BACKUP DATABASE” на стр. 80
- “ROLLFORWARD DATABASE” на стр. 146
- “db2move - Database Movement Tool Command” в *Command Reference*

db2Restore - Восстановление базы данных

Повторно создает поврежденную базу данных, которая была ранее сохранена командой db2Backup - Резервное копирование базы данных. Она восстанавливается до состояния, предшествующего созданию резервной копии. Помимо восстановления новой базы данных, эта утилита также может восстановить базу данных под другим именем.

Эта утилита может восстанавливать резервные копии DB2, созданные в двух предыдущих версиях.

Кроме того, эта утилита может восстанавливать резервные копии табличных пространств.

Область:

Этот API влияет только на разделы базы данных, в которых он вызван.

Авторизация:

Для восстановления существующей базы данных вам потребуются следующие права доступа:

- *sysadm*
- *sysctrl*
- *sysmaint*

Для восстановления новой базы данных вам потребуются следующие права доступа:

- *sysadm*
- *sysctrl*

Необходимое соединение:

С базой данных, при восстановлении в существующую базу данных. Этот API автоматически устанавливает соединение с указанной базой данных и прерывает его после завершения операции.

С экземпляром и базой данных, при восстановлении новой базы данных. Для создания базы данных требуется подключение к экземпляру.

Для восстановления базы данных в новом экземпляре (в соответствии со значением переменной среды DB2INSTANCE) необходимо сначала подключиться к экземпляру, в котором будет находиться новая база данных.

Включаемый файл API:

db2ApiDf.h

Синтаксис C API:

```
/* File: db2ApiDf.h */
/* API: db2Restore */
/* ... */
SQL_API_RC SQL_API_FN
db2Restore (
  db2Uint32 versionNumber,
  void         *pDB2RestoreStruct,
  struct sqlca *pSqlca);
/* ... */

typedef SQL_STRUCTURE db2RestoreStruct
{
  char                        *piSourceDBAlias;
  char                        *piTargetDBAlias;
  char                        oApplicationId[SQLU_APPLID_LEN+1];
  char                        *piTimestamp;
  char                        *piTargetDBPath;
  char                        *piReportFile;
  struct db2TablespaceStruct  *piTablespaceList;
  struct db2MediaListStruct   *piMediaList;
  char                        *piUsername;
  char                *piPassword;
  char                        *piNewLogPath;
  void                        *piVendorOptions;
  db2Uint32                   iVendorOptionsSize;
  db2Uint32                   iParallelism;
  db2Uint32                   iBufferSize;
  db2Uint32                   iNumBuffers;
  db2Uint32                   iCallerAction;
  db2Uint32 iOptions;
} db2BackupStruct;

typedef SQL_STRUCTURE db2TablespaceStruct
{
  char                        **tablespaces;
  db2Uint32                   numTablespaces;
} db2TablespaceStruct;

typedef SQL_STRUCTURE db2MediaListStruct
{
  char                        **locations;
  db2Uint32                   numLocations;
  char                        locationType;
} db2MediaListStruct;
/* ... */
```

Общий синтаксис API:

```
/* Файл: db2ApiDf.h */
/* API: db2gRestore */
/* ... */
SQL_API_RC SQL_API_FN
db2gRestore (
  db2Uint32 versionNumber,
  void         *pDB2gRestoreStruct,
```

```
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2gRestoreStruct
{
  char                         *piSourceDBAlias;
  db2Uint32                    iSourceDBAliasLen;
  char                         *piTargetDBAlias;
  db2Uint32                    iTargetDBAliasLen;
  char                         *poApplicationId;
  db2Uint32                    iApplicationIdLen;
  char                         *piTimestamp;
  db2Uint32                    iTimestampLen;
  char                         *piTargetDBPath;
  db2Uint32                    iTargetDBPathLen;
  char                         *piReportFile;
  db2Uint32                    iReportFileLen;
  struct db2gTablespaceStruct *piTablespaceList;
  struct db2gMediaListStruct  *piMediaList;
  char                         *piUsername;
  db2Uint32                    iUsernameLen;
  char                *piPassword;
  db2Uint32 iPasswordLen;
  char                         *piNewLogPath;
  db2Uint32                    iNewLogPathLen;
  void                         *piVendorOptions;
  db2Uint32                    iVendorOptionsSize;
  db2Uint32                    iParallelism;
  db2Uint32                    iBufferSize;
  db2Uint32                    iNumBuffers;
  db2Uint32                    iCallerAction;
  db2Uint32 iOptions;
} db2gBackupStruct;

typedef SQL_STRUCTURE db2gTablespaceStruct
{
  struct db2Char              *tablespaces;
  db2Uint32                    numTablespaces;
} db2gTablespaceStruct;

typedef SQL_STRUCTURE db2gMediaListStruct
{
  struct db2Char              *locations;
  db2Uint32                    numLocations;
  char                         locationType;
} db2gMediaListStruct;

typedef SQL_STRUCTURE db2Char
{
   char            *pioData;
   db2Uint32       iLength;
   db2Uint32       oLength;
} db2Char;
/* ... */
```

Параметры API:

versionNumber
Входной. Задает уровень версии и выпуска для структуры, передаваемой во втором параметре - pParamStruct.

pDB2RestoreStruct
Входной. Указатель на структуру *db2RestoreStruct*.

pSqlca Выходной. Указатель на структуры *sqlca*.

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

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

piTargetDBAlias
Входной. Строка, содержащая алиас базы данных назначения. Если этот параметр пуст, то будет применяться *piSourceDBAlias*.

iTargetDBAliasLen
Входной. 4-байтное целое без знака, равное длине (в байтах) алиаса целевой базы данных.

oApplicationId
Выходной. 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.

poApplicationId
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.

iApplicationIdLen
Входной. 4-байтное целое без знака, задающее длину (в байтах) буфера poApplicationId. Должно быть равно SQLU_APPLID_LEN+1 (определено в `sqlutil`).

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

iTimestampLen
Входной. 4-байтное целое без знака, задающее длину (в байтах) буфера piTimestamp.

piTargetDBPath
Входной. Строка, содержащая относительное или полное имя каталога базы данных назначения на сервере. Используется, если во время операции восстановления требуется создать новую базу данных, в противном случае игнорируется.

piReportFile
Входной. Если задано имя файла, оно должно быть полным. Сообщает о файлах, которые остались не связанными в результате быстрого согласования во время операции восстановления.

iReportFileLen
Входной. 4-байтное целое без знака, задающее длину (в байтах) буфера piReportFile.

piTablespaceList
Входной. Список восстанавливаемых табличных пространств. Применяется при восстановлении подмножества табличных пространств из образа резервной копии базы данных или табличного пространства. См. структуру *DB2TablespaceStruct*. Существуют следующие ограничения:

- База данных должна быть восстановимой, то есть должно быть включено сохранение журналов или обработчики пользователя.
- 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.
- Утилита повтора транзакций обеспечивает синхронизацию табличных пространств, восстановленных в другом разделе базы данных, содержащем те же табличные пространства. Если запрошено восстановление табличного пространства, и значение *piTablespaceList* равно NULL, то утилита восстановления попытается восстановить все табличные пространства из образа резервной копии.

При восстановлении табличного пространства, которое было переименовано после создания резервной копии, нужно использовать при вызове команды восстановления. Если используется старое имя табличного пространства, оно не будет найдено.

piMediaList
Входной. Исходные носители для резервной копии. Передаваемая информация зависит от значения поля locationType. Допустимые значения locationType (определены в `sqlutil`:

SQLU_LOCAL_MEDIA
Локальные устройства (сочетание лент, дисков или дискет).

SQLU_TSM_MEDIA
TSM. Если указатель равен NULL, то применяется общая

библиотека TSM, поставляемая вместе с DB2. 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 поле Положение.

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

piUsername
Входной. Строка с именем пользователя, используемым при установке соединения. Может иметь пустое значение (NULL).

iUsernameLen
Входной. 4-байтное целое без знака, задающее длину (в байтах) piUsername. Задайте значение ноль, если имя пользователя не указано.

piPassword
Входной. Строка с паролем, используемым с этим именем пользователя. Может иметь пустое значение (NULL).

iPasswordLen
Входной. 4-байтное целое без знака, задающее длину (в байтах) piPassword. Задайте значение ноль, если пароль не указан.

piNewLogPath
Входной. Строка, представляющая путь, который будет применяться для журнала после того, как восстановление будет завершено. Если это поле пусто, то будет применяться путь по умолчанию.

iNewLogPathLen
Входной. 4-байтное целое без знака, задающее длину (в байтах) *piNewLogPath*.

piVendorOptions
Входной. Используется для передачи информации от прикладной программы к функции 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.

iVendorOptionsSize
Входной. Длина *piVendorOptions*; не должна превышать 65535 байт.

iParallelism

Входной. Степень параллелизма (число манипуляторов буферов). Минимальное значение - 1. Максимальное - 1024. Значение по умолчанию - 1.

iBufferSize

Входной. Размер буфера резервного копирования в страницах по 4 Кбайта. Минимальное значение - 8 страниц. Значение по умолчанию - 1024 страницы. Заданный размер буфера должен быть равен размеру буфера для резервной копии или кратен ему.

iNumBuffers

Входной. Задает число используемых буферов восстановления.

iCallerAction

Входной. Задает выполняемое действие. Допустимые значения (заданы в `db2ApiDf`):

DB2RESTORE_RESTORE

Начать операцию восстановления.

DB2RESTORE_NOINTERRUPT

Начать операцию восстановления. Указывает, что восстановление будет запущено автономно, а для сценариев, в которых обычно требуется вмешательство пользователя, будет предпринята попытка выполнения без возвращения к вызывающей программе, либо будет выдана ошибка. Это значение параметра CallerAction можно использовать, например, когда вы знаете, что все необходимые носители для операции восстановления смонтированы, а подсказки утилиты нежелательны.

DB2RESTORE_CONTINUE

Продолжить операцию восстановления после того, как пользователь выполнил действия, запрошенные утилитой восстановления (например, смонтировать новую ленту).

DB2RESTORE_TERMINATE

Прервать операцию восстановления после того, как пользователь не смог выполнить действия, запрошенные утилитой.

DB2RESTORE_DEVICE_TERMINATE

Удалить конкретное устройство из списка устройств, используемых для восстановления. Когда входные данные на устройстве исчерпаны, утилита восстановления возвращает предупреждение вызывающей программе. Снова запустите утилиту восстановления с этим значением параметра CallerAction, чтобы удалить устройство, для которого выдано такое предупреждение, из списка используемых устройств.

DB2RESTORE_PARM_CHK
Используется для проверки параметров без выполнения операции восстановления. 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 DB2RESTORE_CONTINUE для выполнения действия.

DB2RESTORE_PARM_CHK_ONLY
Используется для проверки параметров без выполнения операции восстановления. Перед возвращением из вызова завершается установленное этим вызовом соединение с базой данных и последующие вызовы не требуются.

DB2RESTORE_TERMINATE_INCRE
Прервать инкрементную операцию восстановления до ее завершения.

DB2RESTORE_RESTORE_STORDEF
Начальный вызов. Запрошено переопределение контейнера табличного пространства.

DB2RESTORE_STORDEF_NOINTERRUPT
Начальный вызов. Процесс восстановления будет выполняться без прерываний. Запрошено переопределение контейнера табличного пространства.

iOptions
Входной. Способ объединения параметров восстановления. Для получения значения *iOptions* опции комбинируются с помощью побитового оператора OR. Допустимые значения (заданы в `db2ApiDf`):

DB2RESTORE_OFFLINE
Выполнить операцию восстановления автономно.

DB2RESTORE_ONLINE
Выполнить операцию восстановления оперативно.

DB2RESTORE_DB
Восстановить все табличные пространства в базе данных. Это необходимо сделать в автономном режиме.

DB2RESTORE_TABLESPACE
Восстановить из резервной копии только табличные пространства, перечисленные в параметре *piTablespaceList*. Может выполняться в оперативном или автономном режиме.

DB2RESTORE_HISTORY
Восстанавливать только файл хронологии.

DB2RESTORE_INCREMENTAL
Выполнить ручную инкрементную операцию восстановления.

DB2RESTORE_AUTOMATIC
Выполнить автоматическую инкрементную операцию восстановления. Должно быть указано вместе с DB2RESTORE_INCREMENTAL.

DB2RESTORE_DATALINK
Выполнить операции согласования. В таблицах, где определен столбец DATALINK, должна быть задана опция RECOVERY YES.

DB2RESTORE_NODATALINK
Не выполнять операции согласования. Все таблицы со столбцами DATALINK переводятся в состояние DataLink_Roconcile_pending (DRP). В таблицах, где определен столбец DATALINK, должна быть задана опция RECOVERY YES.

DB2RESTORE_ROLLFWD
Оставлять базу данных в состоянии ожидания повтора транзакций после ее успешного восстановления.

DB2RESTORE_NOROLLFWD
Не оставлять базу данных в состоянии ожидания повтора транзакций после ее успешного восстановления. Это значение нельзя указывать для оперативных резервных копирований или для восстановления табличных пространств. Если после успешной операцией восстановления база данных находится в состоянии ожидания повтора транзакций, то перед использованием базы данных для нее необходимо выполнить команду db2Rollforward - Повторить транзакции.

tablespaces
Указатель на список табличных пространств, для которых нужно сделать резервные копии. В программах C список должен быть строкой, завершающейся символом NULL. В общем случае, это список структур *db2Char*.

numTablespaces
Число записей в списке *tablespaces*.

locations
Указатель на список положений носителей. В программах C список должен быть строкой, завершающейся символом NULL. В общем случае, это список структур db2Char.

numLocations
Число записей в списке *locations*.

locationType
Символ, обозначающий тип носителя. Допустимые значения (определенные в `sqlutil`):

SQLU_LOCAL_MEDIA
Локальные устройства (ленты, диски, дискеты или именованные конвейеры).

SQLU_TSM_MEDIA
Tivoli Storage Manager.

SQLU_OTHER_MEDIA
Библиотека другого поставщика.

SQLU_USER_EXIT
Обработчик пользователя (доступен только в OS/2).

pioData
Указатель на буфер символьных данных.

iLength
Входной. Размер буфера *pioData*.

oLength
Выходной. Зарезервирован для будущего использования.

Замечания по использованию:

При автономном восстановлении эта утилита устанавливает соединение с базой данных в монопольном режиме. Утилита не будет выполнена, если какая-либо программа, включая вызываемую, уже подключена к восстанавливаемой базе данных. Кроме того, запрос не будет выполнен, если утилита выполняет операцию восстановления, и какая-либо программа, включая вызываемую, уже подключена к любой базе данных на той же рабочей станции. При успешном подключении API блокирует доступ другим программам до тех пор, пока восстановление не будет завершено.

Текущий файл конфигурации базы данных будет заменен резервной копией только в том случае, если он поврежден. При замене файла отправляется подтверждающее сообщение.

Резервная копия базы данных или табличного пространства должна была быть сделана с помощью db2Backup - Создать резервную копию базы данных.

Если вызвано действие DB2RESTORE_NOINTERRUPT, то восстановление будет продолжено без выдачи дополнительных запросов программе. Если вызвано действие DB2RESTORE_RESTORE, а данные восстанавливаются в существующую базу данных, то утилита вернет управление вызывающей программе и потребует у нее вмешательства пользователя. После обработки

ответа пользователя программа снова вызовет RESTORE DATABASE, указав действие DB2RESTORE_CONTINUE (продолжить восстановление) или DB2RESTORE_TERMINATE (прервать восстановление). Утилита завершит работу, возвратив SQLCODE в параметре *sqlca*.

Для того чтобы закрыть устройство после выполнения операции, укажите действие DB2RESTORE_DEVICE_TERMINATE. Например, если пользователь восстанавливает из трех томов с магнитной лентой с помощью двух устройств, и одна лента уже восстановлена, то программа получит управление от API с SQLCODE, указывающим код ошибки. Программа может попросить пользователя вставить другую ленту, а если пользователь укажет, что лент больше нет, то вернуть управление API с действием SQLUD_DEVICE_TERMINATE, указывающим конец носителя. Операция с этим устройством будет прекращена, а остальные устройства продолжат обработку восстанавливаемых наборов данных (число сегментов в наборе указывается при сохранении на последнем носителе). Эти действия могут применяться и для поддерживаемых устройств других фирм.

Для проверки параметров перед возвратом в приложение укажите действие DB2RESTORE_PARM_CHK.

При перенаправлении укажите действие DB2RESTORE_RESTORE_STORDEF; оно применяется в сочетании с sqlbstsc - Задать контейнеры табличных пространств.

Если во время восстановления базы данных возникнет ошибка, то успешно связаться с базой данных удастся только после успешного восстановления. В этом случае при попытке обращения к базе данных появится сообщение об ошибке. Если в сохраненной базе данных не настроено восстановление с повтором транзакций, и есть допустимый файл конфигурации, в котором указан один из этих параметров, то после восстановления пользователю придется либо создать новую резервную копию базы данных, либо отключить параметры сохранения журналов и пользовательских обработчиков перед подключением к базе данных.

Хотя восстановленная таблица и не будет отброшена (кроме восстановления в несуществующую базу данных), в случае сбоя восстановления использовать эту базу данных будет нельзя.

Если указано, что должен быть восстановлен файл хронологии, то он заменит существующий файл, уничтожив тем самым информацию обо всех изменениях, внесенных после сохранения. Если это нежелательно, то восстановите файл хронологии в новую или тестовую базу данных, не уничтожая существующий файл.

Если при сохранении был включен повтор транзакций, то базу данных можно привести в состояние, непосредственно предшествовавшее ошибке или сбою, с помощью команды db2Rollforward (после завершения db2Restore). Если база данных восстановима, то после восстановления она будет приведена в состояние ожидания повтора транзакций.

Если база данных была сохранена в автономном режиме, и после восстановления для нее не требуется выполнять повтор транзакций, то для восстановления можно применять опцию DB2RESTORE_NOROLLFWD. В этом случае база данных станет доступной сразу после восстановления Если сохранение было сделано в оперативном режиме, то после завершения восстановления для базы данных нужно обязательно выполнить повтор транзакций.

Примеры, связанные с данной темой:

- “dbrecov.sqc -- How to recover a database (C)”
- “dbrecov.sqC -- How to recover a database (C++)”

Сеансы восстановления - Примеры для CLP

Пример 1

Ниже приведен стандартный сценарий неинкрементного восстановления с перенаправлением для базы данных с алиасом MYDB:

1. Введите команду RESTORE DATABASE с опцией REDIRECT.

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

2. Вызовите команду SET TABLESPACE CONTAINERS для каждого табличного пространства, чьи контейнеры требуется переопределить. Например, в среде Windows:

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

   Для того чтобы проверить, что контейнеры восстановленной базы данных - именно те, которые заданы на этом шаге, вызовите команду LIST TABLESPACE CONTAINERS для каждого табличного пространства, расположение контейнеров которого было переопределено.

3. После успешного выполнения 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.

Примечания:

1. После успешного выполнения шага 1 и до выполнения шага 3 операцию восстановления можно прервать с помощью следующей команды:

   ```
   db2 restore db mydb abort
   ```

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

Пример 2

Ниже приведен стандартный сценарий инкрементного восстановления с перенаправлением для базы данных с алиасом MYDB и следующими образами резервной копии:

```
backup db mydb
Резервное копирование выполнено. Отметка времени копии: <ts1>

backup db mydb incremental
Резервное копирование выполнено. Отметка времени копии: <ts2>
```

1. Вызовите команду RESTORE DATABASE с опциями INCREMENTAL и REDIRECT.

   ```
   db2 restore db mydb incremental taken at <ts2> replace existing redirect
   ```

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

   ```
   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 выполнения шагов 1 и 2 введите команду:

   ```
   db2 restore db mydb continue
   ```

4. Теперь можно вызвать остальные команды инкрементного восстановления:

   ```
   db2 restore db mydb incremental taken at <ts1>
   db2 restore db mydb incremental taken at <ts2>
   ```

   Это завершающий шаг операции перенаправленного восстановления.

Примечания:

1. После успешного выполнения шага 1 и до выполнения шага 3 операцию восстановления можно прервать с помощью следующей команды:

   ```
   db2 restore db mydb abort
   ```

2. После успешного выполнения шага 3 и до вызова команд на шаге 4 операцию восстановления можно прервать с помощью следующей команды:

   ```
   db2 restore db mydb incremental abort
   ```

3. Если шаг 3 завершается неудачно, или если операция восстановления была прервана, перенаправленное восстановление можно перезапустить с шага 1.
4. Если при выполнении одной из команд восстановления на шаге 4 возникла ошибка, эту команду можно вызвать еще раз, чтобы продолжить процесс восстановления.

Пример 3

Ниже приведен стандартный сценарий автоматического инкрементного восстановления с перенаправлением для той же базы данных:

1. Вызовите команду RESTORE DATABASE с опциями INCREMENTAL AUTOMATIC и REDIRECT.

   ```
   db2 restore db mydb incremental automatic taken at <ts2>
          replace existing redirect
   ```

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

   ```
   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 выполнения шагов 1 и 2 введите команду:

   ```
   db2 restore db mydb continue
   ```

   Это завершающий шаг операции перенаправленного восстановления.

Примечания:

1. После успешного выполнения шага 1 и до выполнения шага 3 операцию восстановления можно прервать с помощью следующей команды:

   ```
   db2 restore db mydb abort
   ```

2. Если шаг 3 завершился неудачно, либо операция восстановления была прервана, то операцию восстановления с перенаправлением можно выполнить заново, начиная с шага 1, вызвав следующую команду:

   ```
   db2 restore db mydb incremental abort
   ```

Ссылки, связанные с данной темой:

- “RESTORE DATABASE” на стр. 105
- “LIST TABLESPACE CONTAINERS Command” в *Command Reference*
- “SET TABLESPACE CONTAINERS Command” в *Command Reference*

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.

В этом приложении рассматриваются следующие темы:

Обзор повтора транзакций

В простейшем варианте команды DB2® ROLLFORWARD DATABASE нужно задать только алиас базы данных, для которой нужно выполнить восстановление с повтором транзакций. Например:

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

В этом примере команда возвращает:

```
                                    Состояние повтора транзакций

 Алиас входной базы данных          = sample
 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          = 2001-03-11-02.39.48.000000

DB20000I  Команда ROLLFORWARD выполнена успешно.
```

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 выполнить нельзя. Можно выполнить только восстановление после сбоя или восстановление версии. 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.

Понятия, связанные с данным:

- “Использование файла положения копии загрузки” на стр. 142
- “Что такое журналы восстановления” на стр. 38

Ссылки, связанные с данной темой:

- “ROLLFORWARD DATABASE” на стр. 146
- “Параметры конфигурации для записи в журнал базы данных” на стр. 44

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

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

Предварительные требования для установки:

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 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.

База данных может быть как локальной, так и удаленной.

Ограничения:

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.

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

Процедура:

Утилиту повтора транзакций можно вызвать с помощью процессора командной строки (CLP), окна Повтор транзакций для базы данных Центра управления или API **db2Rollforward**.

Пример команды ROLLFORWARD DATABASE, введенной в командной строке:

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

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.

Подробную информацию можно найти в электронной справке Центра управления.

Понятия, связанные с данным:

- “Administrative APIs in Embedded SQL or DB2 CLI Programs” в *Application Development Guide: Programming Client Applications*

- “Встраиваемые модули Центра управления - Введение” в *Руководство администратора: Реализация*

Ссылки, связанные с данной темой:

- “db2Rollforward - Повтор транзакций базы данных” на стр. 158

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.

При выполнении повтора транзакций для табличного пространства DB2® пропускает файлы, которые гарантированно не содержат записей журнала, связанных с табличным пространством. Для того чтобы были обработаны все файлы журнала, присвойте переменной реестра DB2_COLLECT_TS_REC_INFO значение false.

В файле хронологии изменений табличного пространства (DB2TSCHG.HIS), расположенном в каталоге базе данных, содержится информация о том, какие журналы нужно обработать для каждого табличного пространства. Вы можете просмотреть содержимое этого файла с помощью утилиты **db2logsForRfwd** и удалить из него записи с помощью команды PRUNE HISTORY. Во время восстановления базы данных файл DB2TSCHG.HIS восстанавливается из резервной копии и обновляется в соответствии с текущим состоянием базы данных во время повтора транзакций. Если информация о файле журнала отсутствует, то считается, что этот файл необходим для восстановления всех табличных пространств.

Поскольку информация о файле журнала выгружается на диск после того, как журнал становится неактивным, эта информация может быть утеряна в результате сбоя. Для того чтобы компенсировать эту потерю, во время операций восстановления, начинающихся с середины файла журнала, считается, что во всем журнале содержится информация об изменении всех табличных пространств системы. После этого будут обработаны активные журналы и будет воссоздана информация для этих журналов. Если информация о старых или архивных файлах журнала была утеряна в результате сбоя, и сведения об этих файлах отсутствуют в файле данных, то во время восстановления табличного пространства считается, что эти файлы содержат сведения об изменении всех табличных пространств.

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 данные таблицы и объекты типа long хранятся в разных табличных пространствах, и данные объектов типа long были реорганизованы, то оба вида табличных пространств должны быть совместно восстановлены и для них, также совместно, должен быть выполнен 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 материализованной таблицы запроса или промежуточной таблицы
- Материализованной таблицей запроса или промежуточной таблицей для таблицы из другого табличного пространства

Необходимо повторить транзакции для обоих табличных пространств до одного и того же момента времени. В противном случае материализованная таблица запроса или промежуточная таблица будет переведена в состояние отложенной проверки по окончании операции повтора транзакций. Материализованную таблицу запроса потребуется полностью обновить, а промежуточная таблица будет помечена как неполная.

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. В противном случае по окончании операции повтора транзакций дочерняя таблица во взаимоотношении реляционной целостности будет переведена в состояние отложенной проверки. Когда впоследствии дочерняя таблица будет проверяться на отсутствие нарушений ограничений, будет выполнена проверка всей таблицы. Вместе с дочерней таблицей в состояние отложенной проверки будут переведены следующие таблицы, если они существуют:

- Все дочерние материализованные таблицы запросов для исходной дочерней таблицы
- Все дочерние промежуточные таблицы для исходной дочерней таблицы
- Все дочерние таблицы внешних ключей для исходной дочерней таблицы

Для того чтобы вывести эти таблицы из состояния отложенной проверки, их необходимо полностью обработать. 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.)

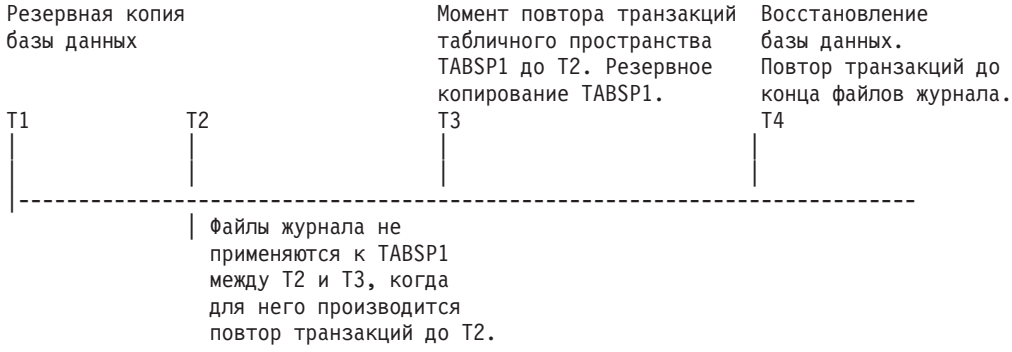


Рисунок 15. Требования к резервной копии табличного пространства

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.

Понятия, связанные с данным:

- “Использование файла положения копии загрузки” на стр. 142

Ссылки, связанные с данной темой:

- “ROLLFORWARD DATABASE” на стр. 146

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.

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.

Предварительные требования:

Чтобы отброшенную таблицу можно было восстановить, у табличного пространства, в котором расположена эта таблица, должна быть включена опция DROPPED TABLE RECOVERY. Эту опцию можно включить во время создания табличного пространства или позднее, при помощи оператора ALTER TABLESPACE. 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. По умолчанию при создании табличных пространств данных опция восстановления отброшенной таблицы включается.

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.

Ограничения:

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 больших табличных пространств. При

попытке восстановить отброшенную таблицу, содержащую столбцы типа большой объект или LONG VARCHAR, в сгенерированном файле экспорта для них будет установлено пустое значение. Опцию DROPPED TABLE RECOVERY можно задать только для обычных табличных пространств, но не для временных и больших табличных пространств.

- 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.

Процедура:

За один раз может быть восстановлена только одна отброшенная таблица. Отброшенную таблицу можно восстановить так:

1. Найдите отброшенную таблицу с помощью команды LIST HISTORY DROPPED TABLE. 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.

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.

Ссылки, связанные с данной темой:

- “ALTER TABLESPACE statement” в *SQL Reference, Том 2*
- “CREATE TABLE statement” в *SQL Reference, Том 2*
- “ROLLFORWARD DATABASE” на стр. 146
- “LIST HISTORY” на стр. 247

Использование файла положения копии загрузки

Переменная реестра DB2LOADREC идентифицирует файл с информацией о расположении загружаемой копии. 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)

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. Для многораздельной базы данных переменная реестра DB2LOADREC должна быть задана для каждого сервера раздела базы данных командой **db2set**.
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, применяемых при
                                       восстановлении
SESsionnumber  NULL                  * Число сеансов, применяемых при
                                       восстановлении
TYPeofmedia    L                     * Тип среды - L для локального устройства
                                                   A для TSM
                                                   O для других поставщиков
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 стоять 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 (0). При типе L требуется число положений, за которыми идут записи положений. Для типа A дополнительного ввода не требуется. Для типа 0 требуется имя совместно используемой библиотеки.
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.

Ссылки, связанные с данной темой:

- Приложение F, “Tivoli Storage Manager” на стр. 341

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 записях журналов DB2® в качестве базиса для временных отметок используется системное время каждого компьютера. Если часы системы переводят вперед, в журнал автоматически будет записано новое время системы. Хотя часы системы можно перевести назад, время в журнале 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* сокращает количество ошибок этого типа:

- Конфигурируемые значения этого параметра лежат в диапазоне от 1 минуты до 24 часов.
- 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, находится ли разница между временем на узле, требующем соединения, и его собственным временем в диапазоне, заданном параметром *max_time_diff*. Если этот диапазон превышен, требование на соединение отклоняется.
- Транзакция изменения, обращающаяся к 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.

Ссылки, связанные с данной темой:

- “Maximum Time Difference Among Nodes configuration parameter - max_time_diff” в *Руководство администратора: Производительность*

Преобразование системного времени клиента на сервере

В этом разделе описано, каким образом в среде клиент-сервер генерируются отметки системного времени:

- Если в операции повтора транзакций было задано местное время, то во всех возвращаемых сообщениях также будет указано местное время.

  Примечание: Все отметки времени преобразуются на сервере, а в среде многораздельной базы данных - и на узле каталога.

- Строка с отметкой времени преобразуется на сервере в GMT, поэтому время будет соответствовать часовому поясу сервера, а не клиента. Если клиент и сервер находятся в разных часовых поясах, то следует применять местное время сервера.

- Если строка с отметкой времени содержит значение, близкое ко времени перехода на летнее или зимнее время, то для того чтобы правильно задать время завершения операции нужно знать, будет ли операция завершена до или после такого перехода.

Понятия, связанные с данным:

- “Обзор повтора транзакций” на стр. 129
- “Синхронизация системного времени в системе многораздельной базы данных” на стр. 144

ROLLFORWARD DATABASE

Восстанавливает базы данных, применяя транзакции, записанные в файлах журнала базы данных. Вызывается после восстановления базы данных или табличного пространства из резервной копии, либо в случае временного отключения табличного пространства базой данных из-за ошибки носителя. Для восстановления с повтором транзакций база данных должна быть восстанавливаемой (т.е. должен быть включен хотя бы один из параметров *logretain* и *userexit*).

Область:

В среде многораздельной базы данных эту команду можно вызывать только из раздела каталога. Восстановление базы данных или табличного пространства с повтором транзакций до определенного момента времени применяется ко всем разделам, перечисленным в файле `db2nodes.cfg`. Восстановление базы данных или табличного пространства с повтором транзакций до конца журналов применяется к указанным разделам. Если разделы не указаны, то операция применяется ко всем разделам, перечисленным в файле `db2nodes.cfg`; если для какого-либо раздела восстановление с повтором транзакций не требуется, то этот раздел игнорируется.

Авторизация:

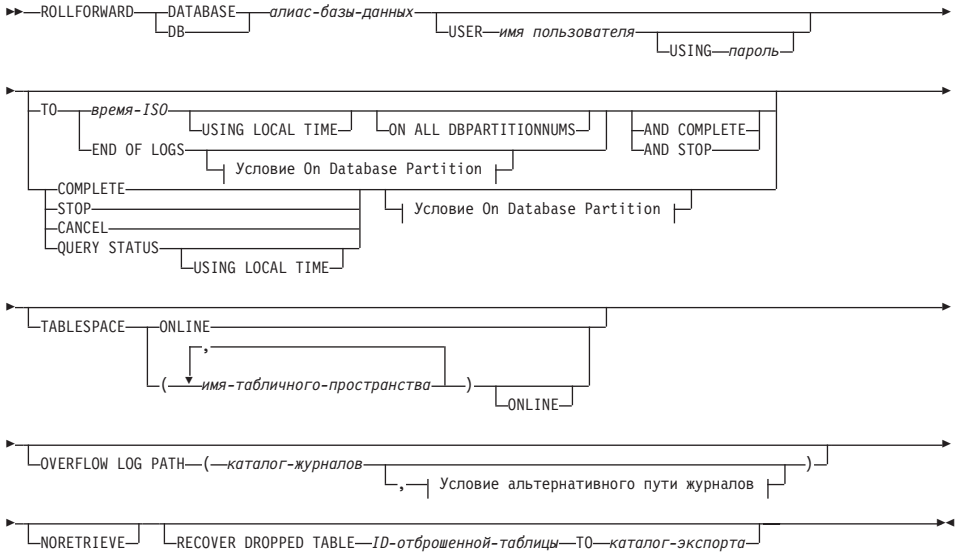
Один из следующих вариантов:

- *sysadm*
- *sysctrl*
- *sysmaint*

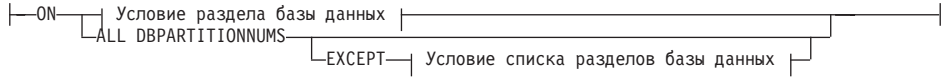
Необходимое соединение:

Нет. Эта команда устанавливает соединение с базой данных.

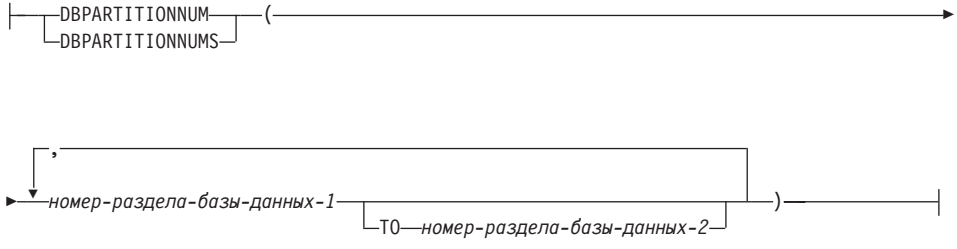
Синтаксис команды:



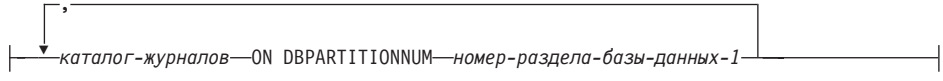
Условие раздела базы данных:



Условие списка разделов базы данных:



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.

USING LOCAL TIME

Разрешает пользователю выполнить повтор транзакций до момента времени, заданного с применением местного времени, а не GMT. Такой подход упрощает повтор транзакций до определенного момента времени в локальных системах и устраняет возможные ошибки пользователей при преобразовании местного времени во время GMTe.

Примечания:

1. Если пользователь указал для повтора транзакций местное время, то во всех возвращаемых сообщениях также будет применяться местное время. Обратите внимание, что все значения времени преобразуются на сервере, а в случае MPP - в разделе каталога базы данных.

2. Строка метки времени преобразуется на сервере в GMT, т.е. применяется местное время часового пояса сервера, а не клиента. Если клиент и сервер находятся в разных часовых поясах, то следует применять местное время сервера. В этом заключается отличие от опции местного времени Центра управления, на котором применяется местное время клиента.
3. Если момент времени близок к моменту перевода часов при переходе на летнее или зимнее время, то важно учесть, на какой момент приходится время завершения операции - до или после перевода часов - и правильно задать время завершения.

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 DBPARTITIONNUMS

Указывает, что необходимо повторить транзакции для всех разделов, перечисленных в файле `db2nodes.cfg`. Это значение по умолчанию, если не указано условие раздела базы данных.

EXCEPT

Указывает, что транзакции необходимо повторить для всех разделов, перечисленных в файле `db2nodes.cfg`, за исключением указанных в списке разделов базы данных.

ON DBPARTITIONNUM / ON DBPARTITIONNUMS

Повторить транзакции для набора разделов базы данных.

раздел-базы-данных-номер-1

Задает номер раздела базы данных в списке разделов.

номер-раздела-базы-данных-2

Задает второй номер раздела базы данных для включения разделов с *номера-раздела-базы-данных-1* по *номер-раздела-базы-данных-2* включительно в список разделов базы данных.

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 табличные пространства, для которых выполнялось восстановление с повтором транзакций, переводятся в состояние ожидания 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 возвращается информация о состоянии каждого раздела. Возвращаемая информация содержит следующие поля:

Номер раздела базы данных

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 не обязательно наличие в каждом восстанавливаемом разделе всех перечисленных в списке табличных пространств. 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 файла журнала по умолчанию *для всех разделов*. Для одораздельных баз данных можно задавать относительный путь.

Примечание: Параметр OVERFLOW LOG PATH переопределяет значение параметра конфигурации базы данных OVERFLOWLOGPATH (если такое значение задано).

каталог журнала ON DBPARTITIONNUM
В среде многораздельной базы данных это полное имя альтернативного файла журнала по умолчанию для выбранного раздела.

NORETRIEVE
Позволяет пользователю управлять выбором файлов журналов, применяемых для повтора транзакций на машине, находящейся в состоянии ожидания, отключая применение архивных журналов. Преимущества такого подхода проявляются в следующих ситуациях:

- Выбор файлов журналов для повтора транзакций позволяет гарантировать, что состояние ожидающей машины соответствует состоянию рабочей машины с отставанием на X часов, не влияя при этом на обе системы.
- Если у ожидающей машины нет доступа к архивам (т.е. если архив TSM разрешает извлекать архивные файлы только исходной машине)
- Также возможна ситуация, что во время архивации файлов в рабочей системе ожидающая машина может извлекать тот же самый файл до его заполнения. Опция Noretrieve позволяет избежать такой ошибки.

RECOVER DROPPED TABLE ID-отброшенной-таблицы
Указывает, что во время операции повтора транзакций нужно восстановить отброшенную таблицу. ID таблицы можно получить при помощи команды LIST HISTORY.

TO каталог-экспорта
Задает каталог, в который нужно записать файлы, содержащие данные этой таблицы. Доступ к каталогу должен быть у всех разделов базы данных.

Examples:

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)

Существует три раздела базы данных: 1, 2 и 3. Табличное пространство TBS1 определено во всех разделах, а табличное пространство TBS2 - в разделах 0 и 2. После восстановления базы данных из резервной копии в разделе 1, а TBS1 в разделах 0 и 2, повторить транзакции базы данных в разделе 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 в разделах 0 и 2 отключены.”).

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

Это приведет к повтору транзакций для TBS1 в разделах 0 и 2. В данном случае условие TABLESPACE(TBS1) - необязательное.

Пример 6 (MPP)

После восстановления табличного пространства TBS1 из резервной копии только в разделах 0 и 2 повторить транзакции для TBS1 в разделах 0 и 2:

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

Раздел 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 в разделе 1. Будет получено сообщение SQL4906N.

```
db2 rollforward db sample to end of logs on dbpartitionnums (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 в разделе 1, а повтор транзакций для всех частей должен проводиться совместно.

Примечание: При повторе транзакций для табличного пространства до определенного момента времени условие для разделов не воспринимается. Операция повтора транзакций должна выполняться во всех разделах, в которых расположено табличное пространство.

После восстановления TBS1 в разделе 1:

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

Эта операция завершится успешно.

Пример 7 (многораздельная база данных)

После восстановления табличного пространства из резервной копии во всех разделах 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 во всех разделах **

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 в восьми разделах (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). Разделы, в которых расположено табличное пространство, указывать не требуется. Утилита по умолчанию использует файл db2nodes.cfg.

Пример 9 (многораздельная база данных)

Восстановить с повтором транзакций шесть небольших табличных пространств, расположенных в одноразделной группе разделов базы данных (в разделе 6):

```
db2 rollforward database dwtest to end of logs on dbpartitionnum (6)
   tablespace(tsstore, tssbuyer, tsstime, tsswhse, tsslscat, tssvendor)
```

Эта операция успешно выполняется до конца журналов (не до определенного момента времени).

Примечания по использованию:

При восстановлении из образов, созданных во время динамического резервного копирования указанный момент времени для повтора транзакций должен быть позже момента завершения динамического резервного копирования. Если повтор транзакций был остановлен до этого момента времени, то база данных останется в состоянии ожидания повтора транзакций. Если повтор транзакций выполняется для табличного пространства, то оно остается в состоянии выполнения повтора транзакций.

Если выполняется повтор транзакций до определенного момента времени для одного или нескольких табличных пространств, то он должен продолжаться по крайней мере до минимального момента восстановления, который соответствует последнему обновлению системных каталогов для этого табличного пространства или входящих в него таблиц. Минимальный момент восстановления (по скоординированному универсальному времени, UTC) для таблицы можно определить при помощи команды LIST TABLESPACES SHOW DETAIL.

Повторение транзакций для баз данных может потребовать восстановления с загрузкой при помощи лентопротяжных устройств. Если вам предложено установить другую ленту, вы можете ответить следующим образом:

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

d Прервать работу устройства. Прекратить работу с устройством, выдающим предупреждающие сообщения (например, когда лент больше нет)

t Прервать. Прервать работу всех устройств.

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.

Совместимость:

Для совместимости с версиями, предшествующими версии 8:

- Можно использовать ключевое слово NODE вместо DBPARTITIONNUM.
- Можно использовать ключевое слово NODES вместо DBPARTITIONNUMS.

Ссылки, связанные с данной темой:

- “BACKUP DATABASE” на стр. 80

• “RESTORE DATABASE” на стр. 105

db2Rollforward - Повтор транзакций базы данных

Восстанавливает базы данных, применяя транзакции, записанные в файлах журнала базы данных. Вызывается после восстановления базы данных или табличного пространства из резервной копии, либо в случае временного отключения табличного пространства базой данных из-за ошибки носителя. Для восстановления с повтором транзакций база данных должна быть восстанавливаемой (т.е. должен быть включен хотя бы один из параметров *logretain* и *userexit*).

Область:

В среде многораздельной базы данных этот API можно вызывать только из раздела каталога. Восстановление базы данных или табличного пространства с повтором транзакций до определенного момента времени применяется ко всем серверам разделов базы данных, перечисленным в файле `db2nodes.cfg`. Восстановление базы данных или табличного пространства с повтором транзакций до конца журналов применяется к указанным серверам разделов. Если серверы разделов базы данных не указаны, то применяются все серверы разделов, перечисленные в файле `db2nodes.cfg`; если для какого-либо сервера раздела повтор транзакций не требуется, то этот сервер игнорируется.

Авторизация:

Одни из следующих:

- *sysadm*
- *sysctrl*
- *sysmaint*

Необходимое соединение:

Нет. Этот API устанавливает соединение с базой данных.

Включаемый файл API:

db2ApiDf.h

Синтаксис API C:

```
/* File: db2ApiDf.h */
/* API: db2Rollforward */
/* ... */
SQL_API_RC SQL_API_FN
db2Rollforward_api (
  db2Uint32 versionNumber,
  void *pDB2RollforwardStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2RollforwardStruct
{
  struct db2RfwdInputStruct   *roll_input;
  struct db2RfwdOutputStruct  *roll_output;
} db2RollforwardStruct;

typedef SQL_STRUCTURE db2RfwdInputStruct
{
  sqluint32                   version;
  char                        *pDbAlias;
  db2Uint32                   CallerAction;
  char                        *pStopTime;
  char                        *pUserName;
  char                        *pPassword;
  char                        *pOverflowLogPath;
  db2Uint32                   NumChngLgOvrflw;
  struct sqlurf_newlogpath    *pChngLogOvrflw;
  db2Uint32                   ConnectMode;
  struct sqlu_tablespace_bkrst_list *pTablespaceList;
  db2int32                    AllNodeFlag;
  db2int32                    NumNodes;
  SQL_PDB_NODE_TYPE           *pNodeList;
  db2int32                    NumNodeInfo;
  char                        *pDroppedTblID;
  char                        *pExportDir;
  db2Uint32                   RollforwardFlags;
} db2RfwdInputStruct;

typedef SQL_STRUCTURE db2RfwdOutputStruct
{
  char                        *pApplicationId;
  sqlint32                    *pNumReplies;
  struct sqlurf_info          *pNodeInfo;
} db2RfwdOutputStruct;
/* ... */
```

Общий синтаксис API:

```
/* File: db2ApiDf.h */
/* API: db2Rollforward */
/* ... */
SQL_API_RC SQL_API_FN
db2gRollforward_api (
  db2Uint32 versionNumber,
  void *pDB2gRollforwardStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2gRollforwardStruct
{
  struct db2gRfwdInputStruct  *roll_input;
  struct db2RfwdOutputStruct  *roll_output;
} db2gRollforwardStruct;

SQL_STRUCTURE db2gRfwdInputStruct
{
  db2Uint32                   DbAliasLen;
  db2Uint32                   StopTimeLen;
  db2Uint32                   UserNameLen;
  db2Uint32                   PasswordLen;
  db2Uint32                   OvrflwLogPathLen;
  db2Uint32                   DroppedTblIDLen;
  db2Uint32                   ExportDirLen;
  sqluint32                   Version;
  char                        *pDbAlias;
  db2Uint32                   CallerAction;
  char                        *pStopTime;
  char                        *pUserName;
  char                        *pPassword;
  char                        *pOverflowLogPath;
  db2Uint32                   NumChngLgOvrflw;
  struct sqlurf_newlogpath    *pChngLogOvrflw;
  db2Uint32                   ConnectMode;
  struct sqlu_tablespace_bkrst_list *pTablespaceList;
  db2int32                    AllNodeFlag;
  db2int32                    NumNodes;
  SQL_PDB_NODE_TYPE           *pNodeList;
  db2int32                    NumNodeInfo;
  char                        *pDroppedTblID;
  char                        *pExportDir;
  db2Uint32                   RollforwardFlags;
};

typedef SQL_STRUCTURE db2RfwdOutputStruct
{
  char                        *pApplicationId;
  sqlint32                    *pNumReplies;
  struct sqlurf_info          *pNodeInfo;
} db2RfwdOutputStruct;
/* ... */
```

Параметры API:

versionNumber
Входной. Задает версию и выпуск структуры, переданной во втором параметре.

pDB2RollforwardStruct
Входной. Указатель на структуру *db2RollforwardStruct*.

pSqlca Выходной. Указатель на структуры *sqlca*.

roll_input
Входной. Указатель на структуру *db2RfwdInputStruct*.

roll_output
Выходной. Указатель на структуру *db2RfwdOutputStruct*.

DbAliasLen
Входной. Задает длину алиаса базы данных в байтах.

StopTimeLen
Входной. Задает длину параметра времени завершения в байтах. Задайте значение ноль, если время остановки не задается.

UserNameLen
Входной. Задает длину имени пользователя в байтах. Задайте значение ноль, если имя пользователя не указано.

PasswordLen
Входной. Задает длину пароля в байтах. Задайте значение ноль, если пароль не указан.

OverflowLogPathLen
Входной. Задает длину альтернативного пути журнала в байтах. Задайте значение ноль, если путь журнала переполнения не задается.

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 (определены в `sqlutil`):

SQLUM_ROLLFWD
Повтор транзакций до момента времени, указанного в *pPointInTime*. В случае повтора транзакций для базы данных база данных остается в состоянии *отложенного повтора транзакций*. В случае повтора транзакций для табличного

пространства до заданного момента времени табличное пространства остается в состоянии *выполняемого повтора транзакций*.

SQLUM_STOP
Завершить 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
Повтор транзакций до момента времени, указанного в *pPointInTime*, и завершение восстановления с повтором транзакций. 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 без фактического 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 нельзя применять во время фактического повтора транзакций. Она может применяться только в том случае, если повтор транзакций приостановлен (т.е. ожидается STOP), или если при повторе произошел системный сбой. Используйте эту опцию с осторожностью.

Повторение транзакций для баз данных может потребовать восстановления с загрузкой при помощи лентопротяжных устройств. Если необходимо вмешательство пользователя, API применения транзакций возвращает предупреждающее сообщение. API можно вызвать заново, указав одну из следующих опций вызова:

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. В дополнение к активным файлам журналов пользователь должен также переместить в *logpath* архивные файлы журналов, поскольку они тоже могут применяться данной утилитой. Если в каталоге *logpath* недостаточно места, могут возникнуть сложности. В таких случаях используется альтернативный путь журналов. Во время восстановления с повтором транзакций поиск необходимых файлов журнала сначала выполняется в *logpath*, а затем в альтернативном пути. Файлы журнала, необходимые для восстановления табличного пространства с повтором транзакций, можно поместить либо в *logpath*, либо 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 случае применения локального сервера с одноразделной базе данных альтернативный путь может быть относительным.

NumChngLgOvrflw
Только в многораздельных базах данных. Число измененных альтернативных путей журналов. Эти пути переопределяют альтернативный путь журнала по умолчанию только для указанного сервера раздела базы данных.

pChngLogOvrflw
Только в многораздельных базах данных. Указатель на структуру, содержащую полные имена измененных альтернативных путей журналов. Эти пути переопределяют альтернативный путь журнала по умолчанию только для указанного сервера раздела базы данных.

ConnectMode
Входной. Допустимые значения (определены в `sqlutil`):

SQLUM_OFFLINE
Повтор транзакций в автономном режиме. Значение обязательно должно быть указано для восстановления базы данных с повтором транзакций.

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 в многораздельных базах данных. Входной. Указывает, что повтор транзакций следует выполнить для всех серверов разделов базы данных, перечисленных в `db2nodes.cfg`. Допустимые значения:

SQLURF_NODE_LIST
Применить к серверам разделов баз данных, перечисленных в параметре *pNodeList*.

SQLURF_ALL_NODES
Применить ко всем серверам разделов баз данных. Параметр *pNodeList* должен быть пустым (NULL). Это значение по умолчанию.

SQLURF_ALL_EXCEPT
Применить ко всем серверам разделов баз данных, кроме перечисленных в параметре *pNodeList*.

SQLURF_CAT_NODE_ONLY
Применить только к разделу каталога. Параметр *pNodeList* должен быть пустым (NULL).

NumNodes
Входной. Указывает число серверов разделов баз данных в массиве *pNodeList*.

pNodeList
Входной. Указатель на массив номеров серверов разделов баз данных, для которых необходимо выполнить восстановление с повтором транзакций.

NumNodeInfo
Входной. Определяет размер выходного параметра *pNodeInfo*, в котором должна размещаться информация о состоянии всех разделов баз данных, для которых выполняется повтор транзакций. В одноразделльной базе данных значение этого параметра должно быть равно 1. Значение этого параметра должно совпадать с числом серверов разделов базы данных, для которых вызывается этот API.

pDroppedTblID
Входной. Строка, содержащая ID отброшенной таблицы, для которой выполняется восстановление.

pExportDir
Входной. Каталог, в который будет экспортирована отброшенная таблица.

RollforwardFlags
Входной. Задает флаги повтора транзакций. Допустимые значения (определены в `sqlpapiRollforward`):

SQLP_ROLLFORWARD_LOCAL_TIME
Разрешает пользователю выполнить повтор транзакций до момента времени, заданного с применением местного времени, а не GMT. Такой подход упрощает повтор транзакций до определенного момента времени в локальных системах и устраняет возможные ошибки пользователей при преобразовании местного времени во время GMTe.

SQLP_ROLLFORWARD_NO_RETRIEVE
Задает файлы журналов, применяемые для повтора транзакций на машине, находящейся в состоянии ожидания, отключая применение архивных журналов. Выбор файлов журналов для повтора транзакций позволяет гарантировать, что состояние ожидающей машины соответствует состоянию рабочей машины с отставанием на X часов, не влияя при этом на обе системы. Эта опция полезна в том случае, если у ожидающей машины нет доступа к архивам, например, если архив TSM разрешает извлекать архивные файлы только исходной машине. Также возможна ситуация, когда ожидающая машина пытается получить неполный файл журнала в то время, как рабочая система архивирует тот же самый файл.

pApplicationId
Выходной. ID приложения.

pNumReplies
Выходной. Число полученных ответов.

pNodeInfo
Выходной. Информация о разделе базы данных.

Синтаксис REXX API:

```
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 REXX API:

алиас-базы-данных
Алиас базы данных, для которой выполняется повтор транзакций.

значение
Составная переменная хоста REXX, содержащая выходные значения. В следующем примере XXX задает имя переменной хоста:

XXX.0	Число элементов в этой переменной
XXX.1	ID программы
XXX.2	Число ответов, полученных от узлов
XXX.2.1.1	Для первого узла: номер сервера раздела базы данных
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	Для второго узла: номер сервера раздела базы данных

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 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. В следующем примере XXX задает имя 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. В следующем примере XXX задает имя 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 REXX хоста, содержащая список серверов разделов баз данных. В следующем примере XXX задает имя переменной хоста:

XXX.0 Число узлов

XXX.1 Первый узел

XXX.2 Второй узел

XXX.x и так далее.

Примечания по использованию:

Диспетчер баз данных применяет информацию, хранящуюся в архивных и активных файлах журналов, для воспроизведения транзакций, выполненных в базе данных с момента последнего резервного копирования.

Действие, выполняемое при вызове этого API, зависит от значения флага *rollforward_pending*, установленного для базы данных перед вызовом. Определить это значение можно при помощи вызова ″db2CfgGet - Get Configuration Parameters″. Если база данных находится в состоянии ожидания применения транзакций, то флаг *rollforward_pending* содержит значение `DATABASE`. Если одно или несколько табличных пространств находится в состоянии `SQLB_ROLLFORWARD_PENDING` или `SQLB_ROLLFORWARD_IN_PROGRESS`, то этот флаг содержит значение `TABLESPACE`. Если ни база данных, ни табличные пространства не требуют повтора транзакций, то флаг *rollforward_pending* содержит значение `NO`.

Если база данных находится в состоянии ожидания повтора транзакций, то при вызове этого API будет выполнен повтор транзакций. После успешного повтора транзакций в том случае, если не возникло ошибок и все табличные пространства активны, они будут возвращены в обычное состояние. Если флагу *rollforward_pending* присвоено значение `TABLESPACE`, то повтор транзакций будет выполнен только для тех табличных пространств, которые находятся в состоянии ожидания повтора транзакций, либо только для перечисленных табличных пространств.

Примечание: Если повтор транзакций для табличного пространства завершился неудачно, то табличные пространства, для которых выполнялся повтор, будут находиться в состоянии SQLB_ROLLFORWARD_IN_PROGRESS. При следующем вызове ROLLFORWARD DATABASE будут обработаны только табличные пространства, находящиеся в состоянии SQLB_ROLLFORWARD_IN_PROGRESS. Если список табличных пространств включает не все табличные пространства, находящиеся в состоянии SQLB_ROLLFORWARD_IN_PROGRESS, то необязательные таблицы будут переведены в состояние SQLB_RESTORE_PENDING.

Если база данных не находится в состоянии ожидания повтора транзакций и момент времени не задан, то для всех табличных пространств, находящихся в состоянии ROLFORWARD-IN-PROGRESS, будет выполнен повтор транзакций до конца журналов. Если нет табличных пространств, находящихся в состоянии ROLFORWARD-IN-PROGRESS, то для всех табличных пространств, находящихся в состоянии ROLFORWARD-PENDING, будет выполнен повтор транзакций до конца журналов.

Данный API считывает файлы журналов, начиная с файла, соответствующего образу резервной копии. Имя этого файла можно определить, вызвав этот API с опцией вызова SQLUM_QUERY перед повтором транзакций из файлов журналов.

Транзакции, хранящиеся в файлах журналов, применяются к базе данных. Журнал обрабатывается до достижения момента времени, указанного в параметре времени завершения, либо до обработки всей доступной информации журналов.

Восстановление останавливается в любой из следующих ситуаций:

- Больше нет файлов журналов
- Метка времени, указанная в записи файла журнала, больше, чем момент времени, указанный в параметре времени завершения.
- При чтении файла журнала произошла ошибка.

Некоторые транзакции повторить невозможно. Значение, возвращенное в параметре *pLastCommitTime*, указывает метку времени последней принятой транзакции, примененной к базе данных.

Если восстановление базы данных потребовалось из-за ошибки оператора, то можно указать в параметре времени завершения *pStopTime* метку времени, чтобы восстановление было завершено на моменте времени, предшествующем ошибке. Это применимо только к полному восстановлению баз данных с повтором транзакций, а также к случаю повтора транзакций в табличном пространстве до определенного момента времени. Таким образом можно также

остановить восстановление до того, как произойдет ошибка чтения файла журнала, обнаруженная во время предыдущей неудачной попытки восстановления.

Если флаг *rollforward_recovery* содержит значение DATABASE, то с базой данных нельзя будет работать до тех пор, пока повтор транзакций не будет прерван. Прерывать операцию можно вызовом API с опцией вызова SQLUM_STOP или SQLUM_ROLLFORWARD_STOP для вывода базы данных из состояния ожидания повтора транзакций. Если флаг *rollforward_recovery* равен TABLESPACE, значит с базой данных можно работать. Однако, табличные пространства, находящиеся в состоянии SQLB_ROLLFORWARD_PENDING и SQLB_ROLLFORWARD_IN_PROGRESS, будут недоступны до тех пор, пока не будет вызван этот API для повтора транзакций к табличным пространствам. Если выполняется повтор транзакций до определенного момента времени для табличных пространств, то после успешного повтора эти табличные пространства переключаются в состояние ожидания резервного копирования.

Если в опции *RollforwardFlags* указано значение SQLP_ROLLFORWARD_LOCAL_TIME, то все возвращаемые пользователю сообщения будут содержать значения местного времени. Все преобразования времени выполняются на сервере, а в многораздельной базе данных - в разделе каталога. Строка метки времени преобразуется на сервере в GMT, т.е. применяется местное время часового пояса сервера, а не клиента. Если клиент и сервер находятся в разных часовых поясах, то следует применять местное время сервера. В этом заключается отличие от опции местного времени Центра управления, на котором применяется местное время клиента. Если момент времени близок к моменту перевода часов при переходе на летнее или зимнее время, то важно учесть, на какой момент приходится время завершения операции - до или после перевода часов - и правильно задать время завершения.

Ссылки, связанные с данной темой:

- “SQLCA” в *Administrative API Reference*

Примеры, связанные с данной темой:

- “dbrecov.sqc -- How to recover a database (C)”
- “dbrecov.sqC -- How to recover a database (C++)”

Сеансы повтора транзакций - Примеры для 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 узах 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 в группе разделов базы данных на одном узле (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).

Часть 2. Системы высокой доступности

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”). В этом разделе обсуждаются следующие вопросы:

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 многораздельной базы данных для эффективной обработки транзакций DB2® может использовать как внутрираздельный, так и межраздельный параллелизм. *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.
- Должна быть возможность быстро восстановить систему, если произошла аппаратная или программная ошибка или случилась авария. В DB2 используется непрерывная система контрольных точек и есть возможность параллельного восстановления, что обеспечивает предельно быстрое восстановления при отказах.

  Способность системы быстро восстанавливаться зависит также от наличия надежной резервной копии и алгоритма ее восстановления.
- 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. 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 можно найти в документе “IBM® DB2 Universal Database™ Enterprise Edition for AIX® and HACMP/ES”, расположенном на Web-сайте “DB2 UDB and DB2 Connect Online Support” (http://www.ibm.com/software/data/pubs/papers/).

- Microsoft® Cluster Server для операционных систем Windows®.

  За информацией о продукте Microsoft Cluster Server обратитесь к следующим документам, которые можно найти на Web-сайте “DB2 UDB and DB2 Connect™ Online Support” (http://www.ibm.com/software/data/pubs/papers/): “Implementing IBM DB2 Universal Database Enterprise - Extended Edition with Microsoft Cluster Server”, “Implementing IBM DB2 Universal Database Enterprise Edition with Microsoft Cluster Server” и “DB2 Universal Database for Windows: High Availability Support Using Microsoft Cluster Server - Overview”.

- Серверы Sun Cluster или VERITAS Cluster для операционной среды Solaris.

  За информацией о продукте Sun Cluster 3.0 обратитесь к документу “DB2 and High Availability on Sun Cluster 3.0”, который расположен на Web-сайте “DB2 UDB and DB2 Connect Online Support” (http://www.ibm.com/software/data/pubs/papers/). Информацию о VERITAS

Cluster Server можно найти в документе “DB2 and High Availability on VERITAS Cluster Server”, который также расположен на Web-сайте “DB2 UDB and DB2 Connect Online Support”.

- Multi-Computer/ServiceGuard для Hewlett-Packard.

  За информацией о MC/ServiceGuard для Hewlett-Packard обратитесь к документу, в котором обсуждается реализация DB2 IBM и его сертификация для применения программы с высокой доступностью MC/ServiceGuard фирмы Hewlett-Packard. Этот документ можно найти на Web-сайте “IBM Data Management Products for HP” (http://www.ibm.com/software/data/hp/pdfs/db2mc.pdf).

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.

Понятия, связанные с данным:

- “Параллелизм” в *Руководство администратора: Планирование*
- “Разработка стратегии резервного копирования и восстановления” на стр. 3
- “Высокая доступность с помощью оперативного отделения зеркальной копии и поддержки приостановленного ввода/вывода” на стр. 182
- “Высокая доступность в операционной среде Solaris” на стр. 207
- “Высокая доступность в Sun Cluster 3.0” на стр. 210
- “Обеспечение высокой доступности с помощью VERITAS Cluster Server” на стр. 213
- Глава 7, “Высокая доступность в операционной системе Windows” на стр. 201
- Глава 6, “Высокая доступность в AIX” на стр. 193

Обеспечение высокой доступности путем отправки журналов

Отправка журналов - это процесс копирования всех файлов журналов на резервный компьютер с устройства архивирования или через программу обработчика пользователя, запущенную для исходной базы данных. Для резервной базы данных постоянно выполняется повтор транзакций, зарегистрированных в файлах журнала, созданных на рабочем компьютере. В случае сбоя рабочего компьютера запускается восстановление после отказа и выполняются следующие действия:

- На резервный компьютер передаются остальные журналы.
- Для резервной базы данных выполняется повтор транзакций до конца журналов, после чего она останавливается.
- Клиенты подключаются к резервной базе данных и продолжают свою работу.

На резервном компьютере должны быть свои ресурсы (или диски), однако заданные на нем физические и логические определения должны совпадать с определениями рабочей базы данных. При таком подходе исходная база данных восстанавливается на резервном компьютере с помощью утилиты восстановления или функции отделения зеркальной копии.

Для того чтобы в случае аварии можно было восстановить базу данных, должны быть соблюдены следующие условия:

- Архив и исходная копия должны географически располагаться в разных местах.

- На компьютере резервной базы данных необходимо создать удаленную зеркальную копию журнала
- Для того чтобы предотвратить потерю данных нужно обеспечить синхронизацию зеркальных копий. Это можно сделать путем применения функции зеркального копирования журнала DB2® или современной дисковой подсистемы, такой как ESS или EMC. Для того чтобы минимизировать влияние восстановления после аварии на производительность, рекомендуется использовать кэш NVRAM в локальной и удаленной системе.

Примечания:

1. Если в резервной базе данных обрабатывается запись журнала, указывающая, что в исходной базе данных были восстановлены индексы, то на резервном сервере индексы не восстанавливаются автоматически. Индекс восстанавливается при первом подключении к базе данных, либо при первом обращении к индексу после выхода резервного сервера из состояния отложенного повтора транзакций. После восстановления индексов на исходном сервере его рекомендуется синхронизировать с резервным сервером.
2. Если в исходной базе данных была выполнена утилита загрузки с опцией COPY YES, то резервной базе данных нужно предоставить доступ к образу копии.
3. Если в исходной базе данных была выполнена утилита загрузки с опцией COPY NO, то резервную базу данных необходимо синхронизировать. В противном случае табличное пространство будет переведено в состояние отложенного восстановления.
4. Резервный компьютер можно инициализировать двумя способами:
   a. Путем восстановления на этом компьютере резервной копии.
   b. Путем создания в рабочей системе отделенной зеркальной копии и вызова команды **db2inidb** с опцией STANDBY.

   Команду ROLLFORWARD можно вызвать для резервной системы только после ее инициализации.
5. Операции, не зарегистрированные в журнале, не будут выполнены в резервной базе данных. В связи с этим после выполнения таких операций рекомендуется синхронизировать резервную базу данных. Это можно сделать путем оперативного создания отделенной зеркальной копии и приостановки ввода-вывода.

Понятия, связанные с данным:

- “Высокая доступность с помощью оперативного отделения зеркальной копии и поддержки приостановленного ввода/вывода” на стр. 182

Задачи, связанные с данной темой:

- “Применение отделенной зеркальной копии в качестве резервной базы данных” на стр. 184

Ссылки, связанные с данной темой:

- Приложение G, “Обработчик пользователя для восстановления баз данных” на стр. 345

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 отделения основной копии базы данных от дополнительной.

Если вы не хотите создавать резервную копию для большой базы данных с помощью утилиты резервного копирования DB2®, вы можете сделать резервную копию с зеркальной копии с помощью приостановленного ввода-вывода и функции отделения 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, для того чтобы ее можно было использовать:

- Как клон базы данных
- Как резервную базу данных
- Как резервную копию

Эту команду можно вызвать только для отделенной зеркальной копии. Такую копию можно использовать только после того, как будет выполнена эта команда.

В среде многораздельных баз данных необязательно приостанавливать операции записи на всех разделах одновременно. Вы можете приостановить подмножество из одного или нескольких разделов и создать отдельные зеркальные копии для автономного резервного копирования. Если в это подмножество входит узел каталога, то его следует приостановить последним.

В среде многораздельных баз данных надо выполнить команду **db2inidb** для всех разделов, прежде чем можно будет использовать отделенную копию какого-либо из разделов. С помощью команды **db2_all** утилиту можно запустить во всех разделах одновременно.

Примечание: Убедитесь, что отделенная зеркальная копия содержит все контейнеры и каталоги, относящиеся к базе данных, в том числе каталог томов.

Ссылки, связанные с данной темой:

- “db2inidb - инициализировать зеркальную копию базы данных” на стр. 238

Оперативное зеркальное разделение

Создание клона базы данных

Ограничения:

Для клонированной базы данных нельзя создать резервную копию, восстановить эту копию в исходной системе и повторить транзакции из файлов журналов, созданных в исходной системе.

Процедура:

Чтобы клонировать базу данных, выполните следующие действия:

1. Приостановите операции ввода/вывода в исходной базе данных:
   ```
   db2 set write suspend for database
   ```
2. Воспользуйтесь соответствующими командами 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, и начнет новую цепочку журналов, для того чтобы журналы основной базы данных не могли повторно использоваться в клонированной базе данных.

Понятия, связанные с данным:

- “Высокая доступность с помощью оперативного отделения зеркальной копии и поддержки приостановленного ввода/вывода” на стр. 182

Ссылки, связанные с данной темой:

- “db2inidb - инициализировать зеркальную копию базы данных” на стр. 238

Применение отделенной зеркальной копии в качестве резервной базы данных

Процедура:

Для того чтобы отделенная зеркальная копия использовалась как резервная база данных, выполните следующие действия:

1. Приостановите операции ввода/вывода в исходной базе данных:
   ```
   db2 set write suspend for database
   ```
2. Воспользуйтесь соответствующими командами 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
   ```

   Примечание: Если у вас есть только 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 файлы журналов.
7. Повторите для базы данных транзакции до конца журналов или до определенного момента времени.
8. Продолжайте получать файлы журналов и повторять транзакции до конца журналов или до определенного момента времени для резервной базы данных.
9. Для активизации резервной базы данных вызовите команду ROLLFORWARD с опцией STOP.

Понятия, связанные с данным:

- “Высокая доступность с помощью оперативного отделения зеркальной копии и поддержки приостановленного ввода/вывода” на стр. 182

Задачи, связанные с данной темой:

- “Создание клона базы данных” на стр. 183
- “Применение отделенной зеркальной копии в качестве резервной копии” на стр. 185

Ссылки, связанные с данной темой:

- “db2inidb - инициализировать зеркальную копию базы данных” на стр. 238

Применение отделенной зеркальной копии в качестве резервной копии

Процедура:

Для применения отделенной зеркальной копии в качестве “резервной копии” выполните следующие действия:

1. Приостановите операции ввода/вывода в исходной базе данных:
   ```
   db2 set write suspend for database
   ```
2. Воспользуйтесь соответствующими командами 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. В исходной системе происходит отказ, и требуется восстановление из резервной копии.
5. Остановите исходный экземпляр базы данных:
   ```
   db2stop
   ```
6. Воспользуйтесь командами операционной системы, чтобы скопировать отделенные данные поверх исходной системы. **Не копируйте отдельные файлы журналов**, поскольку для восстановления путем повтора транзакций необходимы журналы исходной системы.
7. Запустите исходный экземпляр базы данных:
   ```
   db2start
   ```
8. Инициализируйте исходную базу данных:
   ```
   db2inidb алиас_базы_данных as mirror
   ```
9. Повторите транзакции для исходной базы данных до конца журналов или до определенного момента времени и остановите операцию.

Понятия, связанные с данным:

- “Высокая доступность с помощью оперативного отделения зеркальной копии и поддержки приостановленного ввода/вывода” на стр. 182

Задачи, связанные с данной темой:

- “Создание клона базы данных” на стр. 183
- “Применение отделенной зеркальной копии в качестве резервной базы данных” на стр. 184

Ссылки, связанные с данной темой:

- “db2inidb - инициализировать зеркальную копию базы данных” на стр. 238

Функция монитора ошибок в системах на основе UNIX

В системах на основе UNIX® для повышения доступности среды DB2® без кластеров может применяться функция монитора ошибок. Эта функция представляет собой цепочку процессов, которые совместно контролируют работу DB2. Демон *init* контролирует работу координатора монитора ошибок (FMC), FMC контролирует работу мониторов ошибок, а мониторы ошибок контролируют работу DB2.

Координатор монитора ошибок (FMC) - это процесс функции монитора ошибок, который запускается во время загрузки UNIX. Демон *init* запускает FMC и перезапускает его в случае аварийного завершения FMC. FMC запускает один монитор ошибок для каждого экземпляра DB2. Каждый монитор ошибок выполняется в виде процесса демона с теми же привилегиями пользователя, что и экземпляр DB2. Для того чтобы убедиться, что монитор ошибок преждевременно не завершил свою работу, работа монитора постоянно контролируется. В случае сбоя монитора ошибок он перезапускается процессом FMC. Каждый монитор ошибок, в свою очередь, контролирует работу одного экземпляра DB2. В случае преждевременного завершения работы экземпляра DB2 монитор ошибок перезапускает экземпляр.

Примечания:

1. Если применяется продукт для поддержки кластеров с высокой доступностью (HACMP или MSCS), то функцию монитора ошибок необходимо выключить, так как запуском и завершением работы экземпляра управляет этот продукт.
2. Монитор ошибок деактивируется только в том случае, если была вызвана команда **db2stop**. Если работа экземпляра DB2 была завершена любым другим способом, то монитор ошибок снова запустит экземпляр.

Файл реестра монитора ошибок

Если запущен демон монитора ошибок, то на каждом физическом компьютере для каждого экземпляра создается файл реестра монитора ошибок. Значения, указанные в этом файле, определяют параметры работы мониторов ошибок. Файл реестра расположен в каталоге `/sqllib/` и называется `fm.<имя_компьютера>.reg`. Для изменения файла служит команда **db2fm**. Этот файл содержит следующие записи:

```
FM_ON = no
FM_ACTIVE = yes
START_TIMEOUT = 600
STOP_TIMEOUT = 600
STATUS_TIMEOUT = 20
STATUS_INTERVAL = 20
RESTART_RETRIES = 3
ACTION_RETRIES = 3
NOTIFY_ADDRESS = <имя_экземпляра>@<имя_компьютера>
```

где:

FM_ON
: Указывает, следует ли запускать монитор ошибок. Если значение параметра равно `NO`, то демон монитора ошибок не будет запущен, либо будет выключен, если он уже запущен. Значение по умолчанию равно `NO`.

FM_ACTIVE
: Указывает, активен ли монитор ошибок. Монитор ошибок будет работать лишь в том случае, если оба параметра FM_ON и FM_ACTIVE равны `YES`. Если параметр FM_ON равен `YES`, а FM_ACTIVE равен `NO`, то демон монитора ошибок будет запущен, но не будет активен. Это означает, что в случае завершения работы DB2 монитор ошибок не будет пытаться снова запустить DB2. Значение по умолчанию равно `YES`.

START_TIMEOUT
: Задает интервал времени, в течение которого монитор ошибок должен запустить контролируемую службу. Значение по умолчанию равно 600 секунд.

STOP_TIMEOUT
: Задает интервал времени, в течение которого монитор ошибок должен завершить работу контролируемой службы. Значение по умолчанию равно 600 секунд.

STATUS_TIMEOUT
: Задает интервал времени, в течение которого монитор ошибок должен получить информацию о состоянии контролируемой службы. Значение по умолчанию равно 20 секунд.

STATUS_INTERVAL
: Задает минимальный интервал времени между двумя вызовами функции для получения состояния контролируемой службы. Значение по умолчанию равно 20 секунд.

RESTART_RETRIES

Указывает, сколько раз монитор ошибок должен пытаться получить информацию о состоянии контролируемой службы, если первая попытка была неудачной. После того как число попыток достигнет указанного значения, монитор ошибок выполнит процедуру активации службы. Значение по умолчанию равно 3.

ACTION_RETRIES

Указывает, сколько раз монитор ошибок должен пытаться активизировать службу. Значение по умолчанию равно 3.

NOTIFY_ADDRESS

Задает адрес электронной почты, по которому монитор ошибок будет отправлять уведомления. Значение по умолчанию - <имя_экземпляра>@<имя_компьютера>

Для изменения файла можно воспользоваться командой **db2fm**. Например:

Для того чтобы присвоить параметру START_TIMEOUT значение 100 секунд для экземпляра DB2INST1, вызовите следующую команду:

```
db2fm -i db2inst1 -T 100
```

Для того чтобы присвоить параметру STOP_TIMEOUT значение 200 секунд для экземпляра DB2INST1, вызовите следующую команду:

```
db2fm -i db2inst1 -T /200
```

Для того чтобы присвоить параметру START_TIMEOUT значение 100 секунд, а параметру STOP_TIMEOUT - 200 секунд для экземпляра DB2INST1, вызовите следующую команду:

```
db2fm -i db2inst1 -T 100/200
```

Для того чтобы включить монитор ошибок для экземпляра DB2INST1, введите:

```
db2fm -i db2inst1 -f yes
```

Для того чтобы выключить монитор ошибок для экземпляра DB2INST1, введите:

```
db2fm -i db2inst1 -f no
```

Примечание: Если файл реестра монитора ошибок не существует, то применяются значения по умолчанию.

Ссылки, связанные с данной темой:

- “db2fm - Монитор отказов DB2” на стр. 189

db2fm - Монитор отказов DB2

Управляет демоном монитора отказов DB2. Для настройки этого монитора применяется утилита db2fm.

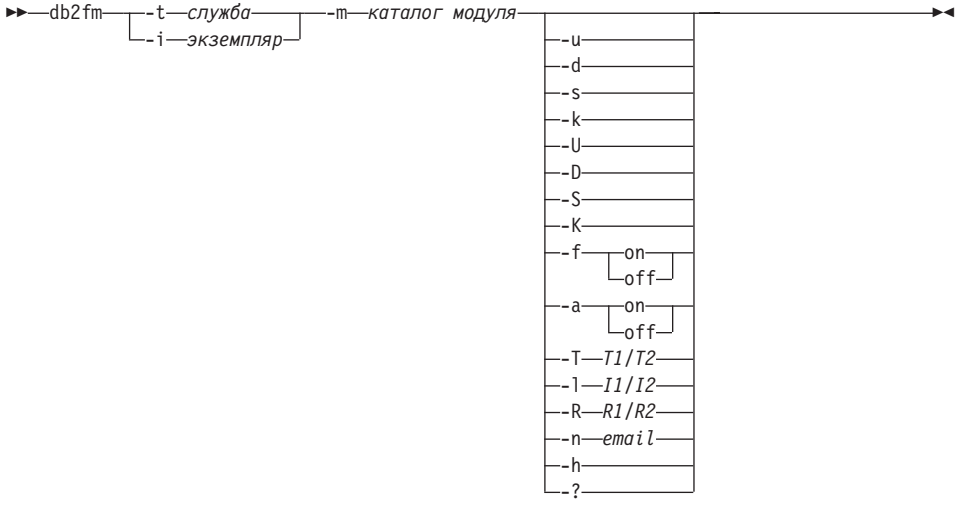
Авторизация:

Права доступа к экземпляру, для которого вы выполняете команду.

Необходимое соединение:

Нет.

Синтаксис команды:



Параметры команды:

-m *каталог модуля*
Задает полный путь к общей библиотеке монитора отказов для отслеживаемого продукта. Значение по умолчанию - $INSTANCEHOME/sqllib/lib/libdb2gcf.

-t *служба*
Уникальное текстовое описание службы.

-i *экземпляр*
Указывает экземпляр службы.

-u Включает службу.

-U Включает демон монитора отказов.

-d Выключает службу.

-D Выключает демон монитора отказов.

-k Снимает процесс службы.

-K Снимает процесс демона монитора отказов.

-s Возвращает состояние службы.

-S Возвращает состояние демона монитора отказов.

Примечание: Возможны следующие состояния службы или монитора отказов:

- Не установлен
- Установлен, но не работает
- Работает, но недоступен (обслуживается)
- Доступен
- Не известен

-f *on*|*off*

Включает или выключает монитор отказов.

Примечание: Если указано значение off, то демон монитора отказов не будет запущен, а если он уже работает, то будет остановлен.

-a *on*|*off*

Активирует или деактивирует отслеживание отказов.

Примечание: Если указано значение off, то монитор отказов не будет активно отслеживать отказы, то есть не будет пытаться вновь запустить остановившиеся службы.

-T *T1/T2*

Переопределяет время ожидания запуска или остановки.

например:

- -T 15/10 указывает оба времени ожидания
- -T 15 изменяет время ожидания запуска на 15 секунд
- -T /10 изменяет время ожидания остановки на 10 секунд

-I *I1/I2*

Задает интервал и время ожидания информации о состоянии.

-R *R1/R2*

Задает число попыток узнать состояние и действие, предпринимаемое при ошибке.

-n *email*

Задает адрес электронной почты для оповещения об ошибках.

-h Показывает информацию о формате команды.

-? Показывает информацию о формате команды.

Замечания по использованию:

1. Эта команда применяется только в UNIX.

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. 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.
- Утилиты клиента HACMP для контроля за состоянием и обнаружения его изменения (в одном или нескольких кластерах) на физических узлах AIX®, расположенных 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 пользовательские события, которые связаны с наблюдением за параметрами аппаратных и программных компонентов.

Одно из стандартных событий - событие node_down (узел выключен). При составлении плана восстановления HACMP допускает два варианта восстановления после отказа: горячее резервирование и взаимную подмену.

Примечание: Если применяется HACMP, то экземпляры DB2® не должны запускаться во время загрузки. Для выполнения этого требования вызовите утилиту **db2iauto**, как показано ниже:

```
db2iauto -off имя_экземпляра
```

где

имя_экземпляра - регистрационное имя этого экземпляра.

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 среде многораздельной базы данных DB2 Universal Database обычно выполняется с режиме взаимной подмены с разделами на каждом узле. Исключение составляет сценарий, в котором узел каталога является частью конфигурации горячего резервирования.

При планировании установки большой системы DB2 в среде RS/6000® SP™, использующей HACMP ES, следует решить, нужно ли разместить узлы кластера в одной стойке RS/6000 SP или их нужно распределить 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. Следует помнить о том, что узлы добавляются в кластер HACMP по отдельности. Запуск конфигурации из нескольких кластеров занимает меньше времени, 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, то эти узлы смогут обращаться к внешним дискам и перезапускать 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 с участием DB2 UDB в среде многораздельной базы данных, внимательно ознакомьтесь с документацией 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).

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. В среде многораздельной базы данных 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 быть задан в одной из следующих форм:

Таблица 1. 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
```

Настройка разделов базы данных DB2 для HACMP ES

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.

Вместе с продуктом DB2 UDB Enterprise Server Edition поставляется (и устанавливается на каждом узле в каталоге `/usr/bin`) файл сценария `rc.db2pe`, помогающий настроить процедуру восстановления после отказа HACMP ES на узлах горячего резервирования или оперативной подмены. Кроме того, размеры пула буферов DB2 могут настраиваться во время отработки отказа в конфигурациях взаимной подмены из `rc.db2pe`. (Когда на одном физическом узле работают два раздела базы данных, можно настроить размеры пула буферов для обеспечения правильного распределения ресурсов.)

Слежение за событиями HACMP ES и события, определяемые пользователем

Примером события, определяемого пользователем, может служить запуск процедуры восстановления после отказа в случае преждевременного завершения процесса на заданном узле. Другие примеры событий, определяемых пользователем, в том числе завершение работы раздела базы данных и принудительное завершение транзакции для освобождения пространства подкачки, приведены в каталоге `samples/hacmp/es`.

Файл правил `/user/sbin/cluster/events/rules.hacmprd` содержит события HACMP. Каждое описание события в этом файле состоит из девяти компонентов:

- Уникальное имя события.
- Статус или спецификатор события. Имя события и статус - определяющие атрибуты правила. Менеджер кластера HACMP ES инициирует восстановление, только если находит правило с именем и статусом, соответствующим имени и статусу события.
- Путь программы ресурсов - полный путь файла *xxx*`.rp`, содержащего программу восстановления.
- Тип восстановления. Этот компонент зарезервирован для будущего использования.
- Уровень восстановления. Этот компонент зарезервирован для будущего использования.
- Имя переменной ресурсов, используемое для событий менеджера событий.
- Вектор экземпляра, используемый для событий менеджера событий. Это набор элементов в форме ″имя=значение″. Значения уникально определяют копию ресурса в системе и, по расширению, копию переменной ресурса.

- Предикат, используемый для событий менеджера событий. Это логическое выражение, сравнивающее переменную ресурса с другими элементами. Когда это выражение истинно, подсистема менеджера событий генерирует информационное событие для менеджера кластера и соответствующей прикладной программы.
- Предикат реактивации, используемый для событий менеджера событий. Этот предикат служит для генерации события, изменяющего статус первичного предиката. В типичном случае этот предикат - инверсия первичного предиката. Он может также использоваться с предикатом события для задания верхней и нижней границы для рассматриваемого условия.

Каждый объект занимает отдельную строку в определении события, даже если эта строка не используется. Если эти строки удалить, менеджер кластера HACMP ES не сможет правильно проанализировать определение события, что может привести к зависанию системы. Все строки, начинающиеся с "#", считаются комментариями.

Примечание: Файл правил занимает ровно по девять строк на каждое определение события, не считая строк комментариев. При добавлении в конец файла правил события, определяемого пользователем, важно удалить лишние пустые строки в конце файла, иначе узел зависнет.

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.

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`).

Более подробная информация о реализации и структуре среды IBM® DB2 Universal Database™ с высокой доступностью в AIX приведена в документах, расположенных на Web-сайте ″DB2 UDB and DB2 Connect™ Support″ (http://www.ibm.com/software/data/pubs/papers/).:

- ″IBM DB2 Universal Database Enterprise Edition for AIX and HACMP/ES″
- ″IBM DB2 Universal Database Enterprise - Extended Edition for AIX and HACMP/ES″

Ссылки, связанные с данной темой:

- “db2start - Start DB2 Command” в *Command Reference*

Глава 7. Высокая доступность в операционной системе Windows

Введение

MSCS - это функция операционных систем Windows® NT Server, Windows 2000 Server и Windows .NET Server. Эта программа позволяет установить соединение между двумя серверами кластера (или максимум четырьмя серверами в DataCenter Server) для обеспечения высокой доступности и упрощения управления данными и прикладными программами. Кроме того, MSCS может автоматически выполнять процедуру восстановления в случае сбоя сервера или прикладной программы. С помощью этой функции можно переместить часть нагрузки на другой сервер, чтобы компьютеры использовались равномерно, и выполнить запланированное обслуживание без простоя.

Функцию MSCS поддерживают следующие продукты DB2®:

- DB2 Universal Database™ Workgroup Server Edition
- DB2 Universal Database Enterprise Server Edition (DB2 ESE)
- DB2 Universal Database Connect Enterprise Edition (DB2 CEE)

Компоненты MSCS DB2

Кластер - это группа из двух или более узлов, каждый из которых представляет собой независимую компьютерную систему. С точки зрения клиентов сети кластер представляет собой один сервер.

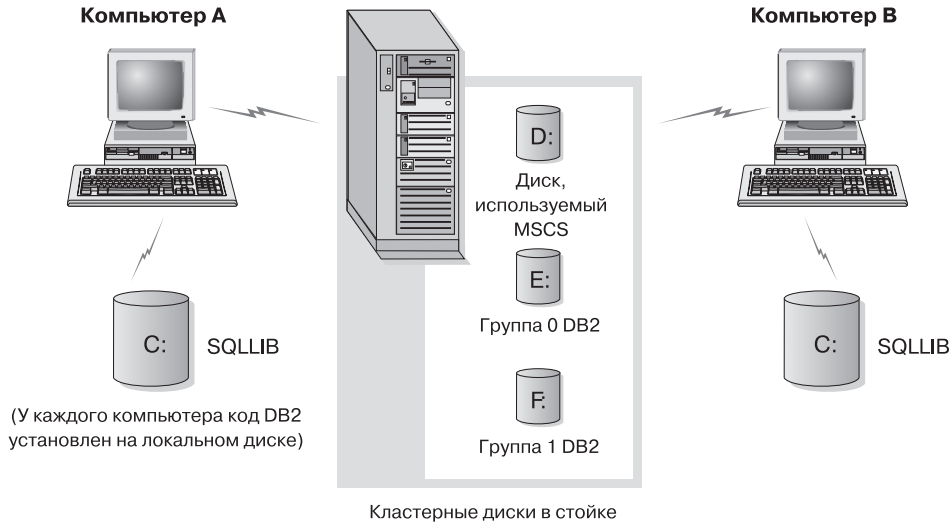


Рисунок 16. Пример конфигурации MSCS

Узлы кластера MSCS соединены друг с другом при помощи шин общей памяти и одной или нескольких сетей, независимых на физическом уровне. Сеть, содержащая серверы, но не содержащая клиенты, подключающиеся к кластеру, называется *частной* сетью. Сеть, содержащая соединения клиентов, называется *общей* сетью. На каждом узле есть локальные диски. Каждая шина общей памяти подключена к одному или нескольким дискам. Каждый диск, подключенный к общей шине, в каждый момент времени принадлежит только одному узлу кластера. Программное обеспечение DB2 хранится на локальном диске. Файлы базы данных DB2 (таблицы, индексы, файлы журналов и т.д.) хранятся на общих дисках. Поскольку MSCS не поддерживает использование непосредственных разделов в кластере, DB2 нельзя настроить на применение непосредственных устройств в среде MSCS.

Ресурс DB2

В среде MSCS ресурсом называется объект, для управление которым применяется программное обеспечение кластера. Примером ресурса может служить диск, IP-адрес или стандартная служба. Для применения DB2 в среде MSCS предусмотрен тип ресурсов ″DB2″. Каждый ресурс DB2 управляет экземпляром DB2, а при работе в среде многораздельной базы данных - разделом базы данных. Имя ресурса DB2 совпадает с именем экземпляра, а в среде многораздельной базы данных имя ресурса DB2 состоит из имени экземпляра и номера раздела (или узла).

Сценарии, выполняемые до и после активации ресурса

Сценарии можно выполнить как до, так и после активации ресурса DB2. Такие сценарии называются сценариями до активации и после активации, соответственно. Сценарии представляют собой файлы .BAT, содержащие команды DB2 и системы.

Если на одном компьютере может быть запущено несколько экземпляров DB2, то с помощью сценариев до и после активации можно изменить конфигурацию таким образом, чтобы все экземпляры были запущены успешно. В случае сбоя может вызываться сценарий после активации для восстановления базы данных вручную. Кроме того, сценарий после активации может применяться для запуска прикладных программ и служб, зависящих от DB2.

Группа DB2

Связанные или зависящие друг от друга ресурсы объединяются в группу ресурсов. Все ресурсы группы одновременно перемещаются между узлами кластера. Например, в стандартной среде кластера с одноразделной базой данных DB2 будет создана группа DB2, содержащая следующие ресурсы:

1. Ресурс DB2. Ресурс DB2 управляет экземпляром DB2 (или узлом).
2. Ресурс IP-адреса. Этот ресурс применяется клиентскими прикладными программами для подключения к серверу DB2.
3. Ресурс сетевого имени. Этот ресурс применяется клиентскими прикладными программами для подключения к серверу DB2 при помощи имени, а не IP-адреса. Ресурс сетевого имени зависит от ресурса IP-адреса. Это необязательный ресурс. (Создание ресурса сетевого имени может привести к снижению производительности восстановления после сбоя.)
4. Один или несколько ресурсов физических дисков. Каждый ресурс физического диска соответствует общему диску кластера.

Примечание: Ресурс DB2 зависит от всех остальных ресурсов группы, поэтому сервер DB2 можно запустить только после активации всех остальных ресурсов.

Конфигурации восстановления после сбоев

Доступны два типа конфигурации:

- Горячее резервирование
- Взаимная подмена

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. 17.

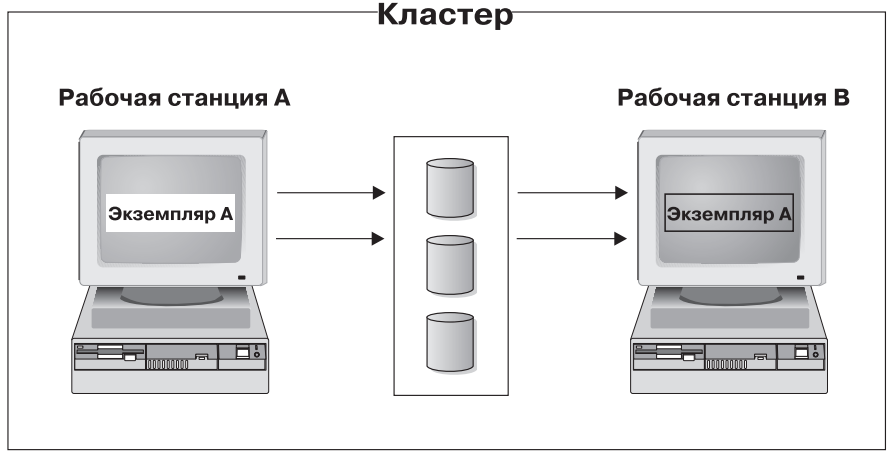


Рисунок 17. Конфигурация горячего резервирования

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. 18 на стр. 205.

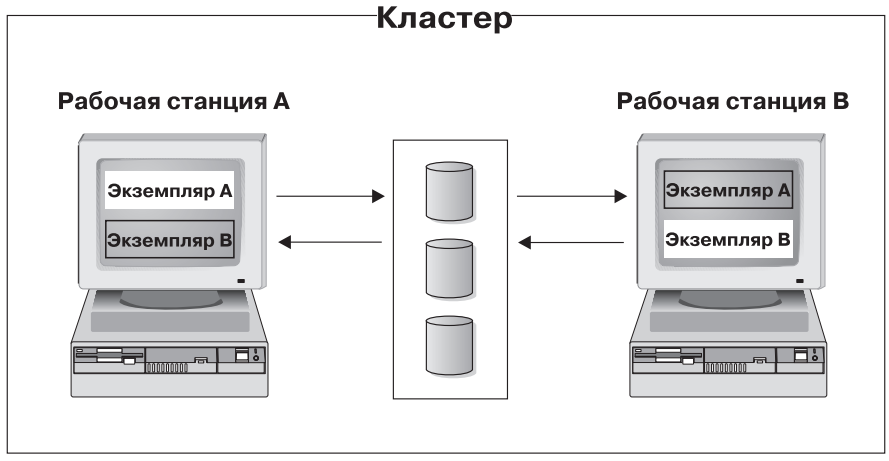


Рисунок 18. Конфигурация взаимной подмены

Более подробную информацию о реализации и структуре среды IBM® DB2 Universal Database с высокой доступностью в операционной системе Windows можно найти в документах, приведенных на Web-сайте ″DB2 UDB and DB2 Connect™ Support″ (http://www.ibm.com/software/data/pubs/papers/).:

- ″Implementing IBM DB2 Universal Database Enterprise - Extended Edition with Microsoft® Cluster Server″
- ″Implementing IBM DB2 Universal Database Enterprise Edition with Microsoft Cluster Server″
- ″DB2 Universal Database for Windows: High Availability Support Using Microsoft Cluster Server - Overview″

Глава 8. Высокая доступность в операционной среде Solaris

Высокая доступность в операционной среде Solaris

Для обеспечения высокой доступности в операционной среде Solaris DB2® может применяться совместно с продуктом Sun Cluster 3.0 (SC3.0) или Veritas Cluster Server (VCS). За информацией о Sun Cluster 3.0 обратитесь к документу “DB2 and High Availability on Sun Cluster 3.0”, который расположен на Web-сайте “DB2 UDB and DB2 Connect Online Support” (http://www.ibm.com/software/data/pubs/papers/). За информацией о VERITAS Cluster Server обратитесь к документу “DB2 and High Availability on VERITAS Cluster Server”, который также расположен на Web-сайте “DB2 UDB and DB2 Connect™ Online Support”.

Примечание: Если применяется продукт Sun Cluster 3.0 или Veritas Cluster Server, то экземпляры DB2 не должны запускаться во время загрузки. Для выполнения этого требования вызовите утилиту **db2iauto**, как показано ниже:

```
db2iauto -off имя_экземпляра
```

где

`имя_экземпляра` - регистрационное имя этого экземпляра.

Высокая доступность

Компьютерные системы, где размещаются службы данных, содержат множество различных компонентов, каждый из которых характеризуется своей ″средней наработкой на отказ″ (mean time before failure, MTBF). Наработка на отказ - это среднее время, в течение которого данный компонент остается исправным. Для качественного жесткого диска MTBF составляет порядка миллиона часов (приблизительно 114 лет). Хотя этот срок может показаться долгим, из каждых 200 дисков один, скорее всего, откажет уже через 6 месяцев.

Для увеличения доступности службы данных существуют разные методы, из которых чаще всего применяется метод кластеров высокой доступности. Кластер, обеспечивающий высокую доступность, состоит из двух или более компьютеров, набора интерфейсов собственной сети, одного или нескольких интерфейсов общедоступной сети и нескольких совместно используемых дисков. Эта особая конфигурация позволяет перемещать службы данных с одного компьютера на другой. Благодаря такой возможности служба данных сохраняет возможность обращаться к своим данным. Перемещение службы данных с

одного компьютера на другой называется *восстановлением после сбоев*, как показано на рис. 19.

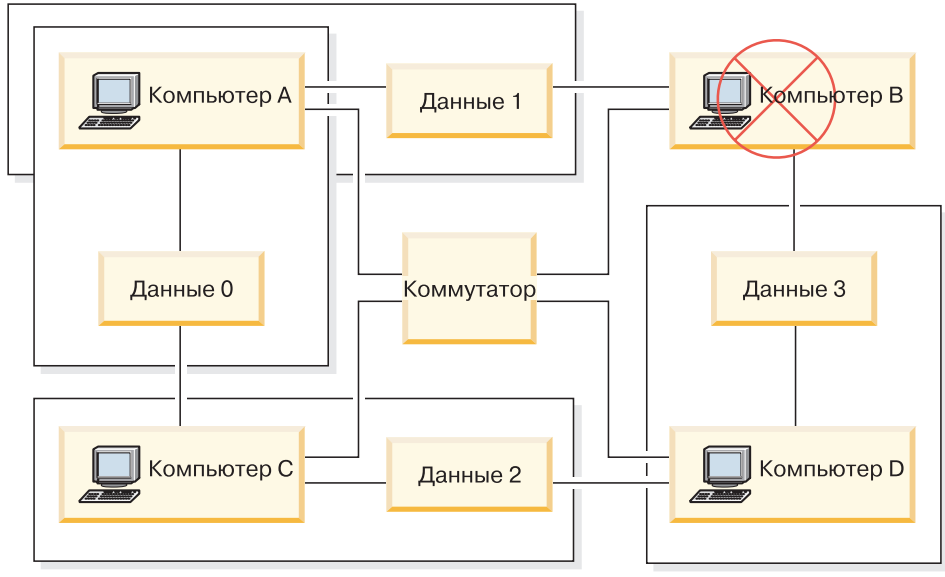


Рисунок 19. Восстановление после сбоев. В случае сбоя компьютера B его служба данных перемещается на другой компьютер кластера, поэтому данные остаются доступными.

Интерфейсы собственной сети служат для обмена сообщениями *работоспособности*, а также управляющими сообщениями между компьютерами в кластере. Интерфейсы общедоступной сети используются для непосредственной связи с клиентами кластера высокой доступности. Диски в кластере высокой доступности соединены с двумя или более компьютерами кластера, поэтому в случае отказа одного из компьютеров другие компьютеры сохраняют к ним доступ.

Служба данных, запущенная в кластере высокой доступности, имеет один или несколько логических интерфейсов общедоступной сети и набор связанных с ними дисков. Клиенты службы данных высокой доступности соединяются через протокол TCP/IP только с логическими сетевыми интерфейсами службы данных. В случае сбоя служба данных вместе со своими логическими сетевыми интерфейсами и набором дисков перемещается на другой компьютер.

Одно из преимуществ кластера высокой доступности состоит в том, что службы данных можно восстанавливать без помощи персонала поддержки, причем в любое время. Другое преимущество заключается в дублировании. Все части кластера, включая сами компьютеры, должны быть продублированы. Кластер должен быть рассчитан на успешное преодоление ошибки в любой отдельной точке.

Как бы ни различались службы данных высокой доступности по своей природе, они подчиняются некоторым общим требованиям. Клиенты службы данных высокой доступности исходят из неизменности ее сетевого адреса и имени хоста и рассчитывают, что они будут посылать свои требования в неизменном виде, на каком бы компьютере ни находилась эта служба данных.

Рассмотрим Web-браузер, обращающийся к Web-серверу с высокой степенью доступности. Он отправляет запрос с URL, который содержит имя хоста и путь к файлу на Web-сервере. Браузер ожидает, что имя хоста и путь к файлу не изменятся после сбоя Web-сервера. Если при загрузке файла с Web-сервера происходит сбой, браузеру потребуется передать запрос 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. Регулярно проверяйте компоненты, склонные к отказам (например, диски), чтобы обеспечить их своевременную замену.

Устойчивость к отказам

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.

Кластер высокой доступности - наиболее общее решение для повышения доступности, так как он масштабируем, прост в использовании и относительно недорог.

Понятия, связанные с данным:

- “Высокая доступность в Sun Cluster 3.0” на стр. 210
- “Обеспечение высокой доступности с помощью VERITAS Cluster Server” на стр. 213

Высокая доступность в Sun Cluster 3.0

В этом разделе приведена обзорная информация о том, как достигается высокая доступность при работе DB2® с Sun Cluster 3.0. В частности, здесь можно найти описание агента высокой доступности, действующего как посредник между двумя программными продуктами (смотрите рис. 20).

Рисунок 20. DB2, Sun Cluster 3.0 и высокая доступность. Взаимосвязь между DB2, Sun Cluster 3.0 и агентом высокой доступности.

Автоматический перенос ресурсов

Sun Cluster 3.0 обеспечивает высокую доступность путем автоматического переноса прикладной программы в случае сбоя. Все узлы периодически опрашиваются, и программное обеспечение кластера автоматически перемещает известную ему прикладную программу с основного узла, на котором произошел сбой, на указанный дополнительный узел. В процессе автоматического переноса ресурсов может возникнуть небольшой перерыв в обслуживании клиентов. Кроме того, клиентам может потребоваться повторно подключиться к серверу.

Однако клиенты не будут знать, к какому физическому серверу они подключаются для работы с прикладной программой и данными. Позволяя другим узлам кластера автоматически принимать на себя нагрузку в случае сбоя основного узла, Sun Cluster 3.0 значительно сокращает время простоя и повышает продуктивность работы.

Общие диски

Для работы Sun Cluster 3.0 необходимы общие диски. Такие диски можно подключить сразу к нескольким узлам. В среде Sun Cluster 3.0 эта возможность обеспечивает высокую доступность дисков. Диски, относящиеся к общей памяти, остаются доступными в случае сбоя отдельного узла, поскольку к ним по-прежнему можно обратиться через другой узел сервера. К общим дискам можно обратиться на глобальном уровне через основной узел. Если на узле, через который клиент обращается к данным, происходит сбой, запросы этого клиента перенаправляются на другой узел, который напрямую подключен к тем же дискам. Для обеспечения избыточности данных при работе с общими дисками администратор томов позволяет использовать зеркальные копии или RAID 5. В настоящий момент Sun Cluster 3.0 поддерживает администраторы томов Solstice DiskSuite и VERITAS Volume Manager. Применение общих дисков в сочетании с зеркальной защитой и чередованием данных обеспечивает защиту как от сбоев узлов, так и от сбоев отдельных дисков.

Глобальные устройства

Глобальные устройства - это устройства высокой доступности, к которым можно обратиться с любого узла кластера, независимо от физического расположения устройства. Все диски настраиваются как глобальные устройства и им назначается ID (DID), относящийся к пространству глобальных имен. Следовательно, диски доступны на любом узле кластера.

Глобальные файловые системы

Файловая система кластера, или глобальная файловая система, является промежуточным звеном между ядром одного узла и администратором томов базовой файловой системы другого узла, к которому физически подключен один или несколько дисков. Файловые системы кластера зависят от глобальных устройств, физически подключенных к одному или нескольким узлам. Однако они не зависят от базовой файловой системы и администратора томов. В настоящее время файловую систему кластера можно создать в UFS с помощью Solstice DiskSuite или VERITAS Volume Manager. Данные доступны всем узлам только в том случае, если файловые системы, расположенные на дисках, смонтированы на глобальном уровне как файловая система кластера.

Группа устройств

Для управления всеми общими дисками должна применяться среда Sun Cluster. Вначале на общем диске создаются группы дисков, для управления которыми применяется Solstice DiskSuite или VERITAS Volume Manager. Затем эти группы регистрируются как группы дисков Sun Cluster. Группа дисков - это одна из разновидностей глобальных устройств. Группы общих дисков обладают высокой доступностью. В случае сбоя узла, который в настоящее применяется для работы с группой дисков, существует альтернативный путь доступа к дискам. В результате возникновения такого сбоя группа дисков становится недоступна лишь на время, необходимое для восстановления и проверки целостности. В течение этого времени все запросы блокируются (незаметно для прикладной программы), пока группа дисков снова не станет доступной.

Менеджер групп ресурсов (RGM)

RGM выполняется на каждом узле кластера и предоставляет один из способов обеспечения высокой доступности. Он автоматически запускает и останавливает ресурсы на выбранных узлах в соответствии с заранее настроенной стратегией. RGM обеспечивает высокую доступность ресурса путем его перезагрузки в случае сбоя узла, то есть завершения работы ресурса на неисправном узле и его запуска на другом узле. Кроме того, RGM автоматически включает и выключает мониторы ресурсов, которые отслеживают сбои ресурсов и перемещают ресурсы, в работе которых возник сбой, на другой узел.

Службы данных

Службой данных называется прикладная программа другой фирмы, которая настроена для работы в кластере, а не на отдельном сервере. Служба данных включает в себя саму прикладную программу и программу Sun Cluster 3.0, которая запускает, останавливает и отслеживает работу прикладной программы. Sun Cluster 3.0 предоставляет методы службы данных, предназначенные для управления работой прикладной программы в кластере и отслеживания ее работы. Эти методы выполняются под управлением Менеджера групп ресурсов (RGM), который с их помощью запускает, останавливает и отслеживает работу прикладных программ на узлах кластера. Такие методы в сочетании с программным обеспечением кластера и общими дисками позволяют прикладным программам работать как службы данных с высокой доступностью. Это означает, что в случае отдельного сбоя в кластере работа прикладной программы не прерывается на длительное время. При этом не важно, где произошел сбой: на узле, в компоненте интерфейса или в самой прикладной программе. Кроме того, RGM управляет ресурсами кластера, в том числе сетевыми ресурсами (именами логических хостов и совместно используемыми адресами) и экземплярами прикладных программ.

Тип ресурса, ресурс и группа ресурсов

Тип ресурса определяется следующими факторами:

1. Прикладной программой, которая будет выполняться в кластере.
2. Управляющими программами, которые применяются RGM в качестве методов обратного вызова для управления прикладными программами как ресурсами кластера.
3. Набором свойств, относящихся к статической конфигурации кластера.

Свойства типа применяются RGM для управления ресурсами этого типа.

Ресурс наследует свойства и значения от своего типа. Он представляет собой экземпляр исходной прикладной программы, выполняемой в кластере. Каждому экземпляру должно быть назначено имя, уникальное в кластере. Каждый ресурс должен относиться к какой-либо группе ресурсов. RGM одновременно включает и выключает все ресурсы группы на одном узле. При включении или выключении группы ресурсов RGM вызывает методы обратного вызова для отдельных ресурсов группы.

Узлы, на которых группа ресурсов включена, называются основными узлами. Группа ресурсов обслуживается всеми основными узлами. С каждой группой ресурсов связано свойство Nodelist. Оно задается администратором кластера и содержит список всех возможных основных узлов группы ресурсов.

Более подробная информация о реализации и структуре среды IBM® DB2 Universal Database на платформе Sun Cluster 3.0 приведена в документе ″DB2 and High Availability on Sun Cluster 3.0″, который можно найти на Web-сайте ″DB2 UDB and DB2 Connect™ Support″ (http://www.ibm.com/software/data/pubs/papers/).

Понятия, связанные с данным:

- “Высокая доступность в операционной среде Solaris” на стр. 207
- “Обеспечение высокой доступности с помощью VERITAS Cluster Server” на стр. 213

Обеспечение высокой доступности с помощью VERITAS Cluster Server

VERITAS Cluster Server позволяет избежать запланированных и незапланированных простоев. Он способствует консолидации серверов и эффективно управляет большим количеством прикладных программ в неоднородной среде. VERITAS Cluster Server поддерживает кластеры, содержащие до 32 узлов, и может работать как в сети хранения данных (SAN), так и в традиционной среде клиент-сервер. С помощью VERITAS Cluster Server можно обеспечить защиту любого ресурса, начиная от одного наиболее важного экземпляра базы данных и заканчивая крупным кластером с большим количеством прикладных программ в среде с сетевой памятью. В этом разделе приведено краткое описание функций VERITAS Cluster Server.

Требования к аппаратному обеспечению

Ниже приведен список аппаратного обеспечения, которое в настоящий момент поддерживается VERITAS Cluster Server:

- Узлы сервера:
  - Любой сервер SPARC/Solaris фирмы Sun Microsystems с операционной системой Solaris 2.6 или более новой и оперативной памятью размером не менее 128 Мб.
- Дисковая память:
  - EMC Symmetrix, IBM® Enterprise Storage Server, HDS 7700 и 9xxx, Sun T3, Sun A5000, Sun A1000, Sun D1000 и любая другая дисковая память, которая поддерживается VCS версии 2.0 или более новой. Информацию о поддерживаемых дисковых подсистемах можно получить у сотрудника сервисного представительства VERITAS или в документации по VCS.
  - В стандартной среде на каждом узле кластера потребуются зеркальные частные диски для двоичных дисков с данными DB2® UDB и дисков, общих для нескольких узлов.
- Соединения по сети:
  - Для подключения к внешней сети могут применяться любые сетевые соединения, поддерживающие адресацию IP.
  - Для соединений контроля за пульсом (внутренних по отношению к кластеру) требуются резервные соединения. Для выполнения этого требования можно установить по два дополнительных контроллера Ethernet на каждом сервере, либо по одному контроллеру Ethernet на каждом сервере и один общий GABdisk для всего кластера

Требования к программному обеспечению

Допустимы следующие компоненты программного обеспечения VERITAS:

- VERITAS Volume Manager версии 3.2 или более новой версии, VERITAS File System версии 3.4 или более новой версии, VERITAS Cluster Server версии 2.0 или более новой версии.
- Выпуск базы данных для DB2 for Solaris версии 1.0 или более новой версии.

Хотя администратор томов не требуется для работы VERITAS Cluster Server, настоятельно рекомендуется установить VERITAS Volume Manager, так как он значительно упрощает процедуры установки, настройки и управления ресурсами.

Восстановление после сбоя

VERITAS Cluster Server повышает доступность кластера, обеспечивая доступность служб прикладных программ, таких как DB2 UDB, путем автоматического переноса прикладных программ в случае сбоя. Для этого регулярно проверяется состояние отдельных узлов кластера и связанных с ним

служб программ. При возникновении сбоя, нарушающего работу службы прикладной программы (в данном случае - службы DB2 UDB), VERITAS Cluster Server и/или VCS HA-DB2 Agent автоматически восстанавливает работу службы. Для этого может потребоваться перезапустить DB2 UDB на том же узле, либо переместить DB2 UDB на другой узел кластера, а затем перезапустить его на этом узле. Если прикладную программу требуется перенести на другой узел, VERITAS Cluster Server перемещает на новый узел всю информацию, связанную с этой программой (т..е. сетевой IP-адрес и идентификатор владельца исходной памяти). В результате перенос службы на другой узел останется незамеченным для пользователей. Пользователи будут по-прежнему обращаться к службе по старым IP-адресам, однако теперь эти адреса будут связаны с другим узлом кластера.

Во время восстановления после сбоя с помощью VERITAS Cluster Server перерыв в работе службы не обязательно будет заметен для пользователей. Это зависит от типа соединения (с сохранением или без сохранения состояния), установленного между клиентом и службой прикладной программы. В случае прикладной программы, применяющей соединение с сохранением состояния (например, DB2 UDB), может возникнуть небольшой перерыв в обслуживании пользователей. Кроме того, пользователям может потребоваться повторно подключиться к серверу после завершения процедуры восстановления. В случае прикладной программы, применяющей соединение без сохранения состояния (например, NFS), может возникнуть небольшая задержка в обслуживании пользователей, однако в общем случае сбой не будет ощутим для пользователей, и им не потребуется повторно входить в систему.

Представление прикладной программы в виде службы, которую можно автоматически переносить между узлами кластера, позволяет VERITAS Cluster Server не только сократить время незапланированных простоев, но и сократить продолжительность запланированных перерывов в работе (например, для обслуживания или обновления системы). Процедуру восстановления после сбоя можно запустить вручную. Если на узле требуется обновить аппаратное обеспечение или операционную систему, DB2 UDB можно перенести на другой узел кластера, а после выполнения обновления вернуть обратно на исходный узел.

В среде кластера такого типа рекомендуется применять прикладные программы, устойчивые к сбоям. Прикладная программа устойчива к сбоям, если она может восстановить свою работу после непредвиденного сбоя, сохранив при этом целостность принятых данных. Такие программы иногда называются прикладными программами *с поддержкой кластеров*. DB2 UDB - это прикладная программа, устойчивая к сбоям.

Общая память

Если Veritas Cluster Server применяется совместно с VCS HA-DB2 Agent, то для его работы требуется общая память. Общая память - это память, которая физически подключена к нескольким узлам кластера. Диски, размещенные в общей памяти, устойчивы к сбоям отдельных узлов, так как после возникновения такого сбоя к дискам по-прежнему можно обратиться через другой узел кластера (или несколько альтернативных узлов).

С помощью VERITAS Cluster Server для работы с общей памятью узлы кластера могут применять такую логическую структуру, как группа дисков. Группа дисков - это набор логических дисков, владельца которых можно автоматически изменять на другой узел кластера. В каждый момент времени группа дисков может быть импортирована только на один узел. Например, если группа дисков A импортирована на узел 1, и на узле 1 возник сбой, то группу дисков A можно экспортировать на этом узле и импортировать ее на другой узел кластера. VERITAS Cluster Server может управлять сразу несколькими группами дисков в одном кластере.

Помимо создания групп дисков, администратор томов позволяет обеспечить избыточность данных в общей памяти с помощью зеркальных копий или RAID 5. В качестве администратора логических томов VERITAS Cluster Server позволяет использовать VERITAS Volume Manager или Solstice DiskSuite. Применение общей памяти совместно с зеркальной защитой и чередованием данных обеспечивает защиту как от сбоев узлов, так и от сбоев отдельных дисков и контроллеров.

Функции Global Atomic Broadcast (GAB) и Latency Transport (LLT) сервера VERITAS Cluster Server

У узлов кластера должна быть возможность взаимодействовать друг с другом для обмена информацией о состоянии аппаратного и программного обеспечения, обновления списка членов кластера и синхронизации этой информации. Функция Global Atomic Broadcast (GAB), основанная на Low Latency Transport (LLT), предоставляет способ обмена информацией между узлами с высокой скоростью и маленькой задержкой, который применяется VERITAS Cluster Server. GAB загружается в виде модуля ядра на все узлы кластера и предоставляет механизм атомарного оповещения, гарантирующий, что все узлы одновременно получат информацию об изменении состояния.

Используя функции взаимодействия ядра с другим ядром, LLT обеспечивает высокоскоростной транспорт с маленькой задержкой для всей информации, которая передается и синхронизируется узлами кластера. GAB работает на основе LLT. VERITAS Cluster Server не использует для передачи пульса протокол IP. Вместо него можно выбрать один из двух надежных протокола связи. Для передачи пульса можно настроить GAB с LLT, либо GABdisk, обеспечивающий передачу пульса на основе диска. Пульс должен передаваться по основному и резервному соединению. Для этого можно настроить два частных соединения

Ethernet между узлами кластера, либо одно частное соединение Ethernet и одно соединение GABdisk. Два соединения GABdisks использовать нельзя, так как информация о состоянии кластера должна передаваться между узлами по частному соединению Ethernet.

За дополнительной информацией о функциях GAB и LLT, а также их настройке в конфигурации VERITAS Cluster Server, обратитесь к руководству VERITAS Cluster Server 2.0 User's Guide for Solaris.

Стандартные и специальные агенты

Агентом называется программа, предназначенная для обеспечения доступности отдельного ресурса или прикладной программы. При запуске агент получает от VCS необходимую информацию о конфигурации. После этого он периодически отслеживает состояние ресурса или прикладной программы и обновляет информацию о состоянии в VCS. В общем случае агенты применяются для включения и выключения ресурсов, а также контроля за их состоянием. Обычно агенты предоставляют службы четырех типов: запуск, остановка, отслеживание и очистка. Запуск и остановка применяются для включения и выключения ресурса, отслеживание применяется для проверки состояния отдельного ресурса или прикладной программы, а очистка применяется во время восстановления.

Вместе с VERITAS Cluster Server поставляется и устанавливается большое количество стандартных агентов. Стандартные агенты - это процессы VCS, управляющие предопределенными типами ресурсов, которые обычно используются в конфигурации кластера (например, IP, смонтированные ресурсы, процессы и общие ресурсы). Такие агенты значительно упрощают процедуру установки и настройки кластера. В состав VERITAS Cluster Server входит около 20 стандартных агентов.

Специальные агенты предназначены для отдельных прикладных программ, например, DB2 UDB. Примером специального агента может служить VCS HA-DB2 Agent, который взаимодействует с VCS с помощью среды VCS Agent.

Ресурсы, типы ресурсов и группы ресурсов VCS

Тип ресурса - это определение объекта, идентифицирующее ресурсы, состояние которых будет отслеживаться в кластере VCS. Определение типа ресурса включает в себя имя типа и набор свойств ресурсов этого типа, важных с точки зрения обеспечения высокой доступности. Ресурс наследует свойства и значения от своего типа. Имя ресурса не должно совпадать с именем другого ресурса в кластере.

Существует два типа ресурсов: постоянные и стандартные (непостоянные). Постоянными называются такие ресурсы, состояние которых отслеживается, но которые не включаются и не выключаются средствами VCS. Примером такого

ресурса может служить контроллер сетевого интерфейса (NIS). Стандартными называются ресурсы, которые включаются и выключаются средствами VCS.

Ресурс - это объект самого нижнего уровня, состояние которого отслеживается. Существуют разные типы ресурсов (например, общие, смонтированные и т.д.). Каждый ресурс должен входить в группу ресурсов. VCS одновременно включает и выключает все ресурсы группы. Для этого VCS вызывает метод запуска или завершения работы для каждого ресурса группы. Существует два типа групп ресурсов: параллельные и с возможностью восстановления после сбоя. В любой конфигурации DB2 UDB с высокой степенью доступности, в том числе в конфигурации с несколькими разделами, должны применяться группы ресурсов с возможностью восстановления после сбоя.

″Основной″ или ″главный″ узел - это узел, который потенциально может стать владельцем ресурса. Список узлов кластера, которые могут стать главными узлами группы ресурсов, хранится в атрибуте группы ресурсов `systemlist`. Если кластер состоит из двух узлов, то, как правило, оба узла указываются в `systemlist`. В кластерах с большим количеством узлов, обслуживающих несколько прикладных программ с высокой доступностью, может требоваться, чтобы некоторые службы прикладных программ (определенные на нижнем уровне как ресурсы) никогда не прерывали свою работу на отдельных узлах.

Можно определить зависимости между группами ресурсов, которые будут учитываться VERITAS Cluster Server при оценке последствий сбоя различных ресурсов и управлении восстановлением. Например, если группу ресурсов ClientApp1 нельзя включить, пока не будет запущена группа ресурсов DB2, то считается, что группа ресурсов ClientApp1 зависит от группы ресурсов DB2.

Более подробная информация о реализации и структуре среды IBM DB2 Universal Database с высокой доступностью, использующей VERITAS Cluster Server, приведена в документе ″DB2 UDB and High Availability with VERITAS Cluster Server″. Для просмотра этого документа перейдите на Web-сайт http://www.ibm.com/support и выполните поиск по ключевому слову ″1045033″.

Понятия, связанные с данным:

- “Высокая доступность в операционной среде Solaris” на стр. 207
- “Высокая доступность в Sun Cluster 3.0” на стр. 210

Часть 3. Приложения

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.

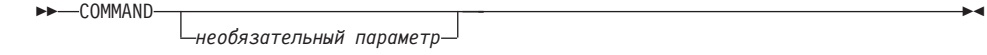
Параметры делятся на ключевые слова и переменные:

- 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:



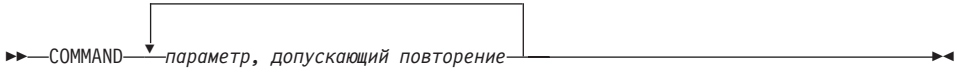
Значения параметров по умолчанию приведены над путем:

Вертикальный ряд параметров, где первый параметр показан на главном пути, указывает на то, что должен быть выбран один из параметров:

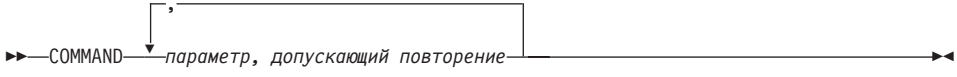
Вертикальный ряд параметров, где первый параметр показан под главным путем, указывает на то, что можно выбрать один из параметров:

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:



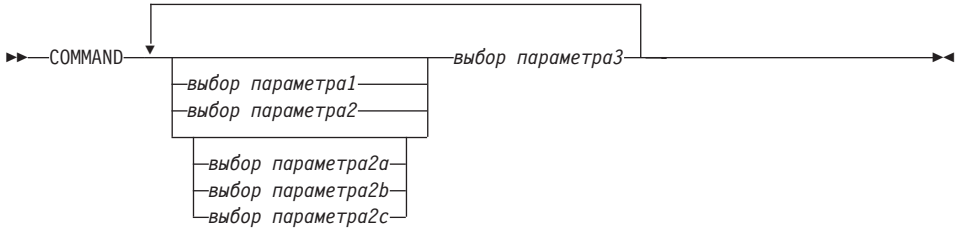
- Если стрелка содержит запятую, элемент может повторяться в списке, где элементы разделены запятыми:



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 операций

В сообщения SQL включаются также сообщения различных утилит. 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.

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"
```

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

Системные команды

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 операционных системах на базе UNIX эта утилита расположена в каталоге sqllib/adsm. В Windows - в каталоге sqllib\bin.

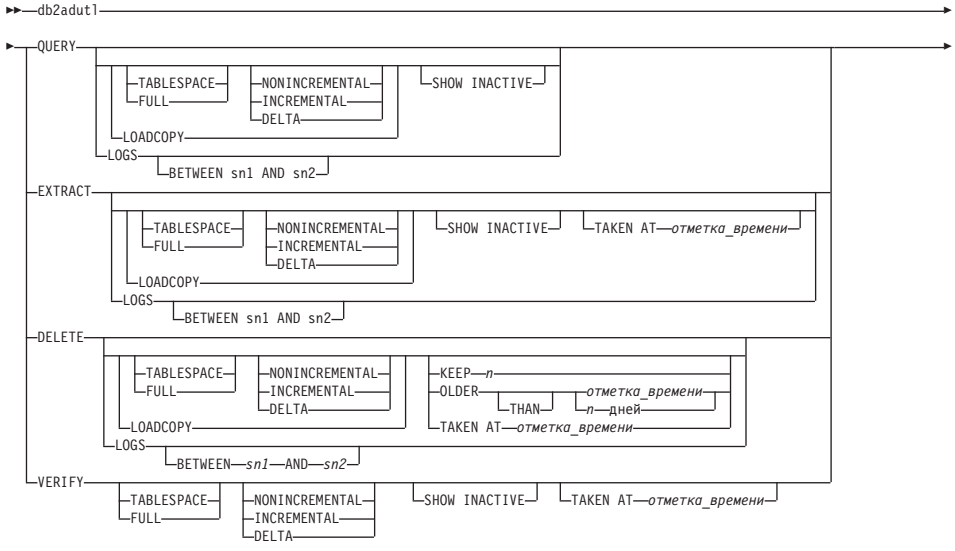
Авторизация:

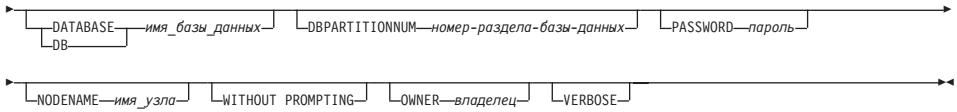
Нет

Обязательное 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.

NONINCREMENTAL
Включает только образы неинкрементальных резервных копий.

INCREMENTAL
Включает только образы инкрементальных резервных копий.

DELTA
Включает только образы инкрементальных разностных резервных копий.

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.

DBPARTITIONNUM *номер-раздела-базы-данных*

Применять операцию только к тем объектам, которые были созданы разделом 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.

VERBOSE

Показывает дополнительную информацию о файле

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
```

Замечания по использованию:

Для ограничения типов образов резервных копий, которые должны быть включены в операцию, можно указать по одному параметру из каждой показанной ниже группы:

Масштаб:

- FULL - Включает только образы резервных копий баз данных.
- TABLESPACE - Включает только образы резервных копий табличных пространств.

Кумулятивность:

- NONINCREMENTAL - Включает только образы неинкрементальных резервных копий.
- INCREMENTAL - Включает только образы инкрементальных резервных копий.
- DELTA - Включает только образы инкрементальных разностных резервных копий.

Совместимость:

Для совместимости с более ранней версией, чем версия 8:

- В параметре DBPARTITIONNUM можно указать ключевое слово NODE.

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.

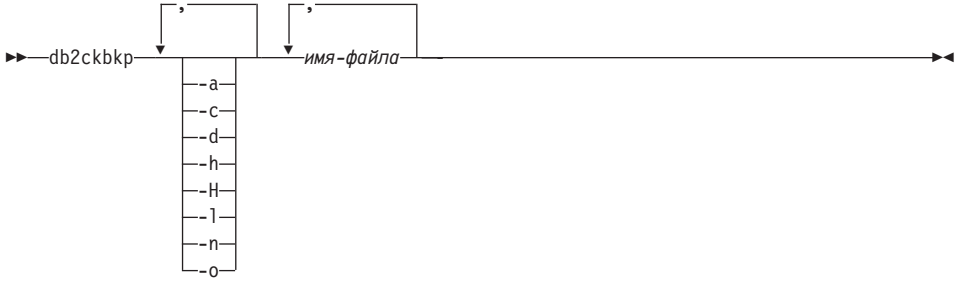
Авторизация:

Доступ к этой утилите может получить любой пользователь, но чтобы запустить ее для конкретной резервной копии, надо иметь разрешение на ее чтение.

Обязательное 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 Показывает информацию заголовка носителя, включая имя и путь имя и путь образа, ожидаемые утилитой восстановления.

-H Показывает ту же информацию, что и ключ -h, но считывает только информацию заголовка носителя, расположенную в первых 4 Кб образа. Образ не проверяется.

Примечание: Эта опция несовместима с другими опциями.

-l Выводит данные заголовков файлов журнала.

-n Запросить монтирование магнитной ленты. Принять одну ленту на устройство.

-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
        Database Partition Number      -- 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 Database Partition No. -- 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:

1. 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.
2. 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
   ```

в Windows:

```
db2ckbkp -d \\.\tape1
```

3. Если образ находится на лентопротяжном устройстве, укажите путь к устройству. Перед монтированием будет показано приглашение, если не была указана опция '-n'. Для нескольких магнитных лент первая лента будет смонтирована на первом указанном устройстве. (В заголовке этой магнитной ленты должно быть указано 001).

   По умолчанию при обнаружении лентопротяжного устройства программа показывает пользователю приглашение для монтирования магнитной ленты. Пользователь может выбрать один из нескольких вариантов. Приглашение и варианты показаны ниже: (`/dev/rmt0` - указанный путь к устройству)

   ```
   Смонтируйте исходный носитель в устройстве /dev/rmt0.
   Продолжить (c), завершить для данного устройства (d),
   прервать работу с инструментом (t)? (c/d/t)
   ```

   Запрос будет показан для каждого указанного устройства, а также при достижении устройством конца магнитной ленты.

Ссылки, связанные с данной темой:

- "db2adutl - Работа с архивными образами TSM" на стр. 227

db2ckrst - Проверить последовательность инкрементного образа восстановления

Запрашивает хронологию базы данных и создает список отметок времени образов резервных копий, необходимых для инкрементного восстановления. Также создается упрощенный синтаксис ручного инкрементного восстановления.

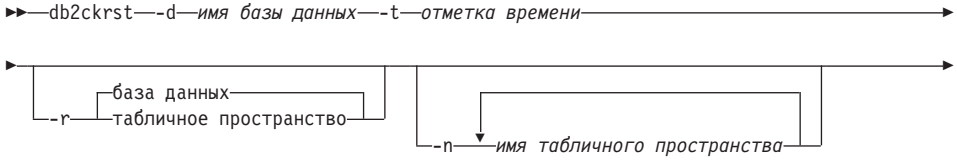
Авторизация:

Нет

Обязательное 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, все остальные опции игнорируются и выводится только справочная информация.

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.

Если указана опция FORCE команды PRUNE HISTORY, появится возможность удалить записи, необходимые для восстановления из более нового, полного образа резервной копии базы данных. По умолчанию команда PRUNE HISTORY не удаляет нужные записи. Опцию FORCE команды PRUNE HISTORY использовать не рекомендуется.

Эта утилита не должна заменять записи о созданных резервных копиях.

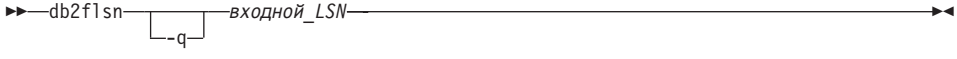
db2flsn - Найти последовательный номера журнала

Возвращает имя файла, содержащего запись журнала, идентифицированную заданным LSN (log sequence number - последовательный номер журнала).

Авторизация:

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:

Контрольный файл заголовка 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

Инициализирует зеркальную копию разделенной базы данных. Зеркальная копия базы данных может быть инициализирована как клон основной базы данных, помещена в состояние ожидания повтора транзакций, или использована в качестве образа резервной копии для восстановления основной базы данных.

Авторизация:

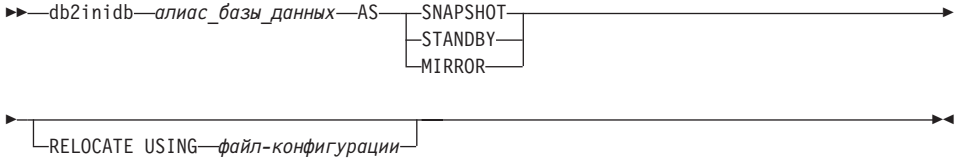
Один из следующих вариантов:

- *sysadm*
- *sysctrl*
- *sysmaint*

Обязательное 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.

STANDBY
Указывает, что база данных будет помещена в состояние отложенного восстановления с повтором транзакций.

Примечание: 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
Указывает, что зеркальная копия базы данных должна применяться в качестве образа резервной копии для восстановления основной базы данных.

RELOCATE USING файл-конфигурации
Указывает, что файлы базы данных должны быть перемещены в соответствии с информацией, указанной в файле конфигурации.

Ссылки, связанные с данной темой:

- “db2relocatedb - Relocate Database Command” в *Command Reference*

db2mscs - Установить утилиту Windows восстановления после сбоя

Создает инфраструктуру для поддержки восстановления DB2 после сбоя в системе Windows с помощью Microsoft Cluster Server (MSCS). Эту утилиту можно использовать для восстановления при отказах как в одноразделной, так и в многораздельной среде.

Авторизация:

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.

-u:экземпляр
Данная опция позволяет отменить действия db2mscs и восстановить состояние экземпляра до установки MSCS.

Замечания по использованию:

Утилита DB2MSCS - это отдельная утилита командной строки, применяемая для преобразования экземпляра без поддержки MSCS в экземпляр с поддержкой MSCS. Эта утилита создает все группы, ресурсы и зависимости MSCS. Кроме того, она также копирует всю информацию DB2 из реестра Windows в раздел реестра, относящийся к кластеру, а также перемещает каталог экземпляра в совместно используемый диск кластера. Утилита DB2MSCS принимает в качестве ввода пользовательский файл конфигурации кластера. Файл DB2MSCS.CFG - это текстовый файл параметров, считываемый утилитой DB2MSCS. Каждый входной параметр указывается на отдельной строке в следующем формате: ПАРАМЕТР=значение. Например:

```
CLUSTER_NAME=FINANCE
GROUP_NAME=Группа DB2
IP_ADDRESS=9.21.22.89
```

Два примера файлов конфигурации находятся в подкаталоге CFG каталога установки DB2. Первый файл, DB2MSCS.EE - это пример для одноразделной базы данных. Второй, DB2MSCS.EEE - пример для многораздельной базы данных.

Параметры файла DB2MSCS.CFG:

DB2_INSTANCE
Имя экземпляра DB2. Этот параметр имеет глобальную область действия и должен быть указан в файле DB2MSCS.CFG только один раз.

DAS_INSTANCE
Имя экземпляра Сервера администратора DB2. Укажите этот параметр для перевода Сервера администратора DB2 в среду MSCS. Этот параметр имеет глобальную область действия и должен быть указан в файле DB2MSCS.CFG только один раз.

CLUSTER_NAME
Имя кластера MSCS. Все ресурсы, перечисленные после этой строки и до следующего параметра CLUSTER_NAME, будут созданы в указанном кластере.

DB2_LOGON_USERNAME
Имя пользователя (учетной записи) службы DB2 в домене (в формате домен\пользователь). Этот параметр имеет глобальную область действия и должен быть указан в файле DB2MSCS.CFG только один раз.

DB2_LOGON_PASSWORD
Пароль учетной записи службы DB2 в домене. Этот параметр имеет глобальную область действия и должен быть указан в файле DB2MSCS.CFG только один раз.

GROUP_NAME
Имя группы MSCS. Если указан этот параметр, и группа MSCS не существует, будет создана новая группа. Если группа уже существует, она будет использована в качестве целевой. Все ресурсы MSCS, перечисленные после этого параметра и до следующего параметра GROUP_NAME, будут созданы в указанной группе или перемещены в эту группу. Задайте этот параметр один раз для каждой группы.

DB2_NODE
Номер раздела (сервера раздела) базы данных, который будет включен в текущую группу MSCS. Если в одной системе существует несколько логических разделов базы данных, для каждого из них должен быть указан отдельный параметр DB2_NODE. Этот параметр должен быть указан после параметра GROUP_NAME, чтобы ресурсы DB2 были созданы в правильной группе MSCS. Для многораздельной базы данных этот параметр обязателен.

IP_NAME
Имя ресурса IP-адреса. Значение параметра IP_NAME произвольно, но должно быть уникальным в кластере. Если задан этот параметр, создается ресурс MSCS типа IP-адрес. Это обязательный параметр для удаленных соединений TCP/IP. Для одораздельной базы данных это необязательный параметр. Рекомендуемое использовать имя хоста, соответствующее IP-адресу.

IP_ADDRESS
IP-адрес ресурса, заданного параметром IP_NAME. Этот параметр обязателен, если указан параметр IP_NAME. Указанный IP-адрес должен быть уникальным в сети.

IP_SUBNET
Маска подсети ресурса, заданного параметром IP_NAME. Этот параметр обязателен, если указан параметр IP_NAME.

IP_NETWORK
Имя сети MSCS, которой принадлежит указанный IP-адрес. Это необязательный параметр. Если он не указан, будет использована первая сеть MSCS, обнаруженная системой. Имя сети MSCS должно быть указана в точности так же, как в разделе Сети программы Управление кластером.

Примечание: Предыдущие четыре параметра совместно задают ресурс IP-адреса.

NETNAME_NAME
Имя ресурса сетевого имени. Этот параметр используется для создания ресурса сетевого имени. Для одноразделной базы данных это необязательный параметр. Этот параметр обязателен в системе, которой принадлежит экземпляр многораздельной базы данных.

NETNAME_VALUE
Значение ресурса Имя сети. Данный параметр обязателен, если указан параметр NETNAME_NAME.

NETNAME_DEPENDENCY
Имя ресурса IP, от которого зависит ресурс Имя сети. Каждый ресурс сетевого имени должен иметь зависимость от ресурса IP-адреса. Это необязательный параметр. Если этот параметр не задан, ресурс сетевого имени будет иметь зависимость от первого ресурса IP в этой группе.

SERVICE_DISPLAY_NAME
Пользовательское имя ресурса Общая служба. Укажите этот параметр для создания ресурса Общая служба.

SERVICE_NAME
Имя службы ресурса Общая служба. Данный параметр обязателен, если указан параметр SERVICE_DISPLAY_NAME.

SERVICE_STARTUP
Необязательный начальный параметр для ресурса Общая служба.

DISK_NAME
Имя ресурса физического диска, который должен быть перемещен в текущую группу. Задайте столько дисковых ресурсов, сколько нужно. Эти дисковые ресурсы уже должны существовать. При настройке экземпляра DB2 утилитой DB2MSCS для поддержки восстановления после сбоя каталог экземпляра копируется в первый диск MSCS этой группы. Для указания другого диска MSCS для каталога экземпляра воспользуйтесь параметром INSTPROF_DISK. Использованное имя диска должно быть указано в точности так же, как в программе Администратор кластера.

INSTPROF_DISK
Необязательный параметр, который задает диск MSCS, содержащий каталог экземпляра DB2. Если этот параметр не указан, утилита DB2MSCS использует первый диск, принадлежащий той же группе.

INSTPROF_PATH
Необязательный параметр, задающий полный путь, в который должен быть скопирован каталог экземпляра. Этот параметр обязателен при использовании дисков IPSHA, дискового ресурса ServerRAID Netfinity (пример: INSTPROF_PATH=p:\db2profs). Параметр INSTPROF_PATH имеет приоритет над параметром INSTPROF_DISK.

TARGET_DRVMAP_DISK
Необязательный параметр, задающий целевой диск MSCS для отображения устройств многораздельной базы данных. Этот параметр задает диск, на котором будет создана база данных, путем отображения его из устройства, заданного командой создания базы данных. Если этот параметр не задан, отображение дисков базы данных надо зарегистрировать вручную с помощью утилиты DB2DRVMP.

DB2_FALLBACK
Необязательный параметр, указывающий, должны ли прикладные программы принудительно выключаться вместе с ресурсом DB2. Если этот параметр не указан, используется значение по умолчанию, равное YES. Для того чтобы приложения не выключались принудительно, укажите в параметре DB2_FALLBACK значение NO.

Команды CLP

ARCHIVE LOG

Закрывает и усекает активный файл журнала восстановимой базы данных. Если включен обработчик пользователя, выдается запрос на архивирование.

Авторизация:

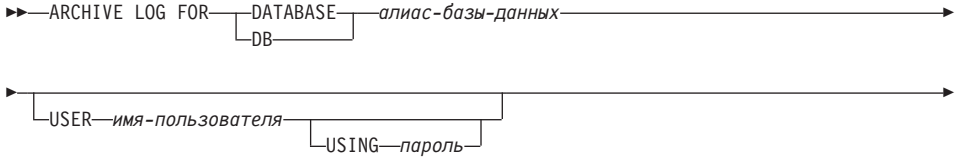
Одни из следующих:

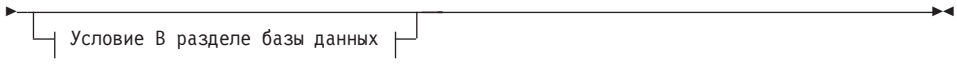
- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Обязательное соединение:

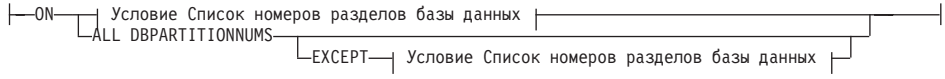
Нет. Эта команда создает новое соединение с базой данных на время выполнения.

Синтаксис команды:

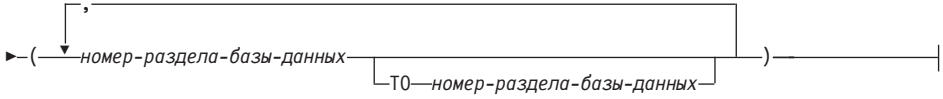




Условие В разделе базы данных:



Условие Список номеров разделов базы данных:



Параметры команды:

DATABASE алиас-базы-данных
Задает алиас базы данных, активный журнал которой должен быть помещен в архив.

USER имя-пользователя
Задает имя пользователя для попытки соединения.

USING пароль
Задает пароль для имени пользователя.

ON ALL DBPARTITIONNUMS
Указывает, что команда должна быть выполнена для всех разделов базы данных, указанных в файле `db2nodes.cfg`. Если условие Номер раздела базы данных не указано, по умолчанию применяется данная опция.

EXCEPT
Указывает, что команда должна быть выполнена для всех разделов базы данных, указанных в файле `db2nodes.cfg`, кроме указанных в списке номеров разделов базы данных.

ON DBPARTITIONNUM/ON DBPARTITIONNUMS
Указывает, что должны быть помещены в архив журналы указанного раздела или набора разделов базы данных.

номер-раздела-базы-данных
Указывает номер раздела базы данных в списке.

TO номер-раздела-базы-данных

Указывается вместе с диапазоном разделов базы данных, журналы которых должны быть помещены в архив. В список номеров баз данных будут включены все разделы базы данных в диапазоне от первого до второго указанного номера 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.

Вызов этой команды возможен только в том случае, если у вызывающей прикладной программы или оболочки нет соединения с указанной базой данных. Таким образом, пользователь не может выполнить команду с непринятыми транзакциями. Команда ARCHIVE LOG не вызывает принудительного приема незаконченных пользовательских транзакций. Если вызывающая прикладная программа или оболочка уже установила соединение с указанной базой данных, выполнение команды будет прервано, и будет возвращено сообщение об ошибке. Если при выполнении команды другая программа производит транзакции с заданной базой данных, возможно небольшое ухудшение производительности, так как команда сбрасывает буфер журнала на диск. Всем остальным транзакциям, пытающимся сохранять в буфер журнальные записи, придется ожидать окончания записи на диск.

Для многораздельной базы данных можно указать подмножество разделов с помощью условия Номер раздела базы данных. Если условие Номер раздела базы данных не указано, по умолчанию команда закрывает и помещает в архив активных журнал каждого раздела базы данных.

Применение этой команды займет часть пространства, выделенного для активных журналов, из-за усечения активного файла журнала. Объем занятого пространства вернется к первоначальному значению после деактивации усеченного журнала. Частое применение этой команды может существенно уменьшить объем свободного пространства для активных журналов, используемого при транзакциях.

Совместимость:

Для совместимости с более ранней версией, чем версия 8:

- Вместо ключевого слова DBPARTITIONNUM можно указать NODE.
- Вместо ключевого слова DBPARTITIONNUMS можно указать NODES.

INITIALIZE TAPE

В операционных системах на базе Windows NT DB2 поддерживает операции резервного копирования и восстановления для лентопротяжных устройств. Эта команда используется для инициализации ленты.

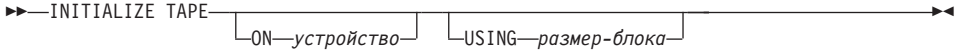
Авторизация:

Нет

Обязательное 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, указываемый для команд BACKUP DATABASE и RESTORE DATABASE, должен делиться на указанный здесь размер 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.

Ссылки, связанные с данной темой:

- “BACKUP DATABASE” на стр. 80
- “RESTORE DATABASE” на стр. 105
- “REWIND TAPE” на стр. 251
- “SET TAPE POSITION” на стр. 251

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. Дополнительные события, фиксируемые в журнале, включают создание, изменение, отбрасывание и переименование табличных пространств, а также реорганизацию, отбрасывание и загрузку таблиц.

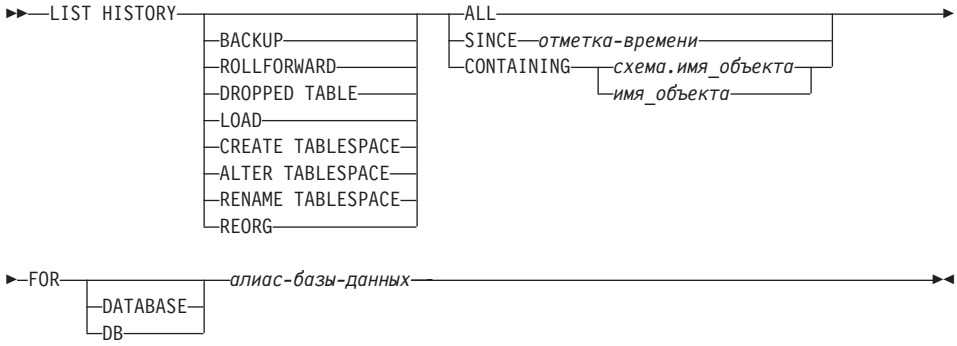
Авторизация:

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.

REORG
Выводит список операций реорганизации.

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 *гггг*мм*дд*чч*нн*сс), 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
```

Примечания 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
   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
```

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).

Авторизация:

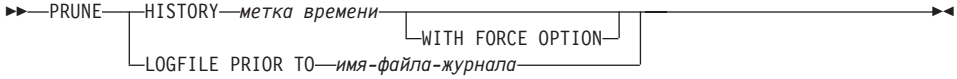
Одни из следующих:

- *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*мм*дд*чч*мм*сс*) 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.

Примечания по использованию:

Сокращение записей резервной копии из файла хронологии приводит к удалению связанных резервных копий файлов на серверах менеджеров связей данных DB2.

REWIND TAPE

В операционных системах на базе Windows NT DB2 поддерживает операции резервного копирования и восстановления для лентопротяжных устройств. Эта команда используется для перемотки ленты.

Авторизация:

Нет

Обязательное соединение:

Нет

Синтаксис команды:

Параметры команды:

ON устройство
Задает допустимое имя ленточного устройства. Значение по умолчанию - `\\.\TAPE0`.

Ссылки, связанные с данной темой:

- “INITIALIZE TAPE” на стр. 246
- “SET TAPE POSITION” на стр. 251

SET TAPE POSITION

В операционных системах на базе Windows NT DB2 поддерживает операции резервного копирования и восстановления для лентопротяжных устройств. Эта команда используется для позиционирования ленты.

Авторизация:

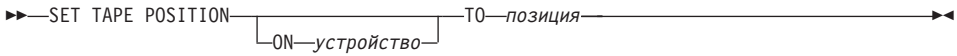
Нет

SET TAPE POSITION

Обязательное 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.

Ссылки, связанные с данной темой:

- “INITIALIZE TAPE” на стр. 246
- “REWIND TAPE” на стр. 251

UPDATE HISTORY FILE

Изменяет положение, тип устройства или комментарий в записи файла хронологии.

Авторизация:

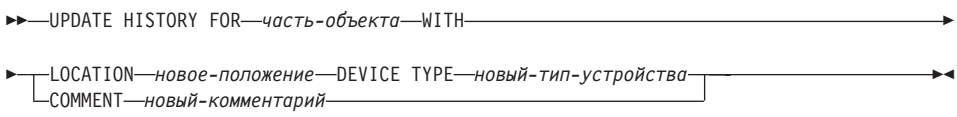
Один из следующих вариантов:

- *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

P Конвейер

N Устройство Null

X XBSA

Q Оператор SQL

O Другое

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
```

Примечания по использованию:

Файл хронологии используется администраторами базы данных для хранения записей. Он используется внутри DB2 при автоматическом восстановлении из инкрементных резервных копий.

Ссылки, связанные с данной темой:

- “PRUNE HISTORY/LOGFILE” на стр. 249

Приложение D. Дополнительные API и связанные с ними структуры данных

db2ArchiveLog - API архивирования активного журнала

Закрывает и усекает активный файл журнала восстановимой базы данных. Если включен обработчик пользователя, посылает требование на архивирование.

Область

В среде MPP этот API закрывает и усекает активные журналы на всех узлах.

Авторизация

Один из следующих вариантов:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Необходимые соединения

Этот API автоматически устанавливает соединение с указанной базой данных. Если это соединение уже существует, возвращается ошибка.

Включаемый файл API

db2ApiDf.h

Синтаксис API 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*.

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.

Примечания 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 - Закончить просмотр файла хронологии

Завершает просмотр файла хронологии и освобождает выделенные для просмотра ресурсы DB2. Обращению к этому API должен предшествовать успешный вызов db2HistoryOpenScan.

Авторизация:

Нет

Необходимое соединение:

Экземпляр. Перед вызовом этого API необязательно вызывать sqleatin.

Включаемый файл API:

db2ApiDf.h

Синтаксис API C:

```
/* File: db2ApiDf.h */
/* API: db2HistoryCloseScan */
/* ... */
SQL_API_RC  SQL_API_FN
  db2HistoryCloseScan (
    db2Uint32 version,
    void *piHandle,
    struct sqlca *pSqlca);
/* ... */
```

Общий синтаксис API:

```
/* File: db2ApiDf.h */
/* API: db2GenHistoryCloseScan */
/* ... */
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 доступа, возвращенный db2HistoryOpenScan.

pSqlca Выходной. Указатель на структуры *sqlca*.

Синтаксис REXX API:

```
CLOSE RECOVERY HISTORY FILE :id-просмотра
```

Параметры REXX API:

id-просмотра
Переменная хоста, содержащая идентификатор просмотра, возвращенный операцией OPEN RECOVERY HISTORY FILE SCAN.

Примечания по использованию:

Подробное описание API работы с файлами хронологии приведено в справке по db2HistoryOpenScan.

Ссылки, связанные с данной темой:

- “db2Prune - Файл хронологии сокращения” на стр. 273
- “db2HistoryUpdate - Обновить файл хронологии” на стр. 270
- “db2HistoryOpenScan - Начать просмотр файла хронологии” на стр. 265
- “db2HistoryGetEntry - Получить следующую запись файла хронологии” на стр. 261
- “SQLCA” в *Administrative API Reference*

Примеры, связанные с данной темой:

- “dbrecov.sqc -- How to recover a database (C)”
- “dbrecov.sqC -- How to recover a database (C++)”

db2HistoryGetEntry - Получить следующую запись файла хронологии

Получает следующую запись файла хронологии. Обращению к этому API должен предшествовать успешный вызов db2HistoryOpenScan.

Авторизация:

Нет

Необходимое соединение:

Экземпляр. Перед вызовом этого API необязательно вызывать sqleatin.

Включаемый файл API:

db2ApiDf.h

Синтаксис API C:

```
/* File: db2ApiDf.h */
/* API: db2HistoryGetEntry */
/* ... */
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:

```
/* File: db2ApiDf.h */
/* API: db2GenHistoryGetEntry */
/* ... */
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 структуры *sqlca*.

iHandle
Входной. Содержит хэндл доступа, возвращенный returned by db2HistoryOpenScan.

iCallerAction
Входной. Задает тип выполняемого действия. 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*.

Синтаксис REXX API:

```
GET RECOVERY HISTORY FILE ENTRY :id-просмотра [USING :значение]
```

Параметры REXX API:

id-просмотра
Переменная хоста, содержащая идентификатор просмотра, который возвращен вызовом OPEN RECOVERY HISTORY FILE SCAN.

значение
Составная переменная REXX хоста, в которую будет помещена информация записи файла хронологии. 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).

Примечания по использованию:

Возвращенные записи будут выбраны с учетом значений, заданных при вызове db2HistoryOpenScan.

Подробное описание API работы с файлами хронологии приведено в справке по db2HistoryOpenScan.

Ссылки, связанные с данной темой:

- “db2Prune - Файл хронологии сокращения” на стр. 273
- “db2HistoryUpdate - Обновить файл хронологии” на стр. 270
- “db2HistoryOpenScan - Начать просмотр файла хронологии” на стр. 265
- “db2HistoryCloseScan - Закончить просмотр файла хронологии” на стр. 260
- “SQLCA” в *Administrative API Reference*
- “db2HistData” на стр. 288

Примеры, связанные с данной темой:

- “dbrecov.sqc -- How to recover a database (C)”
- “dbrecov.sqC -- How to recover a database (C++)”

db2HistoryOpenScan - Начать просмотр файла хронологии

Запускает просмотр файла хронологии.

Авторизация:

Нет

Необходимое соединение:

Экземпляр. Перед вызовом этого API необязательно вызывать sqleatin. Если база данных внесена в каталог как удаленная, производится подключение к удаленному узлу.

Включаемый файл API:

db2ApiDf.h

Синтаксис API C:

```
/* File: db2ApiDf.h */
/* API: db2HistoryOpenScan */
/* ... */
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:

```
/* File: db2ApiDf.h */
/* API: db2GenHistoryOpenScan */
/* ... */
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 структуры *sqlca*.

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. При выходе из API этот параметр содержит число записей файла хронологии, отвечающих указанным условиям.

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. Для получения информации о DDL для записи сразу после извлечения этой записи следует вызвать db2HistoryGetEntry с опцией вызова DB2HISTORY_GET_DDL.

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. Для получения информации о DDL для записи сразу после извлечения этой записи следует вызвать db2HistoryGetEntry с опцией вызова DB2HISTORY_GET_DDL.

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. Затем он может использоваться в db2HistoryGetEntry, и db2HistoryCloseScan.

Синтаксис REXX API:

```
OPEN [BACKUP] RECOVERY HISTORY FILE FOR алиас_базы_данных
[OBJECT имя_объекта] [TIMESTAMP :отметка_времени]
USING :значение
```

Параметры REXX API:

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 REXX хоста, в которую помещается информация о файле хронологии. 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 записей файла хронологии, отвечающих заданным условиям.

Примечания 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, чтобы получить значение *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, чтобы получить данные из файла хронологии.
   - Если db2HistoryGetEntry возвращает SQLCODE `SQL_RC_OK`, то при помощи поля *sqld* структуры *db2HistData* можно определить число возвращенных записей табличного пространства.
   - Если db2HistoryGetEntry возвращает SQLCODE `SQLUH_SQLUHINFO_VARS_WARNING`, значит выделенной памяти было недостаточно для размещения всех табличных пространств, которые DB2 попыталась вернуть; освободите память и повторно выделите структуре *db2HistData* объем памяти, достаточный для размещения записей *oDB2UsedTablespace*, а также присвойте *iDB2NumTablespace* значение *oDB2UsedTablespace*.
   - Если db2HistoryGetEntry возвращает SQLCODE `SQLE_RC_NOMORE`, значит получены все записи файла хронологии.
   - Любой другой SQLCODE означает ошибку.
5. После получения всей информации вызовите db2HistoryCloseScan для освобождения ресурсов, выделенных вызовом db2HistoryOpenScan.

Макрокоманда SQLUHINFOSIZE(*n*) (определенная `sqlutil`) позволяет определить объем памяти, необходимой для структуры *db2HistData* с *n* полями *oTablespace*.

Ссылки, связанные с данной темой:

- “db2Prune - Файл хронологии сокращения” на стр. 273
- “db2HistoryUpdate - Обновить файл хронологии” на стр. 270
- “db2HistoryGetEntry - Получить следующую запись файла хронологии” на стр. 261
- “db2HistoryCloseScan - Закончить просмотр файла хронологии” на стр. 260

- “SQLCA” в *Administrative API Reference*

Примеры, связанные с данной темой:

- “dbrecov.sqc -- How to recover a database (C)”
- “dbrecov.sqC -- How to recover a database (C++)”

db2HistoryUpdate - Обновить файл хронологии

Изменяет положение, тип устройства или комментарий в записи файла хронологии.

Авторизация:

Один из следующих вариантов:

- *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.

Включаемый файл API:

db2ApiDf.h

Синтаксис API C:

```
/* File: db2ApiDf.h */
/* API: db2HistoryUpdate */
/* ... */
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:

```
/* File: db2ApiDf.h */
/* API: db2GenHistoryUpdate */
/* ... */
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 структуры *sqlca*.

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, значение 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, значение 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.

Синтаксис REXX API:

```
UPDATE RECOVERY HISTORY USING :значение
```

Параметры REXX API:

значение
Составная переменная REXX хоста, содержащая информацию о новом расположении записи файла протокола. 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).

Примечания 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 затем передать утилите резервного копирования. Аналогично, если образ

копии загрузки был перенесен, утилите повтора транзакций надо передать информацию о новом положении и типе среды хранения.

Ссылки, связанные с данной темой:

- “db2Rollforward - Повтор транзакций базы данных” на стр. 158
- “db2Prune - Файл хронологии сокращения” на стр. 273
- “db2HistoryOpenScan - Начать просмотр файла хронологии” на стр. 265
- “db2HistoryGetEntry - Получить следующую запись файла хронологии” на стр. 261
- “db2HistoryCloseScan - Закончить просмотр файла хронологии” на стр. 260
- “SQLCA” в *Administrative API Reference*
- “db2Backup - Резервное копирование базы данных” на стр. 86

Примеры, связанные с данной темой:

- “dbrecov.sqc -- How to recover a database (C)”
- “dbrecov.sqC -- How to recover a database (C++)”

db2Prune - Файл хронологии сокращения

Удаляет записи из файла хронологии или файлов журналов в активном пути.

Авторизация:

Один из следующих вариантов:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Необходимое соединение:

База данных. Для удаления записей из файла хронологии какой-либо базы данных, отличной от базы данных по умолчанию, перед вызовом API необходимо установить соединение с этой базой данных.

Включаемый файл API:

db2ApiDf.h

Синтаксис API C:

```
/* Файл: db2ApiDf.h */
/* API: db2Prune */
/* ... */
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: db2GenPrune */
/* ... */
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 структуры *sqlca*.

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 REXX API:

```
PRUNE RECOVERY HISTORY BEFORE :отметка-времени [WITH FORCE OPTION]
```

Параметры REXX API:

отметка-времени
Переменная хоста, содержащая отметку времени. Из файла хронологии будут удалены все записи с отметкой времени, предшествующей или равной указанной.

WITH FORCE OPTION
Если указана эта опция, то файл хронологии будет сокращен в соответствии со значением метки времени, даже если при этом будут удалены 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.

Примечания 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.

Ссылки, связанные с данной темой:

- “db2HistoryUpdate - Обновить файл хронологии” на стр. 270
- “db2HistoryOpenScan - Начать просмотр файла хронологии” на стр. 265
- “db2HistoryGetEntry - Получить следующую запись файла хронологии” на стр. 261
- “db2HistoryCloseScan - Закончить просмотр файла хронологии” на стр. 260
- “SQLCA” в *Administrative API Reference*

Примеры, связанные с данной темой:

- “dbrecov.sqc -- How to recover a database (C)”
- “dbrecov.sqC -- How to recover a database (C++)”

db2ReadLogNoConn - Прочитать журнал без подключения к базе данных

Позволяет извлечь записи из журналов базы данных DB2 UDB и запросить у менеджера журналов информацию о текущем состоянии журналов. Перед вызовом этого API выделите с помощью db2ReadLogNoConnInit память, которая будет передана данному API. После вызова этого API освободите память с помощью db2ReadLogNoConnTerm.

Авторизация:

Одни из следующих:

- *sysadm*
- *dbadm*

db2ReadLogNoConn - Прочитать журнал без подключения к базе данных

Необходимое соединение:

База данных

Включаемый файл API:

db2ApiDf.h

Синтаксис C API:

```
/* Файл: db2ApiDf.h */
/* API: db2ReadLogNoConn */
/* ... */
SQL_API_RC SQL_API_FN
db2ReadLogNoConn (
  db2Uint32 versionNumber,
  void *pDB2ReadLogNoConnStruct,
  struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2ReadLogNoConnStruct
{
  db2Uint32                   iCallerAction;
  SQLU_LSN                    *piStartLSN;
  SQLU_LSN                    *piEndLSN;
  char                        *poLogBuffer;
  db2Uint32                   iLogBufferSize;
  char                        *piReadLogMemPtr;
  db2ReadLogNoConnInfoStruct  *poReadLogInfo;
} db2ReadLogNoConnStruct;

typedef SQL_STRUCTURE db2ReadLogNoConnInfoStruct
{
  SQLU_LSN                    firstAvailableLSN;
  SQLU_LSN                    firstReadLSN;
  SQLU_LSN                    nextStartLSN;
  db2Uint32                   logRecsWritten;
  db2Uint32                   logBytesWritten;
  db2Uint32                   lastLogFullyRead;
  db2TimeOfLog                currentTimeValue;
} db2ReadLogNoConnInfoStruct;

/* ... */
```

Параметры API:

version
Входной. Задает уровень версии и выпуска для структуры, передаваемой во втором параметре - *pDB2ReadLogNoConnStruct*.

pParamStruct
Входной. Указатель на структуру *db2ReadLogNoConnStruct*.

pSqlca Выходной. Указатель на структуры *sqlca*.

iCallerAction
Входной. Задает выполняемую операцию. Допустимые значения:

DB2READLOG_READ
Читает журнал базы данных с начального до конечного номеров последовательности журнала и возвращает записи журнала в этом диапазоне.

DB2READLOG_READ_SINGLE
Читает одну запись журнала (распространяемую или нет), идентифицируемую начальным номером последовательности журнала.

DB2READLOG_QUERY
Запрашивает журнал базы данных. Результаты запроса будут возвращены в структуре db2ReadLogNoConnInfoStruct.

piStartLSN
Входной. Начальный номер последовательности журнала задает относительный начальный адрес в байтах для чтения журнала. Это значение должно быть началом действительной записи журнала.

piEndLSN
Входной. Конечный номер последовательности журнала задает относительный конечный адрес в байтах для чтения журнала. Это значение должно быть больше, чем *piStartLsn*, и не обязательно должно соответствовать концу действительной записи журнала.

poLogBuffer
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.

iLogBufferSize
Входной. Указывает размер буфера журнала в байтах.

piReadLogMemPtr
Входной. Блок памяти размера iReadLogMemoryLimit, выделенный при инициализации. В этой памяти хранятся постоянные данные, которые требуются API при каждом вызове. Вызывающая программа не может выделить повторно или изменить этот блок памяти.

poReadLogInfo
Выходной. Указатель на структуру *db2ReadLogNoConnInfoStruct*.

firstAvailableLSN
Первый доступный LSN в доступных журналах.

firstReadLSN
Первый LSN, прочитанный в этом вызове.

nextStartLSN
Следующий читаемый LSN.

logRecsWritten
Число записей журнала, занесенных в поле буфера журнала *poLogBuffer*.

logBytesWritten
Число байтов, занесенных в поле буфера журнала *poLogBuffer*.

lastLogFullyRead
Число, задающее последний полностью прочитанный файл журнала.

Замечания по использованию:

Для API db2ReadLogNoConn требуется блок памяти, который можно выделить с помощью API db2ReadLogNoConnInit. Блок памяти должен быть передан в качестве входного параметра во все последующие вызовы API db2ReadLogNoConn; изменять его нельзя.

При последующих операциях чтения журнала API потребует диапазон последовательных номеров журнала (LSN) и выделенную память. API вернет последовательность записей журнала на основе опции фильтрации, указанной при инициации, и диапазона LSN. Структура информации о чтении журнала в запросе будет содержать правильный начальный LSN, используемый при чтении. Значение, используемое при чтении в качестве конечного LSN, может быть одним из следующих:

- Значение, большее startLSN, указанного вызывающей стороной.
- FFFF FFFF FFFF что интерпретируется программой асинхронного чтения журнала как конец доступных журналов.

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 UDB, возвращаемых db2ReadLogNoConn, можно найти в разделе Записи журнала DB2 UDB.

После первого чтения указывайте значение nextStartLSN, возвращенное в db2ReadLogNoConnInfoStruct, для чтения последующих записей. Для чтения следующего блока записей повторите вызов с новым начальным LSN и действительным конечным LSN. Код sqlca SQLU_RLOG_READ_TO_CURRENT означает, что программа чтения журнала выполнила чтение до конца доступных файлов журналов.

db2ReadLogNoConn - Прочитать журнал без подключения к базе данных

После завершения работы с API освободите память с помощью db2ReadLogNoConnTerm.

Ссылки, связанные с данной темой:

- “db2ReadLogNoConnInit - Инициализировать чтение журнала без подключения к базе данных” на стр. 280
- “db2ReadLogNoConnTerm - Прервать чтение журнала без подключения к базе данных” на стр. 282

---

db2ReadLogNoConnInit - Инициализировать чтение журнала без подключения к базе данных

Позволяет выделить память, используемую db2ReadLogNoConn для извлечения записей из журналов базы данных DB2 UDB и запроса у менеджера журналов информации о текущем состоянии журналов.

Необходимое соединение:

База данных

Включаемый файл API:

db2ApiDf.h

Синтаксис C API:

```
/* Файл: db2ApiDf.h */
/* API: db2ReadLogNoConnInit */
/* ... */
SQL_API_RC SQL_API_FN
db2ReadLogNoConnInit (
  db2Uint32 versionNumber,
  void * pDB2ReadLogNoConnInitStruct,
  struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ReadLogNoConnInitStruct
{
  db2Uint32                   iFilterOption;
  char                        *piLogFilePath;
  char                        *piOverflowLogPath;
  db2Uint32                   iRetrieveLogs;
  char                        *piDatabaseName;
  char                        *piNodeName;
  db2Uint32                   iReadLogMemoryLimit;
  char                        **poReadLogMemPtr;
} db2ReadLogNoConnInitStruct;
/* ... */
```

Параметры API:

version
Входной. Задает уровень версии и выпуска для структуры, передаваемой во втором параметре - *pDB2ReadLogNoConnInitStruct*.

pParamStruct
Входной. Указатель на структуру *db2ReadLogNoConnInitStruct*.

pSqlca Выходной. Указатель на структуры *sqlca*.

iFilterOption
Входной. Задает уровень фильтрации записей журнала, используемый при чтении этих записей. Допустимые значения:

DB2READLOG_FILTER_OFF
Читать все записи в указанном диапазоне LSN.

DB2READLOG_FILTER_ON
Читать только те записи из заданного диапазона LSN, которые были отмечены как распространяемые. Это обычное поведение API асинхронного чтения журнала.

piLogFilePath
Входной. Каталог, в котором находятся читаемые файлы журнала.

piOverflowLogPath
Входной. Резервный каталог, в котором могут находиться читаемые файлы журнала.

iRetrieveLogs
Входной. Указывает, нужно ли вызывать обработчик пользователя, если файлы журнала не были найдены ни в одном из указанных каталогов. Допустимые значения:

DB2READLOG_RETRIEVE_OFF
Для получения отсутствующих файлов обработчик пользователя вызывать не нужно.

DB2READLOG_RETRIEVE_LOGPATH
Для получения отсутствующих файлов нужно вызвать обработчик пользователя.

DB2READLOG_RETRIEVE_OVERFLOW
Для получения отсутствующих файлов в заданный каталог нужно вызвать обработчик пользователя.

piDatabaseName
Входной. Имя базы данных, содержащей журналы восстановления. Этот параметр обязателен, если указана опция получения, описанная выше.

piNodeName
Входной. Имя узла, содержащего журналы восстановления. Этот параметр обязателен, если указана опция получения, описанная выше.

iReadLogMemoryLimit
Входной. Максимальное число байт, которое API может выделить внутренними процедурами.

poReadLogMemPtr
Выходной. Выделенный API блок памяти размера *iReadLogMemoryLimit*. В этой памяти хранятся постоянные данные, которые требуются API при каждом вызове. Вызывающая программа не может выделить повторно или изменить этот блок памяти.

Замечания по использованию:

Память, инициализированную с помощью db2ReadLogNoConnInit, изменять нельзя.

Если db2ReadLogNoConn больше использоваться не будет, вызовите db2ReadLogNoConnTerm для освобождения памяти, инициализированной с помощью db2ReadLogNoConnInit.

Ссылки, связанные с данной темой:

- “db2ReadLogNoConn - Прочитать журнал без подключения к базе данных” на стр. 276
- “db2ReadLogNoConnTerm - Прервать чтение журнала без подключения к базе данных” на стр. 282

db2ReadLogNoConnTerm - Прервать чтение журнала без подключения к базе данных

Освобождает память, которая использовалась db2ReadLogNoConn и была выделена с помощью db2ReadLogNoConnInit.

Авторизация:

Одни из следующих прав доступа:

- *sysadm*
- *dbadm*

Необходимое соединение:

База данных

Включаемый файл API:

db2ApiDf.h

Синтаксис C API:

```
/* Файл: db2ApiDf.h */
/* API: db2ReadLogNoConnTerm */
/* ... */
SQL_API_RC SQL_API_FN
db2ReadLogNoConnTerm (
  db2Uint32 versionNumber,
  void * pDB2ReadLogNoConnTermStruct,
  struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ReadLogNoConnTermStruct
{
  char                          **poReadLogMemPtr;
} db2ReadLogNoConnTermStruct;
/* ... */
```

Параметры API:

version
Входной. Задает уровень версии и выпуска для структуры, передаваемой во втором параметре - *pDB2ReadLogNoConnTermStruct*.

pParamStruct
Входной. Указатель на структуру *db2ReadLogNoConnTermStruct*.

pSqlca Выходной. Указатель на структуры *sqlca*.

poReadLogMemPtr
Выходной. Указатель на блок памяти, выделенный при инициализации. Указатель будет очищен и получит значение NULL.

Ссылки, связанные с данной темой:

- “db2ReadLogNoConn - Прочитать журнал без подключения к базе данных” на стр. 276
- “db2ReadLogNoConnInit - Инициализировать чтение журнала без подключения к базе данных” на стр. 280

db2ReadLog - Асинхронное чтение журнала

Извлекает записи из журналов базы данных DB2 UDB и Менеджера журналов для получения текущей информации о состоянии журнала. 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.

Авторизация:

Одни из следующих:

- *sysadm*
- *dbadm*

Необходимое соединение:

База данных

Включаемый файл API:

db2ApiDf.h

Синтаксис API C:

```
/* File: db2ApiDf.h */
/* API: db2ReadLog */
/* ... */
SQL_API_RC SQL_API_FN
  db2ReadLog (
    db2Uint32 versionNumber,
    void *pDB2ReadLogStruct,
    struct sqlca *pSqlca);

typedef SQL_STRUCTURE db2ReadLogStruct
{
  db2Uint32                   iCallerAction;
  SQLU_LSN                    *piStartLSN;
  SQLU_LSN                    *piEndLSN;
  char                        *poLogBuffer;
  db2Uint32                   iLogBufferSize;
  db2Uint32                   iFilterOption;
  db2ReadLogInfoStruct        *poReadLogInfo;

typedef SQL_STRUCTURE db2ReadLogInfoStruct
{
  SQLU_LSN                    initialLSN;
  SQLU_LSN                    firstReadLSN;
  SQLU_LSN                    nextStartLSN;
  db2Uint32                   logRecsWritten;
  db2Uint32                   logBytesWritten;
  SQLU_LSN                    firstReusedLSN;
  db2Uint32                   timeOfLSNReuse;
  db2TimeOfLog                currentTimeValue;
} db2ReadLogInfoStruct;

typedef SQL_STRUCTURE db2TimeOfLog
{
  db2Uint32                   seconds;
  db2Uint32                   accuracy;
} db2TimeOfLog;
/* ... */
```

Параметры API:

versionNumber
Входной. Задает версию и выпуск структуры, переданной во втором параметре, *pDB2ReadLogStruct*.

pDB2ReadLogStruct
Входной. Указатель на *db2ReadLogStruct*.

pSqlca Выходной. Указатель на структуры *sqlca*.

iCallerAction
Входной. Задает выполняемую операцию.

DB2READLOG_READ
Считывает и возвращает записи журнала базы данных с заданного начального до конечного номера.

DB2READLOG_READ_SINGLE
Читает одну запись журнала (распространяемую или нет), идентифицируемую начальным номером последовательности журнала.

DB2READLOG_QUERY
Запрашивает журнал базы данных. Результаты запроса будут возвращены в структуре *db2ReadLogInfoStruct*.

piStartLsn
Входной. Начальный номер последовательности журнала задает относительный начальный адрес в байтах для чтения журнала. Это значение должно быть началом действительной записи журнала.

piEndLsn
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.

poLogBuffer
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.

iLogBufferSize
Входной. Указывает размер буфера журнала в байтах.

iFilterOption
Входной. Задает уровень фильтрации считываемых записей журнала. Допустимые значения:

DB2READLOG_FILTER_OFF
Считывать все записи журнала из заданного диапазона LSN.

DB2READLOG_FILTER_ON
Считывать только записи журнала, лежащие в заданном диапазоне LSN и помеченные как распространяемые. Это традиционный алгоритм работы API асинхронного чтения файлов журналов.

poReadLogInfo
Выходной. Структура, содержащая подробную информацию относительно вызова и журнала базы данных.

Примечания по использованию:

Если запрошенное действие - чтение журнала, вызывающая программа задает диапазон номеров записей журнала и буфер для этих записей журнала. Данный API считывает журнал последовательно, в соответствии с заданным диапазоном LSN, и возвращает записи, связанные с таблицами, у которых в опции DATA CAPTURE указано CHANGES, а также структуру db2ReadLogInfoStruct с текущей информацией об активном журнале. Если в качестве действия выбран запрос, то API возвращает структуру db2ReadLogInfoStruct с текущей информацией об активном журнале.

Чтобы использовать программу асинхронного чтения журнала, сначала запросите в журнале базы данных действительный LSN начала записи. После запроса структура информации о чтении журнала (db2ReadLogInfoStruct) будет содержать допустимый начальный LSN (в initialLSN), который можно указывать в вызове чтения. Значение, используемое при чтении в качестве конечного LSN, может быть одним из следующих:

- Значение, большее initialLSN
- FFFF FFFF FFFF, что интерпретируется программой асинхронного чтения журнала как конец текущего журнала.

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. Описание различных записей журнала DB2, возвращаемых db2ReadLog, приведено в разделе Записи журнала DB2 UDB.

Для считывания следующей записи журнала воспользуйтесь полем nextStartLSN, возвращенным в структуре db2ReadLogStruct. 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.

Ссылки, связанные с данной темой:

- “SQLCA” в *Administrative API Reference*

Примеры, связанные с данной темой:

- “dbrecov.sqc -- How to recover a database (C)”
- “dbrecov.sqC -- How to recover a database (C++)”

db2HistData

Эта структура применяется для возврата информации после вызова db2HistoryGetEntry.

Таблица 2. 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 2. 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`0`) 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.
poTablespace	db2Char	Список имен табличных пространств.

Таблица 2. Поля структуры db2HistData (продолжение)

Имя поля	Тип данных	Описание
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. 3.
oObject	char	Уровень операции: `D` - вся база данных, `P` - табличное пространство, `T` - таблица.
oOptype	char	Смотрите Табл. 4 на стр. 291.
oStatus	char	Состояние записи: `A` - action (действие), `D` - deleted (удалена, для использования в дальнейшем), `E` - expired (истек срок действия), `I` - inactive (неактивна), `N` - not yet committed (еще не принята), `Y` - committed или active (принята или активна), `a` - active backup (активное резервное копирование), но некоторые серверы datalink еще не завершили резервное копирование `i` - inactive backup (неактивное резервное копирование), но некоторые серверы datalink еще не завершили резервное копирование
oDeviceType	char	Тип устройства. Это поле задает способ интерпретации поля *oLocation*: `A` - TSM, `C` - client (клиент), `D` - disk (диск), `K` - diskette (дискета), `L` - local (локальный), `O` - other (другой - для поддержки драйверов устройств других вендоров), `P` - pipe (конвейер), `Q` - cursor (курсор), `S` - server (сервер), `T` - tape (лента), `U` - user exit (пользовательская программа выхода).

Таблица 3. 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

Таблица 3. Допустимые значения oOperation в структуре db2HistData (продолжение)

Значение	Описание	Определение на C	Определение на COBOL/FORTRAN
C	загрузка копии	DB2HISTORY_OP_LOAD_COPY	DB2HIST_OP_LOAD_COPY
D	отбрасывание таблицы	DB2HISTORY_OP_DROPPED_ TABLE	DB2HIST_OP_DROPPED_TABLE
Ф	повтор транзакций	DB2HISTORY_OP_ROLLFWD	DB2HIST_OP_ROLLFWD
G	реорганизация таблицы	DB2HISTORY_OP_REORG	DB2HIST_OP_REORG
X	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	восстановить	DB2HISTORY_OP_RESTORE	DB2HIST_OP_RESTORE
T	изменение табличного пространства	DB2HISTORY_OP_ALT_ TABLESPACE	DB2HIST_OP_ALT_TBS
U	выгрузка	DB2HISTORY_OP_UNLOAD	DB2HIST_OP_UNLOAD

Таблица 4. Допустимые значения oOptype в структуре db2HistData

oOperation	oOptype	Описание	Определение C/COBOL/FORTRAN
B	Ф	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
Ф	E	до конца журналов	DB2HISTORY_OPTYPE_EOL
	P	до момента времени	DB2HISTORY_OPTYPE_PIT
G	Ф	автономная	DB2HISTORY_OPTYPE_OFFLINE
	N	оперативная	DB2HISTORY_OPTYPE_ONLINE

Таблица 4. Допустимые значения oOptype в структуре db2HistData (продолжение)

oOperation	oOptype	Описание	Определение C/COBOL/FORTRAN
X	I	вставка	DB2HISTORY_OPTYPE_INSERT
	R	замена	DB2HISTORY_OPTYPE_REPLACE
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	Ф	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 5. 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 6. Поля структуры db2HistoryEID

Имя поля	Тип данных	Описание
ioNode	SQL_PDB_NODE_TYPE	Номер узла.
ioHID	db2Uint32	ID записи локального файла хронологии.

Синтаксис:

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;
/* ... */
```

Ссылки, связанные с данной темой:

- “db2HistoryGetEntry - Получить следующую запись файла хронологии” на стр. 261
- “SQLCA” в *Administrative API Reference*

SQLU-LSN

Это объединение, используемое в API db2ReadLog, содержит определение номера последовательности 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 7. 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:

Структура на языке C

```
typedef union SQLU_LSN
{
unsigned char  lsnChar  [6] ;
unsigned short lsnWord  [3] ;
} SQLU_LSN;
```

Ссылки, связанные с данной темой:

- “db2ReadLog - Асинхронное чтение журнала” на стр. 283

Приложение E. Пример программы восстановления

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

Примечание: Файлы примера dbrecov находятся в каталогах sqllib/samples/c и sqllib/samples/cpp.

```
/****************************************************************************
** Лицензионные материалы - собственность IBM
** Управляются условиями IBM Public License
**
** (C) COPYRIGHT International Business Machines Corp. 2002
** Все права защищены.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
****************************************************************************
**
** Имя исходного файла: dbrecov.sqc
**
** Пример: Восстановление базы данных
**
** Использованные API DB2:
**        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
**         db2CfgGet -- Получить конфигурацию
**         db2CfgSet -- Задать конфигурацию
**        sqlbmtsq - Запрос табличного пространства
**        sqlbstsc - Задать контейнеры табличного пространства
**        sqlbtcq - Запрос контейнера табличного пространства
**         sqlecrea -- Создать базу данных
**        sqledrpd - Отбросить базу данных
**        sqlefmem - Освободить память
**         db2Backup -- Сохранить базу данных
**         db2Restore -- Восстановить базу данных
**         db2ReadLog -- Асинхронное чтение журнала
**         db2ReadLogNoConn -- Чтение журнала без связи с базой данных
**         db2Rollforward -- Повторить транзакции базы данных
**
```

Программа примера со встроенным SQL (dbrecov.sqc)

```
** Использованные операторы SQL:
**         ALTER TABLE
**         COMMIT
**         DELETE
**         INSERT
**         ROLLBACK
**
** Файл вывода: dbrecov.out (есть в электронной справке)
**
** Подробная информация о сохранении и восстановлении базы данных приведена в
** "Руководстве и справочнике по восстановлению данных и высокой доступности".
** Это руководство поможет определить оптимальные способы восстановления базы
** данных и табличного пространства в вашей среде.
**
** 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 Development Guide".
**
** Дополнительную информацию о языке SQL смотрите в справочнике "SQL Reference".
**
** Дополнительная информация об API DB2 приведена в справочнике Administrative
** API Reference.
**
** Самая свежая информация о программировании, компиляции и запуске программ DB2
** приведена на web-сайте по разработке прикладных 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 *);
int DbReadLogRecordsNoConn(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("\nЭтот пример демонстрирует восстановление базы данных.\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("\nИспользование API DB2:\n");
printf("  db2CfgGet -- Получить конфигурацию\n");
printf("для получения конфигурации базы данных\n");
printf("и определения рабочего каталога сервера.\n");

/* получение рабочего пути сервера */
rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
if (rc != 0)
{
  return rc;
}

printf("\nПримечание: Образы резервных копий будут созданы на сервере\n");
printf("              в каталоге %s,\n", serverWorkingPath);
printf("              и не будут удалены программой.\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;
  }

  rc = DbReadLogRecordsNoConn(dbAlias);

  return 0;
} /* конец функции main */

int ServerWorkingPathGet(char dbAlias[], char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  char serverLogPath[SQL_PATH_SZ + 1];
  db2CfgParam cfgParameters[1];
  db2Cfg cfgStruct;
  int len;

  /* инициализация cfgParameters */
  /* SQLF_DBTN_LOGPATH - это маркер неизменяемого параметра конфигурации
     'logpath' базы данных; он используется для получения пути журнала
     сервера */
  cfgParameters[0].flags = 0;
  cfgParameters[0].token = SQLF_DBTN_LOGPATH;
  cfgParameters[0].ptrvalue =
    (char *)malloc((SQL_PATH_SZ + 1) * sizeof(char));

  /* инициализация cfgStruct */
  cfgStruct.numItems = 1;
  cfgStruct.paramArray = cfgParameters;
  cfgStruct.flags = db2CfgDatabase;
  cfgStruct.dbname = dbAlias;

  /* получение конфигурации базы данных */
  /* API db2CfgGet возвращает значения отдельных записей
     в файле конфигурации базы данных */
  db2CfgGet(db2Version810, (void *)&cfgStruct, &sqlca);
  DB2_API_CHECK("путь к журналу сервера -- получение");

  strcpy(serverLogPath, cfgParameters[0].ptrvalue);
  free(cfgParameters[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;
  db2CfgParam cfgParameters[2];
  db2Cfg cfgStruct;

  printf("\n  Создание пустой базы данных '%s' с тем же кодовым набором,
         что и база данных  '%s'.\n", newDbAlias, existingDbAlias);

  /* инициализация cfgParameters */
  cfgParameters[0].flags = 0;
  cfgParameters[0].token = SQLF_DBTN_TERRITORY;
  cfgParameters[0].ptrvalue = (char *)malloc(10 * sizeof(char));
  memset(cfgParameters[0].ptrvalue, '\0', 10);
  cfgParameters[1].flags = 0;
  cfgParameters[1].token = SQLF_DBTN_CODESET;
  cfgParameters[1].ptrvalue = (char *)malloc(20 * sizeof(char));
  memset(cfgParameters[1].ptrvalue, '\0', 20);

  /* инициализация cfgStruct */
  cfgStruct.numItems = 2;
  cfgStruct.paramArray = cfgParameters;
  cfgStruct.flags = db2CfgDatabase;
  cfgStruct.dbname = existingDbAlias;

  /* получение конфигурации базы данных */
  db2CfgGet(db2Version810, (void *)&cfgStruct, &sqlca);
  DB2_API_CHECK("путь к журналу сервера -- получение");

  /* 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 *)cfgParameters[0].ptrvalue);
  strcpy(countryInfo.sqldblocale, (char *)cfgParameters[1].ptrvalue);

  /* создание базы данных */
  sqlecrea(dbName,
```

```
            dbLocalAlias,
            dbPath,
            &dbDescriptor,
            &countryInfo,
            '\0',
            NULL,
            &sqlca);
  DB2_API_CHECK("База данных -- Создание");

  /* освобождение выделенной памяти */
  free(cfgParameters[0].ptrvalue);
  free(cfgParameters[1].ptrvalue);

  return 0;
} /* DbCreate */

int DbDrop(char dbAlias[])
{
  struct sqlca sqlca;

  printf("\n  Удаление базы данных '%s'.\n", dbAlias);

  /* отбрасывание базы данных и удаление ее из каталога */
  sqledrpd(dbAlias, &sqlca);
  DB2_API_CHECK("База данных -- Удаление");

  return 0;
} /* DbDrop */

int DbBackupAndRestore(char dbAlias[],
                       char restoredDbAlias[],
                       char user[],
                       char pswd[],
                       char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  db2CfgParam cfgParameters[1];
  db2Cfg cfgStruct;
  unsigned short logretain;
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];

  db2BackupStruct  backupStruct;
  db2TablespaceStruct   tablespaceStruct;
  db2MediaListStruct mediaListStruct;
  db2Uint32  backupImageSize;
  db2RestoreStruct restoreStruct;
  db2TablespaceStruct   rtablespaceStruct;
  db2MediaListStruct rmediaListStruct;

  printf("\n*********************************************\n");
  printf("*** Сохранение и восстановление базы данных ***\n");
  printf("*********************************************\n");
  printf("\nИспользование API DB2:\n");
  printf("  db2CfgSet -- Задать конфигурацию\n");
```

```
printf("  db2Backup -- Сохранить базу данных\n");
printf("  db2Restore -- Восстановить базу данных\n");
printf("для сохранения и восстановления базы данных.\n");

printf("\n  Обновление конфигурации базы данных \'%s\':\n", dbAlias);
printf("    - Отключение параметра конфигурации LOGRETAIN \n");
printf("        например, указание LOGRETAIN = OFF/NO\n");

/* инициализация cfgParameters */
/* SQLF_DBTN_LOG_RETAIN - это маркер изменяемого параметра конфигурации
   'logretain' базы данных; он используется для изменения файла
   конфигурации базы данных */
cfgParameters[0].flags = 0;
cfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
cfgParameters[0].ptrvalue = (char *)&logretain;

/* отключение параметра 'logretain' конфигурации базы данных */
logretain = SQLF_LOGRETAIN_DISABLE;

/* инициализация cfgStruct */
cfgStruct.numItems = 1;
cfgStruct.paramArray = cfgParameters;
cfgStruct.flags = db2CfgDatabase | db2CfgDelayed;
cfgStruct.dbname = dbAlias;

/* задание конфигурации базы данных */
db2CfgSet(db2Version810, (void *)&cfgStruct, &sqlca);
DB2_API_CHECK("Сохранение протокола базы данных -- Отключение");

/*******************************/
/*    Сохранение базы данных   */
/*******************************/
printf("\n  Сохранение базы данных '%s'...\n", dbAlias);

tablespaceStruct.tablespaces = NULL;
tablespaceStruct.numTablespaces = 0;

mediaListStruct.locations = &serverWorkingPath;
mediaListStruct.numLocations = 1;
mediaListStruct.locationType = SQLU_LOCAL_MEDIA;

backupStruct.piDBAlias = dbAlias;
backupStruct.piTablespaceList = &tablespaceStruct;
backupStruct.piMediaList = &mediaListStruct;
backupStruct.piUsername = user;
backupStruct.piPassword = pswd;
backupStruct.piVendorOptions = NULL;
backupStruct.iVendorOptionsSize = 0;
backupStruct.iCallerAction = DB2BACKUP_BACKUP;
backupStruct.iBufferSize = 16; /*  16 x 4 Кб */
backupStruct.iNumBuffers = 1;
backupStruct.iParallelism =1;
backupStruct.iOptions = DB2BACKUP_OFFLINE | DB2BACKUP_DB;

/* API db2Backup создает резервную копию базы данных.
```

```
   Этот API автоматически устанавливает соединение с указанной базой данных.
   (Кроме того, этот API можно использовать для создания резервной копии
    табличного пространства). */
db2Backup (db2Version810, &backupStruct, &sqlca);

DB2_API_CHECK("База данных -- Резервное копирование");

while (sqlca.sqlcode != 0)
{
  /* продолжение операции резервного копирования */

  /* в зависимости от значения sqlca.sqlcode может потребоваться */
  /* вмешательство пользователя, например, монтирование новой ленты */

  printf("\n  Продолжение сохранения...\n");

  backupStruct.iCallerAction = DB2BACKUP_CONTINUE;

  db2Backup (db2Version810, &backupStruct, &sqlca);

DB2_API_CHECK("База данных -- Резервное копирование");
}

printf("  Сохранение выполнено.\n");
printf("    - размер резервной копии  : %d Мб\n", backupStruct.oBackupSize);
printf("    - каталог резервной копии : %s\n", mediaListStruct.locations[0]);

printf("    - дата резервной копии    : %s\n", backupStruct.oTimestamp);

/*****************************/
/* Восстановление базы данных */
/*****************************/

strcpy(restoreTimestamp, backupStruct.oTimestamp);


printf("\n  Восстановление базы данных ...\n");
printf("    - алиас исходного образа : %s\n", dbAlias);
printf("    - время исходного образа : %s\n", restoreTimestamp);
printf("    - целевая база данных    : %s\n", restoredDbAlias);

rtablespaceStruct.tablespaces = NULL;
rtablespaceStruct.numTablespaces = 0;

rmediaListStruct.locations = &serverWorkingPath;
rmediaListStruct.numLocations = 1;
rmediaListStruct.locationType = SQLU_LOCAL_MEDIA;

restoreStruct.piSourceDBAlias = dbAlias;
restoreStruct.piTargetDBAlias = restoredDbAlias;

restoreStruct.piTimestamp = restoreTimestamp;
restoreStruct.piTargetDBPath = NULL;
restoreStruct.piReportFile = NULL;
restoreStruct.piTablespaceList = &rtablespaceStruct;
```

```
  restoreStruct.piMediaList = &rmediaListStruct;
  restoreStruct.piUsername = user;
  restoreStruct.piPassword = pswd;
  restoreStruct.piNewLogPath = NULL;
  restoreStruct.piVendorOptions = NULL;
  restoreStruct.iVendorOptionsSize = 0;
  restoreStruct.iParallelism = 1;
  restoreStruct.iBufferSize = 1024; /*  1024 x 4 Кб */;
  restoreStruct.iNumBuffers = 1;
  restoreStruct.iCallerAction = DB2RESTORE_RESTORE;
  restoreStruct.iOptions = DB2RESTORE_OFFLINE | DB2RESTORE_DB |
  DB2RESTORE_NODATALINK | DB2RESTORE_NOROLLFWD;

  /* API db2Restore применяется для восстановления базы данных,
     сохраненной с помощью API db2Backup. */
  db2Restore (db2Version810, &restoreStruct, &sqlca);

  EXPECTED_WARN_CHECK("database restore -- start");

  while (sqlca.sqlcode != 0)
  {
    /* продолжение операции восстановления */
    printf("\n  Продолжение восстановления...\n");

    /* в зависимости от значения sqlca.sqlcode может потребоваться
       вмешательство пользователя, например, монтирование новой ленты */

    restoreStruct.iCallerAction = DB2RESTORE_CONTINUE;

    /* восстановление базы данных */
    db2Restore (db2Version810, &restoreStruct, &sqlca);

    DB2_API_CHECK("восстановление базы данных - продолжение");
  }

  printf("\n  Восстановление выполнено.\n");

  return 0;
} /* DbBackupAndRestore */

int DbBackupAndRedirectedRestore(char dbAlias[],
                                 char restoredDbAlias[],
                                 char user[],
                                 char pswd[],
                                 char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  db2CfgParam cfgParameters[1];
  db2Cfg cfgStruct;
  unsigned short logretain;

  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];

  db2BackupStruct  backupStruct;
```

```
db2TablespaceStruct   tablespaceStruct;
db2MediaListStruct mediaListStruct;
db2Uint32  backupImageSize;
db2RestoreStruct restoreStruct;
db2TablespaceStruct   rtablespaceStruct;
db2MediaListStruct rmediaListStruct;

printf("\n**************************************\n");
printf("*** Перенаправленное восстановление ***\n");
printf("**************************************\n");
printf("\nИспользование API DB2:\n");
printf("  db2CfgSet -- Обновить конфигурацию\n");
printf("  db2Backup -- Сохранить базу данных\n");
printf("  sqlecrea -- Создать базу данных\n");
printf("  db2Restore -- Восстановить базу данных\n");
printf("  sqlbmtsq -- Запросить табличное пространство\n");
printf("  sqlbtcq -- Запросить контейнер табличного пространства\n");
printf("  sqlbstsc -- Задать контейнеры табличного пространства\n");
printf("  sqlefmem -- Освободить память\n");
printf("  sqledrpd -- Удалить базу данных\n");
printf("для сохранения и перенаправленного восстановления базы данных.\n");

printf("\n  Обновление конфигурации базы данных \'%s\':\n", dbAlias);
printf("    - Отключение параметра конфигурации LOGRETAIN\n");
printf("        например, указание LOGRETAIN = OFF/NO\n");

/* инициализация cfgParameters */
/* SQLF_DBTN_LOG_RETAIN - это маркер изменяемого параметра конфигурации
   'logretain' базы данных; он используется для изменения файла конфигурации базы
   данных */
cfgParameters[0].flags = 0;
cfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
cfgParameters[0].ptrvalue = (char *)&logretain;

/* отключение параметра 'logretain' конфигурации базы данных */
logretain = SQLF_LOGRETAIN_DISABLE;

/* инициализация cfgStruct */
cfgStruct.numItems = 1;
cfgStruct.paramArray = cfgParameters;
cfgStruct.flags = db2CfgDatabase | db2CfgDelayed;
cfgStruct.dbname = dbAlias;

/* получение конфигурации базы данных */
db2CfgSet(db2Version810, (void *)&cfgStruct, &sqlca);
DB2_API_CHECK("Сохранение протокола базы данных -- Отключение");

/*****************************/
/*   Сохранение базы данных    */
/*****************************/
printf("\n  Сохранение базы данных '%s'...\n", dbAlias);

tablespaceStruct.tablespaces = NULL;
tablespaceStruct.numTablespaces = 0;
```

```
mediaListStruct.locations = &serverWorkingPath;
mediaListStruct.numLocations = 1;
mediaListStruct.locationType = SQLU_LOCAL_MEDIA;

backupStruct.piDBAlias = dbAlias;
backupStruct.piTablespaceList = &tablespaceStruct;
backupStruct.piMediaList = &mediaListStruct;
backupStruct.piUsername = user;
backupStruct.piPassword = pswd;
backupStruct.piVendorOptions = NULL;
backupStruct.iVendorOptionsSize = 0;
backupStruct.iCallerAction = DB2BACKUP_BACKUP;
backupStruct.iBufferSize = 16; /*  16 x 4 Кб */
backupStruct.iNumBuffers = 1;
backupStruct.iParallelism =1;
backupStruct.iOptions = DB2BACKUP_OFFLINE | DB2BACKUP_DB;

/* API db2Backup создает резервную копию базы данных.
   Этот API автоматически устанавливает соединение
   с указанной базой данных.
   (Кроме того, этот API можно использовать для создания резервной копии
    табличного пространства). */
db2Backup (db2Version810, &backupStruct, &sqlca);

DB2_API_CHECK("База данных -- Резервное копирование");

while (sqlca.sqlcode != 0)
{
  /* продолжение операции резервного копирования */

  /* в зависимости от значения sqlca.sqlcode может потребоваться
     вмешательство пользователя, например, монтирование новой ленты */

  printf("\n  Продолжение сохранения...\n");

  backupStruct.iCallerAction = DB2BACKUP_CONTINUE;

  /* резервное копирование базы данных */
  db2Backup (db2Version810, &backupStruct, &sqlca);
}

printf("  Сохранение выполнено.\n");
printf("    - размер резервной копии  : %d MB\n",
       backupStruct.oBackupSize);
printf("    - каталог резервной копии : %s\n",
       mediaListStruct.locations[0]);

printf("    - дата резервной копии    : %s\n", backupStruct.oTimestamp);

/* Если кодовая страница клиента отличается от
   кодовой страницы сервера, чтобы восстановить удаленную базу данных,
   сначала нужно создать пустую базу данных.
   В этом случае удалите символы комментария с вызова DbCreate().
   Этот вызов создаст на сервере пустую базу данных с кодовой
   страницей сервера. */
```

```
/*
  rc = DbCreate(dbAlias, restoredDbAlias);
  if (rc != 0)
  {
  return rc;
  }
*/

  /*****************************/
  /* Восстановление базы данных */
  /*****************************/

  strcpy(restoreTimestamp, backupStruct.oTimestamp);

  rtablespaceStruct.tablespaces = NULL;
  rtablespaceStruct.numTablespaces = 0;

  rmediaListStruct.locations = &serverWorkingPath;
  rmediaListStruct.numLocations = 1;
  rmediaListStruct.locationType = SQLU_LOCAL_MEDIA;

  restoreStruct.piSourceDBAlias = dbAlias;
  restoreStruct.piTargetDBAlias = restoredDbAlias;
  restoreStruct.piTimestamp = restoreTimestamp;
  restoreStruct.piTargetDBPath = NULL;
  restoreStruct.piReportFile = NULL;
  restoreStruct.piTablespaceList = &rtablespaceStruct;
  restoreStruct.piMediaList = &rmediaListStruct;
  restoreStruct.piUsername = user;
  restoreStruct.piPassword = pswd;
  restoreStruct.piNewLogPath = NULL;
  restoreStruct.piVendorOptions = NULL;
  restoreStruct.iVendorOptionsSize = 0;
  restoreStruct.iParallelism = 1;
  restoreStruct.iBufferSize = 1024; /*  1024 x 4 Кб */;
  restoreStruct.iNumBuffers = 1;
  restoreStruct.iOptions = DB2RESTORE_OFFLINE | DB2RESTORE_DB |
  DB2RESTORE_NODATALINK | DB2RESTORE_NOROLLFWD;

  printf("\n  Восстановление базы данных ...\n");
  printf("    - алиас исходного образа : %s\n", dbAlias);
  printf("    - время исходного образа : %s\n", restoreTimestamp);
  printf("    - целевая база данных    : %s\n", restoredDbAlias);

  restoreStruct.iCallerAction = DB2RESTORE_RESTORE_STORDEF;

  /* API db2Restore применяется для восстановления базы данных,
     сохраненной с помощью API db2Backup. */
  db2Restore(db2Version810, &restoreStruct, &sqlca);

  EXPECTED_WARN_CHECK("database restore -- start");

  while (sqlca.sqlcode != 0)
  {
```

```
    /* продолжение операции восстановления */
    printf("\n  Продолжение восстановления...\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  Поиск и повторное определение недоступных контейнеров.\n");
      rc = InaccessableContainersRedefine(serverWorkingPath);
      if (rc != 0)
      {
        return rc;
      }
    }

    restoreStruct.iCallerAction = DB2RESTORE_CONTINUE;

    /* восстановление базы данных */
    db2Restore (db2Version810, &restoreStruct, &sqlca);

    DB2_API_CHECK("восстановление базы данных - продолжение");
  }

  printf("\n  Восстановление выполнено.\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("табличные пространства -- получение");
```

```
/* повторное определение 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("контейнеры табличных пространств -- получение");

  for (contNb = 0; contNb < numContainers; contNb++)
  {
    if (!pContainers[contNb].ok)
    {
      /* повторное определение недоступного контейнера */
      printf("\n    Повторное определение недоступного контейнера:\n");
      printf("      - имя табличного пространства: %s\n",
             ppTablespaces[tspNb]->name);
      printf("      - имя контейнера по умолчанию: %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("      - тип контейнера: путь\n");

          sprintf(pContainers[contNb].name, "%s%sSQLT%04d.%d",
                                            serverWorkingPath, pathSep,
                                            ppTablespaces[tspNb]->id,
                                            pContainers[contNb].id);
          printf("      - имя нового контейнера: %s\n",
                 pContainers[contNb].name);
          break;
        case SQLB_CONT_DISK:
        case SQLB_CONT_FILE:
        default:
          printf("    Неизвестный тип контейнера.\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("контейнеры табличных пространств -- повторное
     определение");

    /* API sqlefmem используется для высвобождения памяти, выделенной
       DB2 под использование с API sqlbtcq (Запрос контейнеров
       табличного пространства). */
    sqlefmem(&sqlca, pContainers);
    DB2_API_CHECK("память контейнеров табличных пространств --
     освобождение");
  }

  /* API sqlefmem используется для высвобождения памяти, выделенной DB2
     под использование с API sqlbmtsq (Запрос табличных пространств). */
  sqlefmem(&sqlca, ppTablespaces);
  DB2_API_CHECK("память табличных пространств -- освобождение");

  return 0;
} /* InaccessableContainersRedefine */

int DbBackupRestoreAndRollforward(char dbAlias[],
                                  char rolledForwardDbAlias[],
                                  char user[],
                                  char pswd[],
                                  char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  db2CfgParam cfgParameters[1];
  db2Cfg cfgStruct;
  unsigned short logretain;

  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];

  db2BackupStruct  backupStruct;
  db2TablespaceStruct   tablespaceStruct;
  db2MediaListStruct mediaListStruct;
  db2Uint32  backupImageSize;
  db2RestoreStruct restoreStruct;
  db2TablespaceStruct   rtablespaceStruct;
  db2MediaListStruct rmediaListStruct;

  db2RfwdInputStruct   rfwdInput;
  db2RfwdOutputStruct  rfwdOutput;
  db2RollforwardStruct  rfwdStruct;

  char rollforwardAppId[SQLU_APPLID_LEN + 1];
  sqlint32 numReplies;
  struct sqlurf_info nodeInfo;

  printf("\n*****************************************\n");
  printf("*** Восстановление с повтором транзакций ***\n");
```

```
printf("*****************************************\n");
printf("\nИспользование API DB2:\n");
printf("  db2CfgSet -- Задать конфигурацию\n");
printf("  db2Backup -- Сохранить базу данных\n");
printf("  sqlecrea -- Создать базу данных\n");
printf("  db2Restore -- Восстановить базу данных\n");
printf("  db2Rollforward -- Повтор транзакций базы данных\n");
printf("  sqledrpd -- Удалить базу данных\n");
printf("для резервного копирования, восстановления и повтора транзакций. \n");

printf("\n  Обновление конфигурации базы данных \'%s\':\n", dbAlias);
printf("    - Включение параметра конфигурации LOGRETAIN \n");
printf("        например, указание LOGRETAIN = RECOVERY/YES\n");

/* инициализация cfgParameters */
cfgParameters[0].flags = 0;
cfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
cfgParameters[0].ptrvalue = (char *)&logretain;

/* включение параметра конфигурации 'logretain' */
logretain = SQLF_LOGRETAIN_RECOVERY;

/* инициализация cfgStruct */
cfgStruct.numItems = 1;
cfgStruct.paramArray = cfgParameters;
cfgStruct.flags = db2CfgDatabase | db2CfgDelayed;
cfgStruct.dbname = dbAlias;

/* получение конфигурации базы данных */
db2CfgSet(db2Version810, (void *)&cfgStruct, &sqlca);
DB2_API_CHECK("Сохранение протокола базы данных -- Включение");

/* запуск операции резервного копирования */
printf("\n  Сохранение базы данных '%s'...\n", dbAlias);

tablespaceStruct.tablespaces = NULL;
tablespaceStruct.numTablespaces = 0;

mediaListStruct.locations = &serverWorkingPath;
mediaListStruct.numLocations = 1;
mediaListStruct.locationType = SQLU_LOCAL_MEDIA;

backupStruct.piDBAlias = dbAlias;
backupStruct.piTablespaceList = &tablespaceStruct;
backupStruct.piMediaList = &mediaListStruct;
backupStruct.piUsername = user;
backupStruct.piPassword = pswd;
backupStruct.piVendorOptions = NULL;
backupStruct.iVendorOptionsSize = 0;
backupStruct.iCallerAction = DB2BACKUP_BACKUP;
backupStruct.iBufferSize = 16; /*  16 x 4 Кб */
backupStruct.iNumBuffers = 1;
backupStruct.iParallelism =1;
backupStruct.iOptions = DB2BACKUP_OFFLINE | DB2BACKUP_DB;
```

```
  /* API db2Backup создает резервную копию базы данных.
     Этот API автоматически устанавливает соединение с указанной базой данных.
     (Кроме того, этот API можно использовать для создания резервной копии
      табличного пространства). */
  db2Backup (db2Version810, &backupStruct, &sqlca);

  DB2_API_CHECK("База данных -- Резервное копирование");

  while (sqlca.sqlcode != 0)
  {
    /* продолжение операции резервного копирования */
    printf("\n  Продолжение сохранения...\n");

    /* в зависимости от значения sqlca.sqlcode может потребоваться
       вмешательство пользователя, например, монтирование новой ленты */

    backupStruct.iCallerAction = DB2BACKUP_CONTINUE;

    /* резервное копирование базы данных */
    db2Backup (db2Version810, &backupStruct, &sqlca);

  DB2_API_CHECK("База данных -- Резервное копирование");
  }

  printf("  Сохранение выполнено.\n");
  printf("    - размер резервной копии  : %d MB\n",
         backupStruct.oBackupSize);
  printf("    - каталог резервной копии : %s\n",
         mediaListStruct.locations[0]);

  printf("    - дата резервной копии    : %s\n", backupStruct.oTimestamp);

  /* 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 */
  /*****************************/

  strcpy(restoreTimestamp, backupStruct.oTimestamp);

  rtablespaceStruct.tablespaces = NULL;
  rtablespaceStruct.numTablespaces = 0;
```

```
rmediaListStruct.locations = &serverWorkingPath;
rmediaListStruct.numLocations = 1;
rmediaListStruct.locationType = SQLU_LOCAL_MEDIA;

restoreStruct.piSourceDBAlias = dbAlias;
restoreStruct.piTargetDBAlias = rolledForwardDbAlias;
restoreStruct.piTimestamp = restoreTimestamp;
restoreStruct.piTargetDBPath = NULL;
restoreStruct.piReportFile = NULL;
restoreStruct.piTablespaceList = &rtablespaceStruct;
restoreStruct.piMediaList = &rmediaListStruct;
restoreStruct.piUsername = user;
restoreStruct.piPassword = pswd;
restoreStruct.piNewLogPath = NULL;
restoreStruct.piVendorOptions = NULL;
restoreStruct.iVendorOptionsSize = 0;
restoreStruct.iParallelism = 1;
restoreStruct.iBufferSize = 1024; /*  1024 x 4 Кб */;
restoreStruct.iNumBuffers = 1;
restoreStruct.iCallerAction = DB2RESTORE_RESTORE;
restoreStruct.iOptions = DB2RESTORE_OFFLINE | DB2RESTORE_DB |
DB2RESTORE_NODATALINK | DB2RESTORE_ROLLFWD;

printf("\n  Восстановление базы данных ...\n");
printf("    - алиас исходного образа : %s\n", dbAlias);
printf("    - время исходного образа : %s\n", restoreTimestamp);
printf("    - целевая база данных    : %s\n", rolledForwardDbAlias);

/* API db2Restore применяется для восстановления базы данных,
   сохраненной с помощью API db2Backup. */
db2Restore (db2Version810, &restoreStruct, &sqlca);

DB2_API_CHECK("восстановление базы данных -- запуск");

while (sqlca.sqlcode != 0)
{
  /* продолжение операции восстановления */
  printf("\n  Продолжение восстановления...\n");

  /* В зависимости от значения sqlca.sqlcode может потребоваться
     вмешательство пользователя, например, монтирование новой ленты */

  restoreStruct.iCallerAction = DB2RESTORE_CONTINUE;

  /* восстановление базы данных */
  db2Restore (db2Version810, &restoreStruct, &sqlca);

  DB2_API_CHECK("восстановление базы данных - продолжение");
}

printf("\n  Восстановление выполнено.\n");


/*****************************/
```

Программа примера со встроенным SQL (dbrecov.sqc)

```
  /*       Восстановление       */
  /*   с повтором транзакций    */
  /*****************************/

  printf("\n  Повтор транзакций базы данных '%s' ...\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.pDroppedTblID = NULL;
  rfwdInput.pExportDir = NULL;
  rfwdInput.NumNodeInfo = 1;
  rfwdInput.RollforwardFlags = 0;

  rfwdOutput.pApplicationId = rollforwardAppId;
  rfwdOutput.pNumReplies = &numReplies;
  rfwdOutput.pNodeInfo = &nodeInfo;

  rfwdStruct.roll_input = &rfwdInput;
  rfwdStruct.roll_output = &rfwdOutput;

  /* повтор транзакций базы данных */
  /* API db2Rollforward восстанавливает базу данных, повторяя
     транзакции, записанные в файлах журналов базы данных. */
  db2Rollforward(db2Version810, &rfwdStruct, &sqlca);

  DB2_API_CHECK("повтор транзакций -- запуск");

  printf("  Повтор транзакций выполнен.\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;
  db2CfgParam cfgParameters[1];
```

```
db2Cfg cfgStruct;
unsigned short logretain;

db2BackupStruct  backupStruct;
db2TablespaceStruct   tablespaceStruct;
db2MediaListStruct mediaListStruct;
db2Uint32  backupImageSize;
db2RestoreStruct restoreStruct;
db2TablespaceStruct   rtablespaceStruct;
db2MediaListStruct rmediaListStruct;

SQLU_LSN startLSN;
SQLU_LSN endLSN;
char *logBuffer;
sqluint32 logBufferSize;
db2ReadLogInfoStruct readLogInfo;
db2ReadLogStruct     readLogInput;
int i;

printf("\n**********************************\n");
printf("*** Асинхронное чтение журнала ***\n");
printf("**********************************\n");
printf("\nИспользование API DB2:\n");
printf("  db2CfgSet -- Задать конфигурацию\n");
printf("  db2Backup -- Сохранить базу данных\n");
printf("  db2ReadLog -- Асинхронное чтение журнала\n");
printf("и операторов SQL:\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("для чтения записей журнала для текущего соединения.\n");

printf("\n  Обновление конфигурации базы данных \'%s\':\n", dbAlias);
printf("    - Включение параметра конфигурации LOGRETAIN \n");
printf("        например, указание LOGRETAIN = RECOVERY/YES\n");

/* инициализация cfgParameters */
cfgParameters[0].flags = 0;
cfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
cfgParameters[0].ptrvalue = (char *)&logretain;

/* включение LOGRETAIN */
logretain = SQLF_LOGRETAIN_RECOVERY;

/* инициализация cfgStruct */
cfgStruct.numItems = 1;
cfgStruct.paramArray = cfgParameters;
cfgStruct.flags = db2CfgDatabase | db2CfgDelayed;
cfgStruct.dbname = dbAlias;

/* получение конфигурации базы данных */
```

```
db2CfgSet(db2Version810, (void *)&cfgStruct, &sqlca);
DB2_API_CHECK("Сохранение протокола базы данных -- Включение");

/* запуск операции резервного копирования */
printf("\n  Сохранение базы данных '%s'...\n", dbAlias);

tablespaceStruct.tablespaces = NULL;
tablespaceStruct.numTablespaces = 0;

mediaListStruct.locations = &serverWorkingPath;
mediaListStruct.numLocations = 1;
mediaListStruct.locationType = SQLU_LOCAL_MEDIA;

backupStruct.piDBAlias = dbAlias;
backupStruct.piTablespaceList = &tablespaceStruct;
backupStruct.piMediaList = &mediaListStruct;
backupStruct.piUsername = user;
backupStruct.piPassword = pswd;
backupStruct.piVendorOptions = NULL;
backupStruct.iVendorOptionsSize = 0;
backupStruct.iCallerAction = DB2BACKUP_BACKUP;
backupStruct.iBufferSize = 16; /*  16 x 4 Кб */
backupStruct.iNumBuffers = 1;
backupStruct.iParallelism =1;
backupStruct.iOptions = DB2BACKUP_OFFLINE | DB2BACKUP_DB;

/* API db2Backup создает резервную копию базы данных.
   Этот API автоматически устанавливает соединение с указанной базой данных.
   (Кроме того, этот API можно использовать для создания резервной копии
    табличного пространства). */
db2Backup (db2Version810, &backupStruct, &sqlca);

DB2_API_CHECK("База данных -- Резервное копирование");

while (sqlca.sqlcode != 0)
{
  /* продолжение операции резервного копирования */
  printf("\n  Продолжение сохранения...\n");

  /* В зависимости от значения sqlca.sqlcode может потребоваться
     вмешательство пользователя, например, монтирование новой ленты */

  backupStruct.iCallerAction = DB2BACKUP_CONTINUE;

  /* резервное копирование базы данных */
  db2Backup (db2Version810, &backupStruct, &sqlca);

DB2_API_CHECK("База данных -- Резервное копирование");
}

printf("  Сохранение выполнено.\n");
printf("    - размер резервной копии  : %d Мб\n", backupStruct.oBackupSize);
printf("    - каталог резервной копии : %s\n", mediaListStruct.locations[0]);

printf("    - дата резервной копии    : %s\n", backupStruct.oTimestamp);
```

```
/* соединение с базой данных */
rc = DbConn(dbAlias, user, pswd);
if (rc != 0)
{
  return rc;
}

/* вызов операторов SQL для заполнения журнала базы данных */
printf("\n  Вызов следующих операторов SQL:\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("Оператор SQL 1 -- вызов");

EXEC SQL COMMIT;
EMB_SQL_CHECK("Оператор SQL 2 -- вызов");

EXEC SQL INSERT INTO emp_resume
  VALUES('000777', 'ascii', 'Это новое возобновление.'),
        ('777777', 'ascii', 'Это еще одно новое возобновление.');
EMB_SQL_CHECK("Оператор SQL 3 -- вызов");

EXEC SQL COMMIT;
EMB_SQL_CHECK("Оператор SQL 4 -- вызов");

EXEC SQL DELETE FROM emp_resume WHERE empno = '000777';
EMB_SQL_CHECK("Оператор SQL 5 -- вызов");

EXEC SQL DELETE FROM emp_resume WHERE empno = '777777';
EMB_SQL_CHECK("Оператор SQL 6 -- вызов");

EXEC SQL COMMIT;
EMB_SQL_CHECK("Оператор SQL 7 -- вызов");

EXEC SQL DELETE FROM emp_resume WHERE empno = '000140';
EMB_SQL_CHECK("Оператор SQL 8 -- вызов");

EXEC SQL ROLLBACK;
EMB_SQL_CHECK("Оператор SQL 9 -- вызов");

EXEC SQL ALTER TABLE emp_resume DATA CAPTURE NONE;
EMB_SQL_CHECK("Оператор SQL 10 -- вызов");
```

```
EXEC SQL COMMIT;
EMB_SQL_CHECK("Оператор SQL 11 -- вызов");

printf("\n  Начало чтения журнала базы данных.\n");

logBuffer = NULL;
logBufferSize = 0;

/* API db2ReadLog (Асинхронное чтение журнала) применяется 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 информации о текущем состоянии
   журнала. */
readLogInput.iCallerAction  = DB2READLOG_QUERY;
readLogInput.piStartLSN     = NULL;
readLogInput.piEndLSN       = NULL;
readLogInput.poLogBuffer    = NULL;
readLogInput.iLogBufferSize = 0;
readLogInput.iFilterOption  = DB2READLOG_FILTER_ON;
readLogInput.poReadLogInfo  = &readLogInfo;

rc = db2ReadLog(db2Version810,
                &readLogInput,
                &sqlca);

DB2_API_CHECK("данные журнала базы данных -- получение");

logBufferSize = 64 * 1024;
logBuffer = (char *)malloc(logBufferSize);

memcpy(&startLSN, &(readLogInfo.initialLSN), sizeof(startLSN));
memcpy(&endLSN, &(readLogInfo.nextStartLSN), sizeof(endLSN));

/* Извлечение записи журналов из журналов базы данных и
   асинхронное чтение первой последовательности журнала. */
readLogInput.iCallerAction  = DB2READLOG_READ;
readLogInput.piStartLSN     = &startLSN;
readLogInput.piEndLSN       = &endLSN;
readLogInput.poLogBuffer    = logBuffer;
readLogInput.iLogBufferSize = logBufferSize;
readLogInput.iFilterOption  = DB2READLOG_FILTER_ON;
readLogInput.poReadLogInfo  = &readLogInfo;

rc = db2ReadLog(db2Version810,
                &readLogInput,
                &sqlca);
if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
{
  DB2_API_CHECK("журналы базы данных -- чтение");
}
else
```

```
  {
    if (readLogInfo.logRecsWritten == 0)
    {
      printf("\n  Журнал базы данных пуст.\n");
    }
  }

  /* вывод буфера журналов */
  rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);

  while (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
    /* чтение следующего журнала */

    memcpy(&startLSN, &(readLogInfo.nextStartLSN), sizeof(startLSN));

    /* Извлечение записи журналов из журналов базы данных и
     асинхронное чтение следующей последовательности журнала. */
     rc = db2ReadLog(db2Version810,
                     &readLogInput,
                     &sqlca);
    if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
    {
    DB2_API_CHECK("журналы базы данных -- чтение");
    }

    /* 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 DbReadLogRecordsNoConn(char dbAlias[])
{
  int rc = 0;
  struct sqlca sqlca;
  char      logPath[SQL_PATH_SZ + 1];
  db2CfgParam cfgParameters[1];
  db2Cfg    cfgStruct;
  char      nodeName[] = "NODE0000\0";
  db2Uint32 readLogMemSize = 4 * 4096;
  char      *readLogMemory = NULL;
  struct db2ReadLogNoConnInitStruct readLogInit;
  struct db2ReadLogNoConnInfoStruct readLogInfo;
```

```
struct db2ReadLogNoConnStruct readLogInput;
SQLU_LSN  startLSN;
SQLU_LSN  endLSN;
char      *logBuffer = NULL;
db2Uint32 logBufferSize = 0;
struct db2ReadLogNoConnTermStruct readLogTerm;

printf("\n************************************\n");
printf("*** Чтение журнала без подключения ***\n");
printf("************************************\n");
printf("\nИспользование API DB2:\n");
printf("  db2ReadLogNoConnInit -- Инициализировать чтение журнала
       без подключения\n");
printf("  db2ReadLogNoConn -- Читать журнал без подключения\n");
printf("  db2ReadLogNoConnTerm -- Завершить чтение журнала без
          подключения\n");
printf("для чтения записей журнала из каталога журналов базы
        данных.\n");

/* Определение каталога файлов журнала */
cfgParameters[0].flags = 0;
cfgParameters[0].token = SQLF_DBTN_LOGPATH;
cfgParameters[0].ptrvalue =
  (char *)malloc((SQL_PATH_SZ + 1) * sizeof(char));

/* инициализация cfgStruct */
cfgStruct.numItems = 1;
cfgStruct.paramArray = cfgParameters;
cfgStruct.flags = db2CfgDatabase;
cfgStruct.dbname = dbAlias;

db2CfgGet(db2Version810, (void *)&cfgStruct, &sqlca);
DB2_API_CHECK("каталог журнала -- получение");

strcpy(logPath, cfgParameters[0].ptrvalue);
free(cfgParameters[0].ptrvalue);

/* Сначала - выделение памяти для контрольных блоков API
   и буфера журнала */
readLogMemory = (char*)malloc(readLogMemSize);

/* Вызов API инициализации для задания контрольных блоков */
readLogInit.iFilterOption       = DB2READLOG_FILTER_ON;
readLogInit.piLogFilePath       = logPath;
readLogInit.piOverflowLogPath   = NULL;
readLogInit.iRetrieveLogs       = DB2READLOGNOCONN_RETRIEVE_OFF;
readLogInit.piDatabaseName      = dbAlias;
readLogInit.piNodeName          = nodeName;
readLogInit.iReadLogMemoryLimit = readLogMemSize;
readLogInit.poReadLogMemPtr     = &readLogMemory;

rc = db2ReadLogNoConnInit(db2Version810,
                          &readLogInit,
                          &sqlca);
if (sqlca.sqlcode != SQLU_RLOG_LSNS_REUSED)
```

```
{
  DB2_API_CHECK("журналы базы данных без подключения -- инициализация");
}

/* Запрос текущей информации журнала */
readLogInput.iCallerAction   = DB2READLOG_QUERY;
readLogInput.piStartLSN      = NULL;
readLogInput.piEndLSN        = NULL;
readLogInput.poLogBuffer     = NULL;
readLogInput.iLogBufferSize  = 0;
readLogInput.piReadLogMemPtr = readLogMemory;
readLogInput.poReadLogInfo   = &readLogInfo;

rc = db2ReadLogNoConn(db2Version810,
                      &readLogInput,
                      &sqlca);
if (sqlca.sqlcode != 0)
{
  DB2_API_CHECK("журналы базы данных без подключения -- запрос");
}

/* Чтение части записей журнала */
logBufferSize = 64 * 1024;
logBuffer = (char *)malloc(logBufferSize);

memcpy(&startLSN, &(readLogInfo.nextStartLSN), sizeof(startLSN));
endLSN.lsnWord[0] = 0xffff;
endLSN.lsnWord[1] = 0xffff;
endLSN.lsnWord[2] = 0xffff;

readLogInput.iCallerAction   = DB2READLOG_READ;
readLogInput.piStartLSN      = &startLSN;
readLogInput.piEndLSN        = &endLSN;
readLogInput.poLogBuffer     = logBuffer;
readLogInput.iLogBufferSize  = logBufferSize;
readLogInput.piReadLogMemPtr = readLogMemory;
readLogInput.poReadLogInfo   = &readLogInfo;

rc = db2ReadLogNoConn(db2Version810,
                      &readLogInput,
                      &sqlca);
if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
{
  DB2_API_CHECK("журналы базы данных без подключения -- чтение");
}
else
{
  if (readLogInfo.logRecsWritten == 0)
  {
    printf("\n  Журнал базы данных пуст.\n");
  }
}

/* Вывод считанных записей журнала */
rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
```

```
  while (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
    /* чтение следующего журнала */
    memcpy(&startLSN, &(readLogInfo.nextStartLSN), sizeof(startLSN));

    /* Извлечение записи журналов из журналов базы данных и
     асинхронное чтение следующей последовательности журнала. */
    rc = db2ReadLogNoConn(db2Version810,
                          &readLogInput,
                          &sqlca);
    if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
    {
    DB2_API_CHECK("журналы базы данных без подключения -- чтение");
    }

    /* вывод буфера журналов */
    rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
  }

  printf("\nЧтение до конца журналов.\n\n");
  free(logBuffer);

  readLogTerm.poReadLogMemPtr = &readLogMemory;

  rc = db2ReadLogNoConnTerm(db2Version810,
                            &readLogTerm,
                            &sqlca);
  if (sqlca.sqlcode != 0)
  {
    DB2_API_CHECK("журналы базы данных без подключения -- завершение");
  }

  return 0;
} /* DbReadLogRecordsNoConn */

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 (recordType == 0x0043)
  { /* компенсация */
    if (recordFlag & 0x0002)
    { /* допускающий распространение */
      logManagerLogRecordHeaderSize = 32;
    }
    else
    {
      logManagerLogRecordHeaderSize = 26;
    }
  }
  else
  { /* без компенсации */
    logManagerLogRecordHeaderSize = 20;
  }

  switch (recordType)
  {
    case 0x008A:
    case 0x0084:
    case 0x0041:
      recordDataBuffer = recordBuffer + logManagerLogRecordHeaderSize;
      recordDataSize = recordSize - logManagerLogRecordHeaderSize;
      rc = SimpleLogRecordDisplay(recordType,
                                  recordFlag,
                                  recordDataBuffer,
                                  recordDataSize);
      break;
    case 0x004E:
    case 0x0043:
      recordHeaderBuffer = recordBuffer + logManagerLogRecordHeaderSize;
      componentIdentifier = *(sqluint8 *)recordHeaderBuffer;
      switch (componentIdentifier)
      {
```

```
        case 1:
          recordHeaderSize = 6;
          break;
        default:
          printf("    Неизвестная сложная запись журнала: %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("    Неизвестная запись журнала: %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 (recordType)
  {
    case 138:
      printf("\n    Тип записи: Локальный ожидающий список\n");
      timeTransactionCommited = *(sqluint32 *)(recordDataBuffer);
      authIdLen = *(sqluint16 *)(recordDataBuffer + 4);
      memcpy(authId, (char *)(recordDataBuffer + 6), authIdLen);
      authId[authIdLen] = '\0';
      printf("      %s: %lu\n",
             "Время фиксации транзакции (в секундах с 01/01/70 UTC)",
             timeTransactionCommited);
      printf("      ID авторизации прикладной программы: %s\n", authId);
      break;
    case 132:
```

```
      printf("\n    Тип записи: Обычная фиксация\n");
      timeTransactionCommited = *(sqluint32 *)(recordDataBuffer);
      authIdLen = (sqluint16) (recordDataSize - 4);
      memcpy(authId, (char *)(recordDataBuffer + 4), authIdLen);
      authId[authIdLen] = '\0';
      printf("      %s: %lu\n",
             "Время фиксации транзакции (в секундах с 01/01/70 UTC)",
             timeTransactionCommited);
      printf("      ID авторизации прикладной программы: %s\n", authId);
      break;
    case 65:
      printf("\n    Тип записи: Обычное прерывание\n");
      authIdLen = (sqluint16) (recordDataSize);
      memcpy(authId, (char *)(recordDataBuffer), authIdLen);
      authId[authIdLen] = '\0';
      printf("      ID авторизации прикладной программы: %s\n", authId);
      break;
    default:
      printf("    Неизвестная простая запись журнала: %d %lu\n",
             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    Тип записи: Обычная\n");
    break;
  case 'C':
    printf("\n    Тип записи: Компенсация\n");
    break;
  default:
    printf("\n    Неизвестный тип сложной записи журнала: %c\n", recordType);
    break;
}

switch (componentIdentifier)
{
  case 1:
    printf("      ID компонента: запись журнала DMS\n");
    break;
  default:
    printf("      Неизвестный ID компонента: %d\n", componentIdentifier);
    break;
}

functionIdentifier = *(sqluint8 *)(recordHeaderBuffer + 1);
switch (functionIdentifier)
{
  case 106:
    printf("      ID функции: Удалить запись\n");
    RID = *(sqluint32 *)(recordDataBuffer + 2);
    subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    printf("        RID: %lu\n", RID);
    printf("        длина подзаписи: %u\n", subRecordLen);
    printf("        смещение подзаписи: %u\n", subRecordOffset);
    subRecordBuffer = recordDataBuffer + 12;
    rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
    break;
  case 111:
    printf("      ID функции: Отменить удаление записи\n");
    RID = *(sqluint32 *)(recordDataBuffer + 2);
    subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    printf("        RID: %lu\n", RID);
    printf("        длина подзаписи: %u\n", subRecordLen);
    printf("        смещение подзаписи: %u\n", subRecordOffset);
    subRecordBuffer = recordDataBuffer + 12;
    rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
    break;
  case 118:
    printf("      ID функции: Вставить запись\n");
    RID = *(sqluint32 *)(recordDataBuffer + 2);
    subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    printf("        RID: %lu\n", RID);
    printf("        длина подзаписи: %u\n", subRecordLen);
    printf("        смещение подзаписи: %u\n", subRecordOffset);
```

```
  subRecordBuffer = recordDataBuffer + 12;
  rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
  break;
case 120:
  printf("      ID функции: Обновить запись\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("        старый RID: %lu\n", oldRID);
  printf("        старая длина подзаписи: %u\n", oldSubRecordLen);
  printf("        старое смещение подзаписи: %u\n", oldSubRecordOffset);
  oldSubRecordBuffer = recordDataBuffer + 12;
  rc = LogSubRecordDisplay(oldSubRecordBuffer, oldSubRecordLen);
  printf("        новый RID: %lu\n", newRID);
  printf("        новая длина подзаписи: %u\n", newSubRecordLen);
  printf("        новое смещение подзаписи: %u\n", newSubRecordOffset);
  newSubRecordBuffer = recordDataBuffer +
                       12 +
                       oldSubRecordLen +
                       recordHeaderSize +
                       12;
  rc = LogSubRecordDisplay(newSubRecordBuffer, newSubRecordLen);
  break;
case 124:
  printf("      ID функции:  Изменить атрибут таблицы\n");
  alterBitMask = *(sqluint32 *)(recordDataBuffer + 2);
  alterBitValues = *(sqluint32 *)(recordDataBuffer + 6);
  if (alterBitMask & 0x00000001)
  {
    /* Изменение значения атрибута 'propagation': */
    printf("        Атрибут распространения изменен на: ");
    if (alterBitValues & 0x00000001)
    {
      printf("ВКЛ.\n");
    }
    else
    {
      printf("ВЫКЛ.\n");
    }
  }
```

```
  if (alterBitMask & 0x00000002)
  {
    /* Изменение значения атрибута 'pending': */
    printf("        Атрибут ожидания изменен на: ");
    if (alterBitValues & 0x00000002)
    {
      printf("ВКЛ.\n");
    }
    else
    {
      printf("ВЫКЛ.\n");
    }
  }
  if (alterBitMask & 0x00010000)
  {
    /* Изменение значения атрибута 'append mode': */
    printf("        Атрибут добавления изменен на: ");
    if (alterBitValues & 0x00010000)
    {
      printf("ВКЛ.\n");
    }
    else
    {
      printf("ВЫКЛ.\n");
    }
  }
  if (alterBitMask & 0x00200000)
  {
    /* Изменение значения атрибута 'LF Propagation': */
    printf("        Атрибут распространения LF изменен на: ");
    if (alterBitValues & 0x00200000)
    {
      printf("ВКЛ.\n");
    }
    else
    {
      printf("ВЫКЛ.\n");
    }
  }
  if (alterBitMask & 0x00400000)
  {
    /* Изменение значения атрибута 'LOB Propagation': */
    printf("        Атрибут распространения LOB изменен на: ");
    if (alterBitValues & 0x00400000)
    {
      printf("ВКЛ.\n");
    }
    else
    {
      printf("ВЫКЛ.\n");
    }
  }
  break;
default:
  printf("      неизвестный идентификатор функции: %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) &&
      (recordType != 16))
  {
    printf("        Неизвестный тип подзаписи: %x\n", recordType);
  }
  else if (recordType == 4)
  {
    printf("        тип подзаписи: Специальная управляющая\n");
  }
  else
  {
    /* recordType == 0 или recordType == 16
     * Тип записи 0 означает обычную запись
     * Тип записи 16 в данной программе
     * обрабатывается так же, как и 0
     */
    printf("        тип подзаписи: возможно обновление, ");
    updatableRecordType = *(sqluint8 *)(recordBuffer + 4);
    if (updatableRecordType != 1)
    {
      printf("Внутренняя управляющая\n");
    }
    else
    {
      printf("Форматированные пользовательские данные\n");
      userDataFixedLength = *(sqluint16 *)(recordBuffer + 6);
      printf("        пользовательские данные фиксированной длины: %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("        пользовательские данные:\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("*** Чтения файла хронологии восстановления базы данных ***\n");
printf("*********************************************************\n");
printf("\nИспользование API DB2:\n");
printf("  db2HistoryOpenScan -- Открыть просмотр файла хронологии
       восстановления\n");
printf("  db2HistoryGetEntry -- Получить следующую запись файла
       хронологии\n");
printf("  db2HistoryCloseScan -- Закрыть просмотр файла хронологии
       восстановления\n");
printf("для чтения файла хронологии восстановления базы данных.\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      */
/*        восстановления базы данных     */
/*****************************************/
printf("\n  Открытие файла хронологии восстановления базы данных '%s'.
       \n", dbAlias);

/* Открытие файла хронологии восстановлений для просмотра */
db2HistoryOpenScan(db2Version810, &dbHistoryOpenParam, &sqlca);
DB2_API_CHECK("файл хронологии восстановления базы данных --
 открытие");

numEntries = dbHistoryOpenParam.oNumRows;

/* dbHistoryOpenParam.oHandle возвращает ссылку для доступа к просмотру */
recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;

/*******************************************/
/*     Чтение записи из файла хронологии   */
/*******************************************/
for (entryNb = 0; entryNb < numEntries; entryNb = entryNb + 1)
{
  printf("\n  Чтение записи номер %u.\n", entryNb);
```

```
    /* получение следующей записи из файла хронологии восстановлений */
    db2HistoryGetEntry(db2Version810, &dbHistoryEntryGetParam, &sqlca);
    DB2_API_CHECK("запись файла хронологии восстановления базы данных --
           чтение")

    /* Вывод записей в файле хронологии восстановлений */
    printf("\n  Вывод записи номер %u.\n", entryNb);
    rc = HistoryEntryDisplay(histEntryData);
  }

  /*****************************************/
  /*        Закрытие файла хронологии      */
  /*        восстановления базы данных     */
  /*****************************************/
  printf("\n  закрытие файла хронологии восстановления базы данных '%s'.\n", dbAlias);

  /* API db2HistoryCloseScan заканчивает просмотр файла хронологии
     восстановлений и освобождает ресурсы DB2, требовавшиеся для просмотра. */
  db2HistoryCloseScan(db2Version810, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("файл хронологии восстановления базы данных -- закрытие");

  /* 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("    часть объекта: %s\n", buf);

  memcpy(buf, histEntryData.oEndTime.pioData,
              histEntryData.oEndTime.oLength);
  buf[histEntryData.oEndTime.oLength] = '\0';
  printf("    время завершения: %s\n", buf);

  memcpy(buf, histEntryData.oFirstLog.pioData,
              histEntryData.oFirstLog.oLength);
  buf[histEntryData.oFirstLog.oLength] = '\0';
  printf("    первый журнал: %s\n", buf);

  memcpy(buf, histEntryData.oLastLog.pioData,
              histEntryData.oLastLog.oLength);
  buf[histEntryData.oLastLog.oLength] = '\0';
  printf("    последний журнал: %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("    спецификатор таблицы: %s\n", buf);

  memcpy(buf, histEntryData.oTableName.pioData,
             histEntryData.oTableName.oLength);
  buf[histEntryData.oTableName.oLength] = '\0';
  printf("    имя таблицы: %s\n", buf);

  memcpy(buf, histEntryData.oLocation.pioData,
             histEntryData.oLocation.oLength);
  buf[histEntryData.oLocation.oLength] = '\0';
  printf("    расположение: %s\n", buf);

  memcpy(buf, histEntryData.oComment.pioData,
             histEntryData.oComment.oLength);
  buf[histEntryData.oComment.oLength] = '\0';
  printf("    комментарий: %s\n", buf);

  memcpy(buf, histEntryData.oCommandText.pioData,
             histEntryData.oCommandText.oLength);
  buf[histEntryData.oCommandText.oLength] = '\0';
  printf("    текст команды: %s\n", buf);
  printf("    ID записи файла хронологии: %u\n", histEntryData.oEID.ioHID);
  printf("    табличные пространства:\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("    тип операции: %c\n", histEntryData.oOperation);
  printf("    масштаб операции: %c\n", histEntryData.oObject);
  printf("    тип операции: %c\n", histEntryData.oOptype);
  printf("    состояние записи: %c\n", histEntryData.oStatus);
  printf("    тип устройства: %c\n", histEntryData.oDeviceType);
  printf("    SQLCA:\n");
  printf("      код sql: %ld\n", histEntryData.poEventSQLCA->sqlcode);
  memcpy(buf, histEntryData.poEventSQLCA->sqlstate, 5);
  buf[5] = '\0';
  printf("      состояние sql: %s\n", buf);
  memcpy(buf, histEntryData.poEventSQLCA->sqlerrmc,
             histEntryData.poEventSQLCA->sqlerrml);
  buf[histEntryData.poEventSQLCA->sqlerrml] = '\0';
  printf("      сообщение: %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("*** Обновление записи файла хронологии восстановления ***\n");
  printf("*********************************************************\n");
  printf("\nИспользование API DB2:\n");
  printf("  db2HistoryOpenScan -- Открыть просмотр файла хронологии
           восстановления\n");
  printf("  db2HistoryGetEntry -- Получить следующую запись файла
           хронологии\n");
  printf("  db2HistoryUpdate -- Обновить файл хронологии восстановления\n");
  printf("  db2HistoryCloseScan -- Закрыть просмотр файла хронологии
           восстановления\n");
  printf("для обновления записи файла хронологии восстановления база
         данных.\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      */
/*        восстановления базы данных     */
/*****************************************/
printf("\n  Открытие файла хронологии восстановления базы данных '%s'.\n",
       dbAlias);

/* API db2HistoryOpenScan запускает просмотр файла хронологии восстановлений */
db2HistoryOpenScan(db2Version810, &dbHistoryOpenParam, &sqlca);
DB2_API_CHECK("файл хронологии восстановления базы данных -- открытие");

/* dbHistoryOpenParam.oHandle возвращает ссылку для доступа к просмотру */
recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;

/*************************************************/
/*      Чтение первой записи из файла хронологии     */
/*************************************************/
printf("\n  чтение первой записи из файла хронологии восстановления.\n");

/* API db2HistoryGetEntry получает следующую запись из файла
   хронологии восстановлений. */
db2HistoryGetEntry(db2Version810, &dbHistoryEntryGetParam, &sqlca);
DB2_API_CHECK("первая запись файла хронологии восстановления -- чтение");
printf("\n  Вывод первой записи.\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, "Новое расположение");
strcpy(newComment, "Новый комментарий");
printf("\n  Обновление первой записи в файле хронологии восстановления:\n");
printf("    новое расположение = '%s'\n", newLocation);
printf("    новый комментарий = '%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(db2Version810, &dbHistoryUpdateParam, &sqlca);
DB2_API_CHECK("первая запись файла хронологии восстановления -- обновление");

rc = DbDisconn(dbAlias);
if (rc != 0)
{
  return rc;
}

/*****************************************/
/*        Закрытие файла хронологии      */
/*        восстановления базы данных     */
/*****************************************/
printf("\n  закрытие файла хронологии восстановления базы данных '%s'.\n", dbAlias);

/* API db2HistoryCloseScan заканчивает просмотр файла хронологии восстановлений и
   освобождает ресурсы DB2, требовавшиеся для просмотра. */
db2HistoryCloseScan(db2Version810, &recoveryHistoryFileHandle, &sqlca);
DB2_API_CHECK("файл хронологии восстановления базы данных -- закрытие");


/*****************************************/
/*     Повторное открытие файла хронологии    */
/*****************************************/
printf("\n  Открытие файла хронологии восстановления базы данных '%s'.\n", dbAlias);

/* запускает просмотр файла хронологии восстановлений */
db2HistoryOpenScan(db2Version810, &dbHistoryOpenParam, &sqlca);
DB2_API_CHECK("файл хронологии восстановления базы данных -- открытие");

recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;

dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;
printf("\n  Чтение первой записи из файла хронологии восстановления.\n");

/*************************************************************************/
/*      Чтение первой записи из файла хронологии после модификации      */
/*************************************************************************/
db2HistoryGetEntry(db2Version810, &dbHistoryEntryGetParam, &sqlca);
DB2_API_CHECK("первая запись файла хронологии восстановления -- чтение");

printf("\n  Вывод первой записи.\n");
rc = HistoryEntryDisplay(histEntryData);

/*****************************************/
/*         Закрытие файла хронологии     */
```

```
  /*         восстановления базы данных       */
  /*******************************************/
  printf("\n  Закрытие файла хронологии восстановления базы данных '%s'.\n",
         dbAlias);

  /* завершение просмотра файла хронологии восстановлений */
  db2HistoryCloseScan(db2Version810, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("файл хронологии восстановления базы данных -- закрытие");

  /* 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("*** Обрезание файла хронологии восстановления ***\n");
  printf("************************************************\n");
  printf("\nИспользование API DB2:\n");
  printf("  db2Prune -- Обрезать файл хронологии восстановления\n");
  printf("и операторов SQL:\n");
  printf("  CONNECT\n");
  printf("  CONNECT RESET\n");
  printf("для обрезания файла хронологии восстановления.\n");

  /* Соединение с базой данных: */
  rc = DbConn(dbAlias, user, pswd);
  if (rc != 0)
  {
    return rc;
  }

  /* Сокращение файла хронологии восстановлений: */
  printf("\n  Обрезать файл хронологии восстановления базы данных '%s'.\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(db2Version810, &histPruneParam, &sqlca);
  DB2_API_CHECK("файл хронологии восстановления базы данных -- обрезание");

  /* Отсоединение от базы данных: */
  rc = DbDisconn(dbAlias);
  if (rc != 0)
  {
    return rc;
  }

  return 0;
} /* DbRecoveryHistoryFilePrune */
```

Программа примера со встроенным SQL (dbrecov.sqc)

Приложение F. Tivoli Storage Manager

При вызове команды BACKUP DATABASE или RESTORE DATABASE можно указать, что для управления операциями резервного копирования и восстановления базы данных и табличного пространства должен применяться продукт Tivoli Storage Manager (TSM). Версия API клиента TSM должна быть не ниже 4.2.0. Исключение составляет 64-разрядная операционная система Solaris, в которой необходим API клиента TSM версии 4.2.1.

Настройка клиента Tivoli Storage Manager

Для того чтобы менеджер баз данных мог использовать опцию TSM, может потребоваться выполнить следующие действия по настройке среды TSM:

1. Необходимо установить и настроить клиент и сервер TSM. Кроме того, должен быть установлен API клиента TSM.
2. Необходимо задать переменные среды, применяемые API клиента TSM:

   DSMI_DIR Задает пользовательский каталог, в котором расположен файл доверенного агента API (`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`).

   Примечание: В среде многораздельной базы данных эти параметры должны быть заданы в каталоге `sqllib/userprofile`.
3. После изменения этих переменных среды в процессе работы менеджера баз данных следует выполнить следующие действия:
   - Остановите менеджер баз данных, введя команду **db2stop**.
   - Запустите менеджер баз данных, введя команду **db2start**.
4. В зависимости от конфигурации сервера, для обращения к серверу TSM клиенту Tivoli может потребоваться пароль. Если в среде TSM задан параметр `PASSWORDACCESS=generate`, то для клиента Tivoli необходимо задать пароль.

   Исполняемый файл `dsmapipw` устанавливается в каталог `sqllib/adsm` владельца экземпляра. Этот файл позволяет устанавливать и изменять пароль TSM.

Для вызова команды dsmapipw вы должны быть зарегистрированы как локальный администратор или пользователь “root”. При выполнении этой команды у пользователя 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. (Новый пароль нужно ввести дважды для контроля ошибок при вводе.)

Примечание: Пользователям, вызывающим команды BACKUP DATABASE и RESTORE DATABASE, не нужно знать этот пароль. Команду dsmapipw надо выполнить только для задания пароля для начального соединения; после этого пароль переустанавливается сервером TSM.

Особенности использования Tivoli Storage Manager

Чтобы использовать отдельные возможности TSM, вам может потребоваться указать полное имя объекта, использующего одну из этих возможностей. (Обратите внимание, что в операционных системах Windows вместо / используется \.) Полное имя:

- Объекта резервной копии всей базы данных:
  /<база_данных>/NODEnnnn/FULL_BACKUP.timestamp.seq_no
- Объекта инкрементной резервной копии базы данных:
  /<база_данных>/NODEnnnn/DB_INCR_BACKUP.timestamp.seq_no
- Объекта разностной инкрементной резервной копии базы данных_
  /<база_данных>/NODEnnnn/DB_DELTA_BACKUP.timestamp.seq_no
- Объекта резервной копии всего табличного пространства:
  /<база_данных>/NODEnnnn/TSP_BACKUP.timestamp.seq_no
- Объекта инкрементной резервной копии табличного пространства:
  /<база_данных>/NODEnnnn/TSP_INCR_BACKUP.timestamp.seq_no
- Объекта разностной инкрементной резервной копии табличного пространства:
  /<база_данных>/NODEnnnn/TSP_DELTA_BACKUP.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.

- Отдельные образы резервных копий помещаются в файловые пространства, которыми управляет TSM. Для работы с отдельными образами резервных копий можно применять только API TSM или утилиту **db2adutl**, которая использует эти API.
- 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.
- Для повышения производительности создайте несколько сеансов (если на сервере TSM есть необходимое аппаратное обеспечение).

Понятия, связанные с данным:

- “Управление файлами журнала” на стр. 50
- “Tivoli Space Manager Hierarchical Storage Manager (AIX)” в *Quick Beginnings for Data Links Manager*

Ссылки, связанные с данной темой:

- “db2adutl - Работа с архивными образами TSM” на стр. 227

Приложение G. Обработчик пользователя для восстановления баз данных

Для автоматического архивирования и восстановления файлов журнала можно написать *программу обработчика пользователя*. 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. 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. 347). 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 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 предусмотрено четыре примера программы обработчика пользователя:

  – `db2uext2.ctsm`

    В этом примере для архивирования и восстановления файлов журнала базы данных применяется 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`

    Этот пример применяет XBSA Draft 0.8, опубликованный группой X/Open. Его 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.ctsm`

    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.

Формат вызова

При вызове программы обработчика пользователя менеджер баз данных передает ей набор параметров (с типом данных CHAR). Формат вызова зависит от операционной системы:

```
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. Допустимы следующие значения: AIX®, Solaris, HP-UX, SCO, Linux 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.

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 Табл. 8 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 8. 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. [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 8. Коды возврата программы обработчика пользователя (продолжение). Применяется только для операций архивирования и восстановления.

Код возврата	Объяснение
[а] Для операции архивирования или восстановления код возврата 4 или 8 приводит к повтору через пять минут. Если программа обработчика пользователя продолжит возвращать 4 или 8 в ответ на запросы к одному и тому же файлу журнала, DB2 будет повторять попытки выполнить операцию, пока операция не завершится успешно. (Это относится к операциям с повтором транзакций и вызовам API **db2ReadLog**, который используется 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, за исключением 0 и 4, выдается оповещение. Оповещение содержит код возврата программы обработчика пользователя и копию входных параметров, переданных в программу обработчика пользователя.	

Приложение H. API резервного копирования и восстановления для продуктов других поставщиков

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 предоставляться производителем в общей библиотеке UNIX или DLL 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.
- Подробности о вызове резервного копирования и восстановления при помощи продуктов других поставщиков.
- Информация о структурах данных, используемых при вызовах API.

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

DB2 будет обращаться к этим функциям, которые должны предоставляться производителем в общей библиотеке UNIX или DLL 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 можно получить из структуры INIT-INPUT, используемой **sqluvint**.

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.

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 в вызове **sqluvint**, содержит информацию, необходимую для идентификации резервной копии. 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”.

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

Файл хронологии можно применять в качестве вспомогательного средства при восстановлении баз данных. Он связан с каждой базой данных и автоматически обновляется при каждой операции резервного копирования и восстановления. 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

Информация о формате файла приведена в справке по db2HistData.

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”.

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 этом случае, если необходимо восстановление, то следует найти резервную копию при помощи файла хронологии.

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*. Укажите имя библиотеки (в системах UNIX) или DLL (в системах Windows).
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.

API резервного копирования и восстановления для продуктов других поставщиков

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*

API (Application Programming Interface, интерфейс прикладного программирования)

Операции резервного копирования и восстановления поддерживаются двумя API: **db2Backup** для резервного копирования и **db2Restore** для восстановления.

Задаваемый параметр	Параметр API (для db2Backup и db2Restore)
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*

Ссылки, связанные с данной темой:

- “sqluvint - Инициализировать устройство и связаться с ним” на стр. 361
- “sqluvget - Чтение данных с устройства” на стр. 364
- “sqluvput - Запись данных на устройство” на стр. 367
- “sqluvend - Отсоединить устройство и освободить его ресурсы” на стр. 369
- “sqluvdel - Удалить сеанс после принятия” на стр. 371
- “DB2-INFO” на стр. 373
- “VENDOR-INFO” на стр. 376

sqluvint - Инициализировать устройство и связаться с ним

Вызов этой функции предоставляет информацию для инициализации и установки логической связи между DB2 и устройством поставщика.

Авторизация:

Одни из следующих:

- *sysadm*
- *dbadm*

Необходимое соединение:

База данных

Включаемый файл API:

sql.h

Синтаксис API 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.

Примечания 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. 9 на стр. 363).

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 выбран уровень подсказок WITHOUT PROMPTING и число съемных носителей больше запрошенного числа сеансов, то DB2 не сможет успешно выполнить операцию.

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 9. 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.

Таблица 9. Допустимые коды возврата для sqluvint и ответные действия DB2 (продолжение)

Литерал в файле заголовка	Описание	Вероятный следующий вызов	Прочие комментарии
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.
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.

Авторизация:

Одни из следующих:

- *sysadm*
- *dbadm*

Необходимое соединение:

База данных

Включаемый файл API:

sqluvend.h

Синтаксис API 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.

Примечания по использованию:

Эта функция используется утилитой восстановления.

Коды возврата:

Таблица 10. 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	
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	
SQLUV_ENDOFMEDIA	Конец носителя, считано > 0 байт (например, конец ленты).	sqluvend	DB2 обрабатывает данные, а затем - условие конца носителя.
SQLUV_IO_ERROR	Ошибка ввода/вывода.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
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.

Авторизация:

Одни из следующих:

- *sysadm*
- *dbadm*

Необходимое соединение:

База данных

Включаемый файл API:

sqluvend.h

Синтаксис API 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.

Примечания по использованию:

Эта функция используется утилитой резервного копирования.

Коды возврата:

Таблица 11. Допустимые коды возврата sqluvput и ответные действия DB2

Литерал в файле заголовка	Описание	Вероятный следующий вызов	Другая информация
SQLUV_OK	Успешное завершение операции	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	Запрошено 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	
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	
SQLUV_LINK_NOT_EXIST	Связь в данный момент не существует.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.

Таблица 11. Допустимые коды возврата sqluvput и ответные действия DB2 (продолжение)

Литерал в файле заголовка	Описание	Вероятный следующий вызов	Другая информация
SQLUV_IO_ERROR	Ошибка ввода-вывода.	sqluvend, action = SQLU_ABORT[a]	Сеанс будет прерван.
Следующий вызов:			
[a] Если следующий вызов - sqluvend, action = SQLU_ABORT, этот сеанс и все другие активные сеансы будут прерваны. Принятые сеансы удаляются последовательностью вызовов sqluvint, sqluvdel и sqluvend.			

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).

Авторизация:

Одни из следующих:

- *sysadm*
- *dbadm*

Необходимое соединение:

База данных

Включаемый файл API:

sql.h

Синтаксис API 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 API.

Примечания 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 код SQLUV_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. В операции резервного копирования принятые сеансы получают последовательность вызовов **sqluvint**, **sqluvdel** и **sqluvend**.

- 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 вызовами **sqluvend** с действием action = SQLUV_ABORT, а принятые сеансы получают последовательность вызовов **sqluvint**, **sqluvdel** и **sqluvend**.

  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 12. Допустимые коды возврата sqluvend и соответствующие им действия DB2

Литерал в файле заголовка	Описание	Вероятный следующий вызов	Другая информация
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.

Авторизация:

Одни из следующих:

- *sysadm*
- *dbadm*

Необходимое соединение:

База данных

Включаемый файл API:

sqluvend.h

Синтаксис API 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.

Примечания 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 13. Допустимые коды возврата для sqluvdel и ответные действия DB2

Литерал в файле заголовка	Описание	Вероятный следующий вызов	Другая информация
SQLUV_OK	успешное завершение операции	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 14. 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 14. 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	Зарезервирован для будущего использования.
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.
password	char	Пароль для узла, на котором была создана резервная копия.
owner	char	ID пользователя, запустившего резервное копирование.

Таблица 14. Поля структуры DB2-INFO (продолжение). Все поля представляют собой символьные строки с нулевым символом-ограничителем.

Имя поля	Тип данных	Описание
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.

DB2-INFO

Синтаксис:

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 15. 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.

Таблица 15. Поля структуры VENDOR-INFO (продолжение). Все поля представляют собой символьные строки с нулевым символом-ограничителем.

Имя поля	Тип данных	Описание
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:

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 16. 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.
num_sessions	unsigned short	Число сеансов, запрошенное пользователем при вызове операции резервного копирования или восстановления из резервной копии.

Синтаксис:

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 17. 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:

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 18. 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:

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 19. 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:

Структура на языке C

```
typedef struct Return_code
{
  sqlint32  return_code,
  char      description[30],
  void      *reserve,
} Return_code;
```

Приложение I. DB2 Universal Database - техническая информация

Обзор технической информации DB2 Universal Database

Техническую информацию DB2 Universal Database можно получить в следующих форматах:

- Книги (в формате PDF и как печатные копии)
- Дерево тем (в формате HTML)
- Справка по инструментам DB2 (в формате HTML)
- Программы примеров (в формате HTML)
- Справка командной строки
- Обучающие программы

В этом разделе приводится обзор поставляемой технической информации с возможными способами ее получения.

Пакеты FixPak для документации DB2

IBM может периодически выпускать пакеты FixPak к документации. Пакеты FixPak к документации позволяют обновлять информацию, установленную с *компакт-диска документации HTML для DB2*, когда становится доступной новая информация.

Примечание: После установки пакетов FixPaks к документации ваша документация в формате HTML будет содержать более свежую информацию, чем печатные руководства по DB2 и книги в формате PDF.

Категории технической информации DB2

Техническая информация DB2 подразделена на следующие категории:

- Базовая информация о DB2
- Информация об управлении
- Информация о разработке программ
- Информация о возможностях для бизнеса
- Информация о DB2 Connect
- Информация Начинаем работу
- Информация по обучающим программам
- Информация о дополнительных компонентах
- Замечания по выпуску

В следующих таблицах содержится информация, необходимая для заказа печатных копий, печати или просмотра файлов PDF, а также поиска каталогов HTML для каждой книги библиотеки DB2. Полное описание каждой из книг библиотеки DB2 можно посмотреть в центре публикаций IBM на странице www.ibm.com/shop/publications/order

Для каждой категории информации на компакт-диске документации в формате HTML предусмотрен свой каталог установки:

путь_компакт_диска_html/doc/htmlcd/*%L*/*категория*

где:

- *путь_компакт_диска_html* - каталог, где установлен компакт-диск HTML.
- *%L* - идентификатор языка. Например, ru_RU.
- *категория* - идентификатор категории. Например, core - идентификатор базовой информации DB2.

В следующих таблицах в столбце имен файла PDF символ на шестой позиции в имени файла обозначает национальную версию книги. Например, имя файла db2d1e80 говорит о том, что это английская версия книги *Administration Guide: Planning* (Руководство администратора: Планирование), а имя файла db2d1r80 соответствует русской версии этой же книги. Для обозначений языков используются на шестой позиции имени файла используются следующие буквы:

Язык	**Обозначение**
Арабский	w
Бразильский португальский	b
Болгарский	u
Хорватский	9
Чешский	x
Датский	d
Голландский	q
Английский	e
Финский	y
Французский	f
Немецкий	g
Греческий	a
Венгерский	h
Итальянский	i
Японский	j
Корейский	k
Норвежский	n
Польский	p
Португальский	v
Румынский	8
Русский	r

Упрощенный китайский	c
Словацкий	7
Словенский	l
Испанский	z
Шведский	s
Традиционный китайский	t
Turkish	m

Если **номера формы нет**, это значит, что книга доступна только в электронном виде, и для нее не существует печатной версии.

Базовая информация о DB2

Информация в этой категории охватывает темы DB2, существенные для всех пользователей DB2. Информация в этой категории будет полезна и программисту, и администратору баз данных, и тому, кто работает с DB2 Connect, Менеджером хранилищ DB2 или с другими продуктами DB2.

Каталог установки для данной категории - `doc/htmlcd/%L/core`.

Таблица 20. Базовая информация о DB2

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Command Reference	SC09-4828	db2n0x80
IBM DB2 Universal Database Glossary (Глоссарий IBM DB2 Universal Database)	Номера формы нет	db2t0x80
IBM DB2 Universal Database Master Index	SC09-4839	db2w0x80
IBM DB2 Universal Database Message Reference, Volume 1 (Справочник по сообщениям IBM DB2 Universal Database, том 1)	GC09-4840 (GH43-0197)	db2m1x80
IBM DB2 Universal Database Message Reference, Volume 2 (Справочник по сообщениям IBM DB2 Universal Database, том 2)	GC09-4841 (GH43-0196)	db2m2x80
IBM DB2 Universal Database What's New (IBM DB2 Universal Database. Что нового)	SC09-4848 (GH43-0198-00)	db2q0x80

Информация об управлении

Информация в этой категории охватывает темы, необходимые для эффективной разработки, реализации и обслуживания баз данных, хранилищ данных и систем объединения DB2.

Каталог установки для данной категории - `doc/htmlcd/%L/admin`.

Таблица 21. Информация об управлении

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Administration Guide: Planning (Руководство администратора IBM DB2 Universal Database: Планирование)	SC09-4822 (GH43-0200)	db2d1x80
IBM DB2 Universal Database Administration Guide: Implementation (Руководство администратора IBM DB2 Universal Database: Реализация)	SC09-4820 (GH43-0202)	db2d2x80
IBM DB2 Universal Database Administration Guide: Performance (Руководство администратора IBM DB2 Universal Database: Производительность)	SC09-4821 (GH43-0201)	db2d3x80
IBM DB2 Universal Database Administrative API Reference	SC09-4824	db2b0x80
IBM DB2 Universal Database Data Movement Utilities Guide and Reference	SC09-4830	db2dmx80
IBM DB2 Universal Database Data Recovery and High Availability Guide and Reference (Справочное руководство по восстановлению данных и высокой доступности IBM DB2 Universal Database)	SC09-4831 (SH43-0210)	db2hax80
IBM DB2 Universal Database Data Warehouse Center Administration Guide	SC27-1123	db2ddx80
IBM DB2 Universal Database Federated Systems Guide	GC27-1224	db2fpx80

Таблица 21. Информация об управлении (продолжение)

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Guide to GUI Tools for Administration and Development (Руководство IBM DB2 Universal Database по инструментам GUI для управления и разработки)	SC09-4851 (GH43-0203)	db2atx80
IBM DB2 Universal Database Replication Guide and Reference	SC27-1121	db2e0x80
IBM DB2 Installing and Administering a Satellite Environment	GC09-4823	db2dsx80
IBM DB2 Universal Database SQL Reference, Volume 1	SC09-4844	db2s1x80
IBM DB2 Universal Database SQL Reference, Volume 2	SC09-4845	db2s2x80
IBM DB2 Universal Database System Monitor Guide and Reference	SC09-4847	db2f0x80

Информация о разработке программ

Информация в этой категории представляет особый интерес для разработчиков и программистов, работающих с DB2. Здесь вы найдете информацию о поддерживаемых языках и компиляторах, а также документацию, требуемую для обращения к DB2 при помощи разнообразных поддерживаемых интерфейсов программирования, таких как встроенный SQL, ODBC, JDBC, SQLj и CLI. При просмотре этой информации в электронном виде доступен также набор программ примеров DB2 в формате HTML.

Каталог установки для данной категории - `doc/htmlcd/%L/ad`.

Таблица 22. Информация о разработке программ

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Application Development Guide: Building and Running Applications	SC09-4825	db2axx80

Таблица 22. Информация о разработке программ (продолжение)

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Application Development Guide: Programming Client Applications	SC09-4826	db2a1x80
IBM DB2 Universal Database Application Development Guide: Programming Server Applications	SC09-4827	db2a2x80
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1	SC09-4849	db2l1x80
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2	SC09-4850	db2l2x80
IBM DB2 Universal Database Data Warehouse Center Application Integration Guide	SC27-1124	db2adx80
IBM DB2 XML Extender Administration and Programming	SC27-1234	db2sxx80

Информация о возможностях для бизнеса

Информация в этой категории описывает, как использовать компоненты, расширяющие возможности центров данных и аналитической обработки в DB2 Universal Database.

Каталог установки для данной категории - `doc/htmlcd/%L/wareh`.

Таблица 23. Информация о возможностях для бизнеса

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Warehouse Manager Information Catalog Center Administration Guide	SC27-1125	db2dix80
IBM DB2 Warehouse Manager Installation Guide	GC27-1122	db2idx80

Информация о DB2 Connect

Информация в этой категории описывает, как работать с данными хоста или iSeries при помощи DB2 Connect Enterprise Edition или DB2 Connect Personal Edition.

Каталог установки для данной категории - `doc/htmlcd/%L/conn`.

Таблица 24. Информация о DB2 Connect

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
Смысловые коды APPC, CPI-C и SNA	Номера формы нет	db2apx80
IBM Connectivity Supplement (Дополнение по возможностям соединений IBM)	Номера формы нет	db2h1x80
IBM DB2 Connect Quick Beginnings for DB2 Connect Enterprise Edition	GC09-4833	db2c6x80
IBM DB2 Connect Quick Beginnings for DB2 Connect Personal Edition (Быстрый старт DB2 Connect для DB2 Connect Personal Edition)	GC09-4834 (GH43-0223)	db2c1x80
IBM DB2 Connect User's Guide (Руководство пользователя IBM DB2 Connect)	SC09-4835 (GH43-0199)	db2c0x80

Информация Начинаем работу

Информация в этой категории полезна при установке и конфигурировании серверов, клиентов и других продуктов DB2.

Каталог установки для этой категории - `doc/htmlcd/%L/start`.

Таблица 25. Информация Начинаем работу

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Quick Beginnings for DB2 Clients (Быстрый старт IBM DB2 Universal Database для клиентов DB2)	GC09-4832 (GH43-0222)	db2itx80

Таблица 25. Информация Начинаем работу (продолжение)

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Quick Beginnings for DB2 Servers (Быстрый старт IBM DB2 Universal Database для серверов DB2)	GC09-4836 (GH43-0221)	db2isx80
IBM DB2 Universal Database Quick Beginnings for DB2 Personal Edition	GC09-4838	db2i1x80
IBM DB2 Universal Database Installation and Configuration Supplement (Дополнение по установке и настройке IBM DB2 Universal Database)	GC09-4837 (GH43-0220)	db2iyx80
IBM DB2 Universal Database Quick Beginnings for DB2 Data Links Manager	GC09-4829	db2z6x80

Информация по обучающим программам

Обучающие программы знакомят вас с функциями DB2 и обучают выполнению различных задач.

Каталог установки для этой категории - `doc/htmlcd/%L/tutr`.

Таблица 26. Информация по обучающим программам

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
Business Intelligence Tutorial: Introduction to the Data Warehouse	Номера формы нет	db2tux80
Business Intelligence Tutorial: Extended Lessons in Data Warehousing	Номера формы нет	db2tax80
Development Center Tutorial for Video Online using Microsoft Visual Basic	Номера формы нет	db2tdx80
Information Catalog Center Tutorial	Номера формы нет	db2aix80
Video Central for e-business Tutorial	Номера формы нет	db2twx80
Visual Explain Tutorial	Номера формы нет	db2tvx80

Информация о дополнительных компонентах

Информация в этой категории описывает, как работать с дополнительными компонентами DB2.

Каталог установки для этой категории - `doc/htmlcd/%L/opt`.

Таблица 27. Информация о дополнительных компонентах

Название	Номер формы	Имя файла PDF
IBM DB2 Life Sciences Data Connect Planning, Installation, and Configuration Guide	GC27-1235	db2lsx80
IBM DB2 Spatial Extender User's Guide and Reference	SC27-1226	db2sbx80
IBM DB2 Universal Database Data Links Manager Administration Guide and Reference	SC27-1221	db2z0x80
IBM DB2 Universal Database Net Search Extender Administration and Programming Guide **Примечание:** Этот документ в виде HTML не устанавливается с компакт-диска документации HTML.	SH12-6740	Нет

Замечания по выпуску

В замечаниях по выпуску предоставляется дополнительная информация, относящаяся конкретно к вашему выпуску продукта и уровню FixPak. В них также содержится сводная информация по обновлениям к документации, включаемым в каждый выпуск и пакет FixPak.

Таблица 28. Замечания по выпуску

Название	Номер формы	Имя файла PDF
Замечания по выпуску DB2	Смотрите примечание.	Смотрите примечание.
Замечания по установке DB2	Доступны только на компакт-диске продукта.	Доступны только на компакт-диске продукта.

Примечание: HTML-версию Замечаний по выпуску можно вызвать через Информационный центр или с компакт-диска продукта. Чтобы посмотреть файл ASCII на платформах UNIX, откройте файл `Release.Notes`. Он расположен в каталоге `DB2DIR/Readme/%L`, где `%L` - национальная версия, а `DB2DIR`:

- `/usr/opt/db2_08_01` - в AIX
- `/opt/IBM/db2/V8.1` - в других операционных системах UNIX

Задачи, связанные с данной темой:

- “Печать книг DB2 из файлов PDF” на стр. 392
- “Заказ печатных копий книг DB2” на стр. 393
- “Обращение к электронной справке” на стр. 394
- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 398
- “Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML” на стр. 399

Печать книг DB2 из файлов PDF

Можно напечатать книги по DB2 из файлов PDF с компакт-диска *Документация по DB2 в формате PDF*. При помощи Adobe Acrobat Reader можно напечатать книгу целиком или же определенный диапазон страниц.

Предварительные требования:

У вас должен быть Adobe Acrobat Reader. Его можно получить на сайт Adobe по адресу www.adobe.com

Процедура:

Чтобы напечатать книгу DB2 из файла PDF:

1. Вставьте компакт-диск *Документация по DB2 в формате PDF* в дисковод. В операционных системах UNIX смонтируйте компакт-диск *Документация по DB2 в формате PDF*. Подробности о том, как смонтировать компакт-диск в операционных системах UNIX, смотрите в книге *Quick Beginnings* (Быстрый старт).
2. Запустите Adobe Acrobat Reader.
3. Откройте файл PDF из одного из следующих мест:
   - В операционных системах Windows:

     Из каталога *x*`:\doc\`*язык*, где *x* - буква дисковода компакт-дисков, а *язык* - двухсимвольный код территории, соответствующий вашему языку (например, RU для русского).
   - В операционных системах UNIX:

     Из каталога `/`*cdrom*`/doc/`*%L* на компакт-диске, где */cdrom* - точка установки компакт-диска, а *%L* - имя требуемой национальной версии.

Задачи, связанные с данной темой:

- “Заказ печатных копий книг DB2” на стр. 393
- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 398
- “Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML” на стр. 399

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 383

Заказ печатных копий книг DB2

Процедура:

Чтобы заказать печатные книги:

- Обратитесь к авторизованному дилеру или торговому представителю IBM. Локального представителя IBM можно найти во каталоге контактных адресов IBM (IBM Worldwide Directory of Contacts) по адресу www.ibm.com/planetwide
- Позвоните по телефону `1-800-879-2755` в США или `1-800-IBM-4YOU` в Канаде.
- С Web-страницы Центра публикаций IBM (IBM Publications Center): www.ibm.com/shop/publications/order

Печатные копии руководств DB2 можно также получить, заказав у поставщика IBM пакеты документации (Doc Packs) для вашего продукта DB2. Пакеты документации - это избранные руководства из библиотеки DB2, облегчающие освоение приобретенного вами продукта DB2. Те же руководства доступны в формате PDF на компакт-диске *Документация по DB2 в формате PDF*, их содержание совпадает с содержанием компакт-диска *Документация по DB2 в формате HTML*.

Задачи, связанные с данной темой:

- “Печать книг DB2 из файлов PDF” на стр. 392
- “Поиск тем при обращении к Информационному центру DB2 из браузера” на стр. 395
- “Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML” на стр. 399

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 383

Обращение к электронной справке

Электронная справка, поставляемая со всеми компонентами DB2, доступна в трех вариантах:

- Справка по окну и записной книжке
- Справка командной строки
- Справка по операторам SQL

В справке по окну и записной книжке объясняются задачи, выполняемые в окне или записной книжке, и описываются органы управления. Эта справка бывает двух типов:

- Справка, вызываемая кнопкой **Справка**
- Всплывающие подсказки

Кнопка **Справка** позволяет обращаться к обзорной информации и информации о предварительных условиях. Всплывающие подсказки описывают органы управления в окне или записной книжке. Справка окна и записной книжки доступна из центров и компонентов DB2, поддерживающих пользовательский интерфейс.

Справка командной строки состоит из справки по командам и справки по сообщениям. Справка по командам объясняет синтаксис команд процессора командной строки. Справка по сообщениям описывает причины появления сообщений об ошибках и необходимые действия в ответ на ошибки.

Справка по операторам SQL состоит из справки SQL и справки SQLSTATE. Система DB2 возвращает SQLSTATE - значения, описывающие ошибки, которые могут возникнуть при выполнении оператора SQL. Справка по SQLSTATE объясняет синтаксис операторов SQL (состояния SQL и коды классов).

Примечание: Справка по SQL недоступна для операционных систем UNIX.

Процедура:

Чтобы обратиться к электронной справке:

- Для справки по окну и записной книжке нажмите кнопку **Справка** или щелкните по интересующему вас органу управления и затем нажмите клавишу **F1**. Если на странице **Общие** записной книжки **Параметры инструментов** включен переключатель **Автоматически выводить всплывающие подсказки**, всплывающие подсказки по органам управления будут появляться также при наведении на них указателя мыши.
- Для справки командной строки откройте процессор командной строки и введите:

– Для справки по командам:

```
? 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.

- Для справки по сообщениям:

```
? XXXnnnnn
```

где *XXXnnnnn* - идентификатор существующего сообщения.

Например, `? SQL30081` выводит справку по сообщению SQL30081.

- Для справки по оператору SQL введите в командной строке DB2:

```
? 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.

Задачи, связанные с данной темой:

- “Поиск тем при обращении к Информационному центру DB2 из браузера” на стр. 395
- “Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML” на стр. 399

Поиск тем при обращении к Информационному центру DB2 из браузера

Обращение к Информационному центру DB2 из браузера дает доступ к информации, необходимой для полного использования всех возможностей DB2 Universal Database и DB2 Connect. Информационный центр DB2 содержит также сведения по основным возможностям и компонентам DB2, включая репликацию, хранилище данных, метаданные и модули расширения DB2.

При обращении из браузера Информационный центр DB2 будет состоять из следующих основных элементов:

Дерево навигации

Дерево навигации расположено в левом фрейме окна браузера. Его можно разворачивать и сворачивать для показа и скрытия тем, глоссария и главного указателя Информационного центра DB2.

Панель инструментов навигации

Панель инструментов навигации расположена в правом фрейме окна браузера. Она содержит кнопки, позволяющие вести поиск в Информационном центре DB2, скрывать дерево навигации и искать текущую тему в этом дереве.

Фрейм содержимого

Фрейм содержимого - это правый нижний фрейм окна браузера. Если щелкнуть по ссылке в дереве навигации, по результату поиска или же перейти по ссылке из другой темы или главного указателя, во фрейме содержимого выводятся темы Информационного центра DB2.

Предварительные требования:

Для доступа к Информационному центру DB2 из браузера необходим один из следующих браузеров:

- Microsoft Explorer Версии 5 или новее
- Netscape Navigator Версии 6.1 или новее

Ограничения:

Информационный центр DB2 содержит только те наборы тем, которые вы установили с компакт-диска *Документация по DB2 в формате HTML*. Если при попытке перехода к теме по ссылке ваш браузер возвратил ошибку `Файл не найден`, необходимо установить дополнительные наборы тем с компакт-диска *Документация по DB2 в формате HTML*.

Процедура:

Чтобы найти тему по ключевым словам:

1. Нажмите на панели инструментов навигации кнопку **Поиск**.
2. В верхнем текстовом поле ввода окна Поиск введите один или несколько терминов, отражающих интересующую вас область, и нажмите кнопку **Поиск**. В поле **Результаты** будет выведен список тем, ранжированных в порядке точности соответствия условиям поиска. Число рядом с каждым результатом поиска отражает точность соответствия (чем больше число, тем лучше соответствие).

   Ввод дополнительных слов для поиска повышает точность запроса, сокращая количество возвращаемых тем.
3. В поле **Результаты** щелкните по заголовку интересующей вас темы. Информация по этой теме будет выведена во фрейме содержимого.

Чтобы найти тему в дереве навигации:

1. Щелкните по значку с книгой у интересующего вас тематического раздела в дереве навигации. Под значком появится список подкатегорий этого раздела.

2. Щелкая по значкам с книгой, раскрывайте далее эти подкатегории, пока не дойдете до категории с нужными сведениями. Заголовки категорий, содержащих ссылки на темы справки, при наведении на них указателя мыши принимают вид подчеркнутой ссылки. Отдельные темы в дереве навигации обозначаются значком страницы.
3. Щелкните по ссылке на нужную тему. Информация по этой теме будет выведена во фрейме содержимого.

Чтобы найти тему или термин в главном указателе:

1. Щелкните по категории “Указатель” в дереве навигации. Категория примет вид дерева навигации со списком расположенных в алфавитном порядке ссылок.
2. Щелкните в этом дереве навигации по ссылке на первый символ термина, относящегося к интересующей вас теме. Во фрейме содержимого появится список терминов, начинающихся с этого символа. Термины, которым соответствует несколько вхождений указателя, будут отмечены значком книги.
3. Щелкните по значку у интересующего вас термина. Под этим термином появится список подчиненных терминов и тем справки. Темы обозначаются значком страницы с подчеркнутым заголовком.
4. Щелкните по заголовку нужной темы. Информация по теме будет выведена во фрейме содержимого.

Понятия, связанные с данным:

- “Доступность” на стр. 405
- “Информационный центр DB2 при обращении из браузера” на стр. 407

Задачи, связанные с данной темой:

- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 398
- “Обновление документации HTML, установленной на вашем компьютере” на стр. 400
- “Устранение ошибок при поиске в документации DB2 с помощью Netscape 4.x” на стр. 402
- “Поиск в документации DB2” на стр. 403

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 383

Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления

Информационный центр DB2 обеспечивает быстрый доступ к информации о программном продукте DB2. Он доступен во всех операционных системах, где доступны инструменты управления DB2.

При обращении из инструментов управления в Информационном центре DB2 выводятся шесть типов информации.

Задачи Основные задания, которые вы можете выполнить в DB2.

Основные понятия
Основные понятия DB2.

Справочник
Справочная информацию по таким элементам DB2, как ключевые слова, команды и API.

Устранение неисправностей
Сообщения об ошибках и информация, которая поможет вам при возникновении проблем с DB2.

Примеры
Ссылки на тексты HTML примеров программ, поставляемых с DB2.

Обучающие программы
Пошаговая помощь для освоения возможностей DB2.

Предварительные требования:

Некоторые ссылки в Информационном центре DB2 указывают на сайты в Интернете. Чтобы посмотреть содержимое таких ссылок, надо соединиться с Интернетом.

Процедура:

Чтобы найти информацию о продукте при обращении к Информационному центру DB2 из инструментов:

1. Запустите Информационный центр DB2 одним из следующих способов:
   - На панели графических инструментов управления щелкните по значку **Информационный центр**. Этот пункт можно также выбрать в меню **Справка**.
   - Введите в командной строке **db2ic**.
2. Щелкните по вкладке типа информации, связанного с информацией, которую вы ищете.
3. Разверните дерево и щелкните по интересующей вас теме. Информационный центр запускает браузер для вывода этой информации.

4. Чтобы найти информацию, не просматривая списки, щелкните по значку **Поиск** справа от списка.

   Когда Информационный центр запустит браузер для вывода информации, вы можете выполнять поиск по всему тексту, щелкнув по значку **Поиск** на навигационной панели.

Понятия, связанные с данным:

- “Доступность” на стр. 405
- “Информационный центр DB2 при обращении из браузера” на стр. 407

Задачи, связанные с данной темой:

- “Поиск тем при обращении к Информационному центру DB2 из браузера” на стр. 395
- “Поиск в документации DB2” на стр. 403

Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML

Все темы в формате HTML, которые можно установить с компакт-диска *Документация по DB2 в формате HTML*, можно также читать непосредственно с этого компакт-диска. Поэтому просмотр документации возможен и без ее установки.

Ограничения:

Поскольку справка по инструментам устанавливается с компакт-диска продукта DB2, а не с компакт-диска *Документация по DB2 в формате HTML*, для просмотра справки необходимо установить этот продукт DB2.

Процедура:

1. Вставьте в дисковод компакт-диск *Документация по DB2 в формате HTML*. В операционных системах UNIX смонтируйте компакт-диск *Документация по DB2 в формате HTML*. Подробности о том, как смонтировать компакт-диск в операционных системах UNIX, смотрите в книге *Quick Beginnings* (Быстрый старт).
2. Запустите ваш браузер и откройте нужный файл:
   - Для операционных систем Windows:

     ```
     e:\program files\IBM\SQLLIB\doc\htmlcd\%L\index.htm
     ```

     где *e* - дисковод компакт-дисков, а %L - необходимая вам национальная версия документации, например, **ru_RU** для русского языка.
   - Для операционных систем UNIX:

     ```
     /cdrom/program files/IBM/SQLLIB/doc/htmlcd/%L/index.htm
     ```

где */cdrom/* - положение, где монтируется компакт-диск, а %L необходимая вам национальная версия документации, например, **ru_RU** для русского языка.

Задачи, связанные с данной темой:

- “Поиск тем при обращении к Информационному центру DB2 из браузера” на стр. 395
- “Копирование файлов с компакт-диска Документация по DB2 в формате HTML на Web-сервер” на стр. 401

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 383

Обновление документации HTML, установленной на вашем компьютере

Теперь есть возможность обновлять файлы HTML, установленные с компакт-диска *Документация по DB2 в формате HTML*, по мере поступления обновлений от IBM. Это можно сделать одним из следующих способов:

- С помощью Информационного центра (если у вас установлены инструменты управления DB2 с графическим интерфейсом).
- С помощью загрузки и применения пакета обновлений FixPak для документации HTML DB2.

Примечание: Эти изменения затронут НЕ программный код DB2, а лишь документацию HTML, установленную с компакт-диска *Документация по DB2 в формате HTML*.

Процедура:

Чтобы изменить вашу локальную документацию с помощью Информационного центра:

1. Запустите Информационный центр DB2 одним из следующих способов:
   - На панели графических инструментов управления щелкните по значку **Информационный центр**. Этот пункт можно также выбрать в меню **Справка**.
   - Введите в командной строке **db2ic**.
2. Убедитесь, что у вашего компьютера есть доступ в Интернет; при необходимости программа обновления будет загружать последние пакеты документации FixPak с сервера IBM.
3. Чтобы начать обновление, выберите в меню **Информационный центр —> Обновить локальную документацию**.
4. Если требуется, введите информацию о вашем прокси-сервере, чтобы соединиться с Интернетом.

При наличии свежего пакета документации FixPak Информационный центр загрузит и применит его.

Чтобы загрузить и применить пакет документации FixPak вручную:

1. Убедитесь, что ваш компьютер соединен с Интернетом.
2. Откройте в вашем браузере страницу поддержки DB2: www.ibm.com/software/data/db2/udb/winos2unix/support.
3. Перейдите по ссылке для Версии 8 и найдите ссылку ″Documentation FixPaks″ (Пакеты документации FixPak).
4. Определите, устарела ли версия вашей локальной документации, сравнив уровень пакета FixPak с уровнем установленной у вас документации. Текущая документация на вашем компьютере имеет следующий уровень: **DB2 v8.1 GA**.
5. Если доступна более новая версия документации, загрузите пакет FixPak для вашей операционной системы. Один пакет FixPak используется для всех платформ Windows, другой пакет FixPak - для всех платформ UNIX.
6. Примените пакет FixPak:
   - Для операционных систем Windows: Пакет документации FixPak - это самораспаковывающийся zip-архив. Поместите загруженный пакет FixPak в пустой каталог и запустите его там. Будет создан исполняемый файл **setup**, при запуске которого начинается установка пакета FixPak.
   - Для операционных систем UNIX: Пакет документации FixPak - это упакованный файл tar.Z. Распакуйте и разархивируйте этот файл. При этом будет создан каталог `delta_install` со сценарием **installdocfix**. Запустите этот сценарий, чтобы установить пакет документации FixPak.

Задачи, связанные с данной темой:

- “Копирование файлов с компакт-диска Документация по DB2 в формате HTML на Web-сервер” на стр. 401

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 383

Копирование файлов с компакт-диска Документация по DB2 в формате HTML на Web-сервер

Вся библиотека с информацией DB2 поступает к вам на компакт-диске *Документация DB2 в формате HTML*; для облегчения доступа к ней ее можно установить на Web-сервере. Для этого просто скопируйте эту документацию на нужных вам языках на ваш Web-сервер.

Примечание: При обращении к документации HTML с Web-сервера через низкоскоростное соединение загрузка может идти медленно.

Процедура:

Чтобы скопировать на Web-сервер файлы с компакт-диска *Документация по DB2 в формате HTML*, используйте соответствующий путь источника:

- Для операционных систем Windows:
  `E:\program files\IBM\SQLLIB\doc\htmlcd\%L\*.*`

  где *E* - буква дисковода компакт-дисков, а *%L* - идентификатор языка.
- Для операционных систем UNIX:
  `/cdrom/program files/IBM/SQLLIB/doc/htmlcd/%L/*.*`

  где *cdrom* - точка монтирования компакт-диска, а *%L* - идентификатор языка.

Задачи, связанные с данной темой:

- “Поиск в документации DB2” на стр. 403

Ссылки, связанные с данной темой:

- “Поддерживаемые DB2 языки интерфейса, национальные версии и кодовые страницы” в *Quick Beginnings for DB2 Servers*
- “Обзор технической информации DB2 Universal Database” на стр. 383

Устранение ошибок при поиске в документации DB2 с помощью Netscape 4.x

Большинство проблем при поиске связаны с поддержкой Java, обеспечиваемой браузерами. В этой задаче описываются возможные обходные приемы для этих проблем.

Процедура:

При работе с Netscape 4.x обычно возникает проблема отсутствия или неверного местонахождения класса защиты. Попытайтесь применить описанный ниже прием, в особенности если на консоли Java браузера появилась следующая строка:

```
Невозможно найти класс java/security/InvalidParameterException
```

- В операционных системах Windows:

  Скопируйте с *компакт-диска документации HTML DB2* файл *x*`:\program files\IBM\SQLLIB\doc\htmlcd\`*locale*`\InvalidParameterException.class`, где *x* - буква дисковода компакт-дисков, а *locale* - нужная национальная версия, в подкаталог `java\classes\java\security\` каталога установки вашего браузера Netscape.

  Примечание: Возможно, надо будет создать подкаталоги `java\security\`.
- В операционных системах UNIX:

Скопируйте с компакт-диска *Документация по DB2 в формате HTML* файл */cdrom*/program files/IBM/SQLLIB/doc/htmlcd/*locale*/InvalidParameterException.class, где *cdrom* - точка монтирования компакт-диска, а *locale* - нужная национальная версия, в подкаталог java/classes/java/security/ каталога установки вашего браузера Netscape.

Примечание: Возможно, надо будет создать подкаталоги java/security/.

Если ваш браузер Netscape по-прежнему не может вывести окно ввода поиска, попытайтесь сделать следующее:

- Закройте все экземпляры браузеров Netscape, чтобы в компьютере не выполнялся программный код Netscape. Затем откройте новый экземпляр браузера Netscape и попытайтесь выполнить поиск снова.
- Очистите кэш браузера.
- Попробуйте использовать другую версию Netscape или другой браузер.

Задачи, связанные с данной темой:

- “Поиск в документации DB2” на стр. 403

Поиск в документации DB2

Необходимую вам информацию можно найти в библиотеке документации DB2. Если щелкнуть по значку поиска на навигационной панели инструментов Информационного центра DB2 (при обращении из браузера), откроется всплывающее окно поиска. Загрузка результатов поиска может занять некоторое время в зависимости от скорости вашего компьютера и сети.

Предварительные требования:

Требуется Netscape Версии 6.1 или новее или же Microsoft Internet Explorer Версии 5 или новее. В вашем браузере должна быть включена поддержка Java.

Ограничения:

Ограничения при поиске документации:

- Поиск не регистрозависим.
- Логические условия поиска не поддерживаются.
- Поиск с символами подстановки и частичный поиск не поддерживается. Так, при поиске *java** (или *java*) это вхождение будет восприниматься просто как строка символов *java** (или *java*), и, например, вхождение *javadoc* не будет найдено.

Процедура:

Для поиска документации DB2:

1. Щелкните по значку **Поиск** на панели инструментов навигации.
2. В верхнем текстовом поле ввода окна Поиск введите (через пробел) один или несколько терминов, отражающих интересующую вас область, и нажмите кнопку **Поиск**. В поле **Результаты** будет выведен список тем, ранжированных в порядке точности соответствия условиям поиска. Число рядом с каждым результатом поиска отражает точность соответствия (чем больше число, тем лучше соответствие).

   Ввод дополнительных слов для поиска повышает точность запроса, сокращая количество возвращаемых тем.
3. В списке **Результаты** щелкните по заголовку интересующей вас темы. Информация по этой теме будет выведена во фрейме содержимого Информационного центра DB2.

Примечание: При выполнении поиска его первый результат (с высшим рангом соответствия) автоматически загружается во фрейм браузера. Чтобы просмотреть содержимое других результатов поиска, щелкните по нужному результату в списке результатов.

Задачи, связанные с данной темой:

- “Устранение ошибок при поиске в документации DB2 с помощью Netscape 4.x” на стр. 402

Электронная информации об устранении неисправностей DB2

В выпуске DB2® UDB Версии 8 больше нет *Руководства по устранению неисправностей*. Информация по устранению неисправностей, ранее содержавшаяся в этом руководстве, теперь включена в другие публикации по DB2. Это позволяет давать вам наиболее свежую доступную информацию. Чтобы найти информацию по утилитам и функциям устранения неисправностей DB2, вызовите Информационный центр DB2 из любого инструмента DB2.

Если вы сталкиваетесь с проблемами и вам нужна помощь в поиске причин и решений, обратитесь на сайт поддержки DB2 (DB2 Online Support). Этот сайт содержит большую, постоянно обновляемую базу данных публикаций DB2, технических замечаний, записей APAR (о проблемах с продуктом), пакетов FixPaks и прочих ресурсов. Для решения ваших проблем можно воспользоваться поиском по сайту.

Сайт поддержки DB2 можно вызвать по адресу www.ibm.com/software/data/db2/udb/winos2unix/support, а также нажатием кнопки **Электронная поддержка** в Информационном центре DB2. На этом сайте теперь доступна также часто обновляемая информация, например, список внутренних кодов ошибок DB2.

Понятия, связанные с данным:

- “Информационный центр DB2 при обращении из браузера” на стр. 407

Задачи, связанные с данной темой:

- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 398

Доступность

Функции доступности помогают пользователям с физическими недостатками, например с ограниченной подвижностью или недостаточным зрением, с успехом пользоваться программными продуктами. В DB2® Universal Database Версии 8 применяются следующие основные функции доступности:

- DB2 позволяет использовать клавиатуру вместо мыши для работы с любыми функциями. Смотрите раздел “Ввод с клавиатуры и навигация”.
- DB2 позволяет настраивать размер и цвет шрифтов. Смотрите раздел “Доступность и дисплей”.
- DB2 позволяет использовать как визуальные, так и звуковые средства оповещения. Смотрите раздел “Альтернативные средства предупреждения” на стр. 406.
- DB2 поддерживает возможности доступности в программах, которые используют API доступности Java™. Смотрите раздел “Совместимость с технологиями для людей с физическими недостатками” на стр. 406.
- DB2 поставляется с документацией в формате, обеспечивающем доступность. Смотрите раздел “Удобный формат документации” на стр. 406.

Ввод с клавиатуры и навигация

Ввод с клавиатуры

Можно работать с инструментами DB2, используя только клавиатуру. Для выполнения операций вместо мыши можно использовать также клавиши или сочетания клавиш.

Фокус ввода с клавиатуры

В системах на основе UNIX фокус ввода с клавиатуры выделяется на экране; тем самым указывается активная область окна, в которую будут вводиться символы при нажатии клавиш.

Доступность и дисплей

В инструментах DB2 используются средства, улучшающие пользовательский интерфейс и облегчающие работу для пользователей со слабым зрением. К них относится поддержка настраиваемых свойств шрифтов.

Параметры шрифтов

Инструменты DB2 позволяют вам при помощи записной книжки Свойства инструментов выбрать цвет, размер и тип шрифта, используемого в меню и для диалоговых окон.

Независимость от цвета

Чтобы использовать любые функции этого продукта, вам не требуется различать цвета.

Альтернативные средства предупреждения

Вы можете задать, в каком виде получать оповещения: в виде звуковых или визуальных сигналов.

Совместимость с технологиями для людей с физическими недостатками

Интерфейс инструментов DB2 поддерживает API доступности Java, что позволяет использовать программы чтения экрана и другие технологии для пользователей с физическими недостатками.

Удобный формат документации

Документация для продуктов семейства DB2 доступна в формате HTML. Это позволяет просматривать документацию, используя предпочтения экрана, заданные для вашего браузера. Это позволяет также использовать программы чтения с экрана и другие технологии для людей с физическими недостатками.

Обучающие программы DB2

Обучающие программы DB2® помогают освоить различные аспекты DB2 Universal Database. Эти программы содержат уроки с пошаговыми указаниями по разработке программ, настройке производительности запросов SQL, работе с хранилищами данных, управлением метаданными и разработке Web-служб, использующих DB2.

Прежде, чем вы начнете:

Прежде чем обращаться к обучающим программам по приведенным ниже ссылкам, надо установить эти программы с компакт-диска *Документация по DB2 в формате HTML*.

Если вы не хотите устанавливать обучающие программы, можно просматривать их HTML-версии непосредственно с компакт-диска *Документация по DB2 в формате HTML*. На компакт-диске *Документация по DB2 в формате PDF* доступны также версии этих обучающих программ в формате PDF.

В некоторых уроках используются примеры данных или кодов программ. Описание необходимых условий для выполнения задач разных обучающих программ смотрите отдельно в каждой программе.

Обучающие программы DB2 Universal Database:

Если вы установили обучающие программы с компакт-диска *Документация по DB2 в формате HTML*, можно для просмотра материала щелкнуть по его заголовку в приведенном ниже списке.

Обучающая программа Business Intelligence Tutorial: Начальные сведения о Центре хранилищ данных
: Выполнение вводных задач работы с хранилищами данных при помощи Центра хранилищ данных.

Обучающая программа Business Intelligence Tutorial: Дополнительные уроки по хранилищам данных
: Выполнение дальнейших задач работы с хранилищами данных при помощи Центра хранилищ данных.

Обучающая программа по Центру разработки для Video Online с помощью Microsoft® Visual Basic
: Построение компонентов программ при помощи дополнительного модуля Development Center для Microsoft Visual Basic.

Обучающая программа по Центру каталогов данных
: Создание каталога данных для поиска и использования метаданных и управление им при помощи Центра каталогов данных.

Обучающая программа по Video Central для электронной коммерции
: Разработка и внедрение усовершенствованных программ DB2 Web Services с использованием продуктов WebSphere®.

Обучающая программа по Visual Explain
: Анализ, оптимизация и настройка операторов SQL для улучшения производительности при помощи Наглядного объяснения.

Информационный центр DB2 при обращении из браузера

Информационный центр DB2® дает доступ ко всей информации, необходимой для полного использования возможностей DB2 Universal Database™ и DB2 Connect™ в вашей работе. Информационный центр DB2 также содержит сведения по основным возможностям и компонентам DB2, включая репликацию, хранилища данных, Центр каталогов данных, Life Sciences Data Connect и модули расширения DB2.

Информационный центр DB2 при обращении из браузера Netscape Navigator Версии 6.1 или новее или Microsoft Internet Explorer Версии 5 или новее

поддерживает перечисленные ниже возможности. Для некоторых из них требуется включить поддержку Java или JavaScript:

Регулярно обновляемая документация
Постоянное обновление тем путем загрузки новейших файлов HTML.

Поиск Поиск по всем темам, установленным на вашей рабочей станции, после щелчка по значку **Поиск** на панели инструментов навигации.

Интегрированное дерево навигации
Поиск любой темы в библиотеке DB2 в одном дереве навигации. По типу содержащейся в нем информации дерево навигации организовано так:

- Задачи содержат пошаговые инструкции по достижению цели.
- Понятия помогают раскрыть содержание вопроса.
- Справочные темы содержат подробную информацию по вопросу, в том числе синтаксис операторов и команд, справку по сообщениям, требования.

Главный указатель
Доступ к информации, установленной с компакт-диска *Документация по DB2 в формате HTML* производится из главного указателя. Термины в указателе располагаются в алфавитном порядке.

Главный глоссарий
В главном глоссарии даются определения терминов, используемых Центром информации DB2. Термины в глоссарии располагаются в алфавитном порядке.

Задачи, связанные с данной темой:

- “Поиск тем при обращении к Информационному центру DB2 из браузера” на стр. 395
- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 398
- “Обновление документации HTML, установленной на вашем компьютере” на стр. 400

Приложение J. 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/регионе 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/регионе, где подобные заявления противоречат местным законам: КОРПОРАЦИЯ INTERNATIONAL BUSINESS MACHINES ПРЕДСТАВЛЯЕТ ДАННУЮ ПУБЛИКАЦИЮ “КАК ЕСТЬ” БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ПРЕДПОЛАГАЕМЫЕ ГАРАНТИИ СОВМЕСТИМОСТИ, РЫНОЧНОЙ ПРИГОДНОСТИ И СООТВЕТСТВИЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. В некоторых странах для определенных сделок подобные оговорки не допускаются; таким образом, это утверждение может не относиться к вам.

Данная информация может содержать технические неточности и типографские опечатки. Периодически в информацию вносятся изменения, они будут включены в новые издания этой публикации. Фирма IBM может в любое время без уведомления вносить изменения и усовершенствования в продукты и программы, описанные в этой публикации.

Любые ссылки в данной информации на Web-сайты, не принадлежащие IBM, приводятся только для удобства и никоим образом не означают поддержки IBM этих Web-сайтов. Материалы этих Web-сайтов не являются частью данного продукта IBM, и вы можете использовать их только на собственную ответственность.

IBM может использовать или распространять присланную вами информацию любым способом, как фирма сочтет нужным, без каких-либо обязательств перед вами.

Если обладателю лицензии на данную программу понадобятся сведения о возможности: (i) обмена данными между независимо разработанными программами и другими программами (включая данную) и (ii) совместного использования таких данных, он может обратиться по адресу:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

Такая информация может быть предоставлена на определенных условиях (в некоторых случаях к таким условиям может относиться оплата).

Лицензированная программа, описанная в данном документе, 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 получены 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. *_вставьте год или годы_*. Все права защищены.

Товарные знаки

Следующие термины, используемые по крайней мере в одном из документов библиотеки документации DB2 UDB, являются товарными знаками корпорации International Business Machines в Соединенных Штатах и/или в других странах.

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

Следующие термины, используемые по крайней мере в одном из документов библиотеки документации DB2 UDB, являются товарными знаками или зарегистрированными товарными знаками других компаний:

Microsoft, Windows, Windows NT и логотип Windows - товарные знаки Microsoft Corporation в Соединенных Штатах и в других странах.

Intel и Pentium - товарные знаки Intel Corporation в Соединенных Штатах и/или других странах.

Java и все товарные знаки на основе Java - товарные знаки Sun Microsystems, Inc. в Соединенных Штатах и/или в других странах.

UNIX - зарегистрированный товарный знак The Open Group в Соединенных Штатах и в других странах.

Названия других компаний, продуктов и услуг могут быть товарными знаками или марками сервиса других фирм.

Индекс

И

К

М

Н

О

П

P

C

Т

У

Ф

Ц

Ч

Как связаться с IBM

В Соединенных Штатах позвоните по одному из следующих номеров:

- 1-800-237-5511, чтобы обратиться в службу поддержки заказчиков
- 1-888-426-4343, чтобы узнать о доступных формах обслуживания.
- 1-800-IBM-4YOU (426-4968), чтобы обратиться в отдел маркетинга и продаж DB2

В Канаде позвоните по одному из следующих номеров:

- 1-800-IBM-SERV (1-800-426-7378), чтобы обратиться в службу поддержки заказчиков
- 1-800-465-9600, чтобы узнать о доступных формах обслуживания.
- 1-800-IBM-4YOU (1-800-426-4968), чтобы обратиться в отдел маркетинга и продаж DB2

Адрес отделения IBM в вашей стране или регионе можно найти на странице IBM Directory of Worldwide Contacts в Интернете по адресу www.ibm.com/planetwide

Информация о продукте

Информацию о продуктах DB2 Universal Database можно получить по телефону или в Интернете по адресу www.ibm.com/software/data/db2/udb

Этот сайт содержит свежую информацию по технической библиотеке, заказу книг, загружаемые клиенты, группы новостей, пакеты FixPaks, новости и ссылки на ресурсы в Интернете.

Если вы находитесь в США, позвоните по одному из следующих номеров:

- 1-800-IBM-CALL (1-800-426-2255), чтобы заказать продукты или получить общую информацию.
- 1-800-879-2755, чтобы заказать публикации.

Информацию о том, как связаться с IBM из других стран, смотрите на странице IBM Worldwide по адресу www.ibm.com/planetwide

Напечатано в Дании

SH43-0187-00

Spine information:

IBM® DB2
Universal Database™

Справочное руководство по восстановлению данных и высокой доступности

Версия 8