IBM® DB2 Universal Database™

Руководство администратора: Реализация

Версия 8

GH43-0202-00

IBM® DB2 Universal Database™

Руководство администратора: Реализация

Версия 8

GH43-0202-00

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

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

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

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

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

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

Для кого предназначена эта книга

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

Структура книги

Эта книга содержит сведения на следующие основные темы:

Реализация вашего проекта

- В разделе Глава 1, “Перед созданием базы данных” описываются предварительные требования для создания базы данных и объектов в базе данных.
- В разделе Глава 2, “Создание базы данных” описаны задачи, связанные с созданием базы данных и объектов в базе данных.
- В разделе Глава 3, “Изменение базы данных” описываются предварительные требования и задачи, связанные с изменением и отбрасыванием базы данных и объектов в базе данных.

Защита базы данных

- В разделе Глава 4, “Управление доступом к базам данных” описывается, как управлять доступом к ресурсам вашей базы данных.
- В разделе Глава 5, “Аудит действий DB2” описывается, как обнаруживать и отслеживать нежелательные или непредвиденные попытки обращения к данным.

Приложения

- В разделе Приложение A, “Правила именования” приводятся правила именования баз данных и объектов.
- В разделе Приложение B, “Службы каталога протокола LDAP (Lightweight Directory Access Protocol)” приводится информация о том, как пользоваться службами каталога LDAP.
- В разделе Приложение C, “Выполнение команд на нескольких разделах базы данных” обсуждается использование сценариев оболочки *db2_all* и *rah* для отправки команд всем разделам в среде многораздельных баз данных.
- Приложение D, “Поддержка Windows Management Instrumentation (WMI)”, содержит информацию об управлении DB2 с помощью WMI.
- В разделе Приложение E, “Как DB2 for Windows NT работает с защитой Windows NT” описывается, как DB2 работает с защитой Windows.
- Приложение F, “Работа с монитором производительности Windows”, описывает применение Монитора производительности Windows для сбора данных о производительности DB2.
- В разделе Приложение G, “Работа с серверами разделов баз данных Windows” описываются утилиты, используемые Windows для работы с серверами разделов базы данных.
- В разделе Приложение H, “Конфигурирование нескольких логических узлов” описывается, как конфигурировать несколько логических узлов в среде многораздельной базы данных.
- В разделе Приложение I, “Расширение Центра управления” приводится информация о том, как расширить Центр управления, добавив новые кнопки на панель инструментов, новые действия, новые определения объектов и действий.

Эта глава была перемещена из руководства *Руководство администратора: Реализация* “Утилиты перемещения данных”.

Примечание: Вся информация об утилитах DB2 для перемещения данных и связанные вопросы из книг *Command Reference* и *Administrative API Reference* собраны в книге *Data Movement Utilities Guide and Reference*.

Книга *Data Movement Utilities Guide and Reference* - это основной источник информации по данным темам.

Дополнительная информация о репликации данных приведена в разделе *Replication Guide and Reference*.

Эта глава была перемещена из руководства *Руководство администратора: Реализация* “Восстановление базы данных”.

Примечание: Вся информация о способах и инструментах для резервного копирования и восстановления данных и связанные вопросы из книг *Command Reference* и *Administrative API Reference* собраны в книге *Справочное руководство по восстановлению данных и высокой доступности*.

Книга *Справочное руководство по восстановлению данных и высокой доступности* - это основной источник информации по данным темам.

Краткий обзор других томов Руководства администратора

Руководство администратора: Планирование

Книга *Руководство администратора: Планирование* посвящена разработке базы данных. В ней освещаются вопросы, связанные с логическими и физическими устройствами и распределенными транзакциями. Отдельные главы и приложения из этого тома кратко описаны здесь:

Концепции баз данных

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

Проектирование баз данных

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

Распределенная обработка транзакций

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

Приложения

- ″Несоответствия между выпусками″ содержит информацию о несоответствиях между версиями 7 и 8, а также возможных несоответствиях в будущем, которые необходимо учитывать.
- Приложение ″Поддержка национальных языков (NLS)″ описывает поддержку национальных языков в DB2, включая информацию о странах, языках и кодовых страницах.

Руководство администратора: производительность

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

Производительность -- Введение

- Раздел ″Введение в производительность″ знакомит с идеями и особенностями управления производительностью DB2 UDB и ее повышения.
- В разделе ″Архитектура и процессы″ излагаются основы архитектуры и процессов DB2 Universal Database.

Настройка производительности прикладных программ

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

Настройка и конфигурирование вашей системы

- В разделе ″Производительность работы″ дается обзор использования памяти менеджером баз данных и описываются другие особенности, влияющие на производительность во время выполнения.
- В разделе ″Использование утилиты ограничения ресурсов″ излагаются начальные сведения об использовании утилиты ограничения ресурсов для управления некоторыми аспектами работы с базой данных.

- В разделе ″Масштабирование вашей конфигурации″ рассматриваются некоторые особенности и задачи, связанные с ростом размера ваших систем баз данных.
- В разделе ″Перераспределение данных между разделами базы данных″ обсуждаются задачи, возникающие в среде многораздельных баз данных в связи с перераспределением данных между разделами.
- В разделе ″Измерение производительности″ представлен обзор вопросов и способов измерения производительности.
- В разделе ″Настройка DB2″ обсуждаются файлы конфигурации менеджера баз данных и базы данных и значения параметров конфигурации DAS.

Приложения

- В приложении ″Реестр и переменные среды DB2″ описываются значения реестра профиля и переменных среды.
- В приложении ″Таблицы и определения объяснения″ описываются таблицы, используемые средствами объяснения DB2, и рассказывается, как создавать эти таблицы.
- В приложении ″Инструменты объяснения SQL″ описывается использование инструментов объяснения DB2: db2expln и dynexpln.
- В приложении ″db2exfmt – инструмент форматирования таблиц объяснения″ описывается использование этого инструмента объяснения DB2 для форматирования данных таблиц объяснения.

Часть 1. Реализация вашего проекта

Глава 1. Перед созданием базы данных

После определения структуры базы данных необходимо создать эту базу данных и объекты в ней. Эти объекты - схемы, группы разделов баз данных, табличные пространства, таблицы, производные таблицы, оболочки, службы, псевдонимы, отображения типов, отображения функций, алиасы, пользовательские типы (UDT), пользовательские функции (UDF), автоматические таблицы AST, триггеры, ограничения, индексы и пакеты. Эти объекты можно создать при помощи операторов SQL в процессоре командной строки или в прикладных программах.

Информацию об операторах SQL смотрите в руководстве *SQL Reference*. Информация о командах приведена в руководстве *Command Reference*. Дополнительную информацию об интерфейсах прикладного программирования (API) см. в руководстве *Administrative API Reference*.

Кроме того, вы можете создать объекты базы данных с помощью Центра управления. Центр управления можно использовать вместо операторов SQL, команд процессора командной строки или API.

В этой главе описание методов выполнения задач с помощью Центра управления выделены рамкой. Сразу после них описаны методы использования для тех же целей командной строки, в некоторых случаях с примерами. Для некоторых задач показан только один метод. Работая с Центром управления, не забывайте, что можно использовать электронную справку, содержащую более подробную информацию, чем представлена здесь.

Эта глава посвящена вопросам, которые нужно знать перед созданием базы данных со всеми ее объектами. Здесь изложены некоторые необходимые понятия и темы, а также описаны операции, которые должны быть выполнены перед созданием базы данных.

В следующей главе кратко описаны различные объекты, которые могут входить в структуру базы данных.

В последней главе этой части книги изложены вопросы, которые надо иметь в виду перед изменением базы данных, и описано изменение и отбрасывание объектов баз данных.

Там, где DB2 Universal Database взаимодействует с операционной системой, описание некоторых тем в этой и последующих главах может содержать отличия, зависящие от операционной системы. Вы можете воспользоваться преимуществами собственных средств операционной системы над средствами,

предлагаемыми DB2 UDB. Конкретные отличия описаны в книге *Быстрый старт* и в документации по операционной системе.

Например, Windows поддерживает тип прикладных программ, называемых “службами”. В DB2 for Windows каждый экземпляр DB2 можно определить как службу. Служба может автоматически запускаться при загрузке системы (этот режим может быть задан пользователем с помощью апплета панели управления Службы или 32-разрядной прикладной программой, использующей функции служб из 32-разрядного интерфейса прикладного программирования (API) Microsoft Windows. Службы могут работать, даже когда в системе не зарегистрирован ни один пользователь.

Если не указано обратное, информация о Windows 9x относится также к Windows 98 и Windows ME. Информация о Windows NT также относится к Windows NT, Windows 2000, Windows XP, и Windows .NET. Ссылка на Windows означает любую поддерживаемую операционную систему Windows.

Создание базы данных - предварительные требования

Перед созданием базы данных нужно рассмотреть следующие темы:

Запуск DB2 UDB в UNIX

При выполнении обычных рабочих операций может потребоваться запуск или остановка DB2; например, необходимо запустить экземпляр перед выполнением следующих заданий:

- Соединение с базой данных этого экземпляра
- Прекомпиляция прикладной программы
- Связывание пакета с базой данных
- Доступ к базам данных хоста.

Предварительные требования для установки:

Чтобы запустить экземпляр DB2 в вашей системе:

1. Зарегистрируйтесь под ID пользователя или именем пользователя, обладающим полномочиями SYSADM, SYSCTRL или SYSMAINT для этого экземпляра, или зарегистрируйтесь как владелец экземпляра.
2. Выполните сценарий запуска следующим образом:

```
. INSTHOME/sqllib/db2profile      (для оболочек Bourne или Korn)
source INSTHOME/sqllib/db2cshrc (для оболочки C)
```

где INSTHOME - начальный каталог нужного экземпляра.

Процедура:

Для запуска экземпляра используйте один из следующих двух методов:

1. Чтобы запустить экземпляр с помощью Центра управления:

a. Раскройте дерево объектов и найдите папку **Экземпляры**.
b. Щелкните правой кнопкой мыши по нужному экземпляру и выберите из всплывающего меню пункт **Запустить**.

2. Чтобы запустить экземпляр из командной строки, введите команду:

```
db2start
```

Задачи, связанные с данной темой:

- “Остановка экземпляра DB2 в UNIX” на стр. 16
- “Выбор текущего экземпляра” на стр. 29
- “Запуск DB2 UDB в Windows” на стр. 5

Запуск DB2 UDB в Windows

При выполнении обычных рабочих операций может потребоваться запуск или остановка DB2; например, необходимо запустить экземпляр перед выполнением следующих заданий:

- Соединение с базой данных этого экземпляра
- Прекомпиляция прикладной программы
- Связывание пакета с базой данных
- Доступ к базам данных хоста.

Предварительные требования для:

Для запуска DB2 в качестве службы с помощью команды **db2start** у учетной записи пользователя должны быть привилегии, необходимые для запуска службы в операционной системе Windows NT. Этот пользователь может быть членом группы администраторов, операторов сервера или привилегированных пользователей.

Процедура:

Для запуска экземпляра используйте один из следующих двух методов:

1. Чтобы запустить экземпляр с помощью Центра управления:

   a. Раскройте дерево объектов и найдите папку **Экземпляры**.
   b. Щелкните правой кнопкой мыши по нужному экземпляру и выберите из всплывающего меню пункт **Запустить**.

2. Чтобы запустить экземпляр из командной строки, введите команду:

   ```
   db2start
   ```

Команда **db2start** запускает DB2 в качестве службы Windows. Для того чтобы запустить DB2 в Windows в виде процесса, укажите ключ ″/D″ в команде **db2start**. Запустить DB2 как службу можно также при помощи Панели управления или команды ″NET START″.

В среде многораздельной базы данных каждый сервер раздела базы данных запускается в виде службы Windows. В такой среде DB2 нельзя запустить в виде процесса, указав ключ ″/D″.

Задачи, связанные с данной темой:

- “Запуск DB2 UDB в UNIX” на стр. 4
- “Остановка экземпляра DB2 в UNIX” на стр. 16
- “Остановка экземпляра DB2 в системе Windows” на стр. 17

Несколько экземпляров менеджера баз данных

На одном сервере можно создать несколько экземпляров менеджера баз данных. Это означает, что можно создать несколько экземпляров одного продукта на одном физическом компьютере и затем выполнять эти экземпляры одновременно. Это обеспечивает гибкость настройки сред.

Несколько экземпляров можно создать, чтобы:

- Отделить среду разработки от производственной среды.
- Отдельно настроить каждый экземпляр для конкретных прикладных программ, которые он будет обслуживать.
- Защитить важную информацию от администраторов. Например, можно разместить базу данных с информацией о заработной плате в отдельном экземпляре, чтобы владельцы других экземпляров не имели доступ к этим данным.

Примечание: (Только в операционной системе UNIX®:) Для исключения конфликтов между экземплярами убедитесь, что в каждом

экземпляре применяется собственная домашняя файловая система. При использовании общей файловой системы могут возникнуть ошибки.

Файлы программ DB2® физически хранятся на одном компьютере в одном месте. Для каждого создаваемого экземпляра создается ссылка на это место, чтобы не дублировать файлы программ для каждого создаваемого экземпляра. Несколько связанных баз данных могут располагаться в одном экземпляре.

Экземпляры вносятся в каталог узлов как локальные или удаленные. Переменная среды DB2INSTANCE задает экземпляр по умолчанию. Можно подключиться (команда **ATTACH**) к другим экземплярам для выполнения операций, которые можно выполнить только на уровне экземпляра (например, создание базы данных, принудительное отсоединение прикладных программ, отслеживание работы базы данных или изменение конфигурации менеджера баз данных). При подключении к экземпляру, который не является экземпляром по умолчанию, информация о связи с этим узлом берется из каталога узлов.

Понятия, связанные с данным:

- “Несколько экземпляров в операционной системе UNIX” на стр. 23
- “Несколько экземпляров в операционной системе Windows” на стр. 24

Задачи, связанные с данной темой:

- “Создание дополнительных экземпляров” на стр. 25

Ссылки, связанные с данной темой:

- “ATTACH Command” в *Command Reference*

Подключение другого экземпляра менеджера базы данных

Для подключения другого экземпляра (возможно, удаленного) применяется команда **ATTACH**.

Предварительные требования:

Уже должно существовать более одного экземпляра.

Процедура:

Для подключения другого экземпляра менеджера баз данных с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Экземпляры**.
2. Выберите экземпляр, к которому нужно подключиться.
3. Щелкните правой кнопкой мыши по имени выбранного экземпляра.
4. В окне DB2 Подключение введите ID пользователя и пароль и нажмите кнопку **OK**.

Чтобы подключиться к экземпляру из командной строки, введите команду:

```
db2 attach to <имя экземпляра>
```

Например, чтобы подключиться к внесенному в каталог узлов экземпляру с именем `testdb2`, введите:

```
db2 attach to testdb2
```

Выполнив требуемые операции для экземпляра `testdb2`, можно затем отключиться **DETACH** от этого экземпляра, введя следующую команду:

```
db2 detach
```

Ссылки, связанные с данной темой:

- "ATTACH Command" в *Command Reference*
- "DETACH Command" в *Command Reference*

Группировка объектов по схеме

Имена объектов базы данных могут представлять собой один идентификатор или *имя со спецификатором схемы*, содержащее два идентификатора. Схема (первая часть двухчастного имени) - это средство классификации и группировки объектов в базе данных. При создании объекта (например, таблицы, производной таблицы, алиаса, особого типа, функции, индекса, пакета или триггера) он назначается одной из схем. Это делается явно или неявно.

При явном использовании схемы для объекта в операторе задается старшая часть двухчастного имени. Например, пользователь A использует оператор CREATE TABLE для создания таблицы в схеме C:

```
CREATE TABLE C.X (COL1 INT)
```

При неявном использовании схемы старшая часть двухчастного имени для объекта не задается. В этом случае имя схемы, используемое в качестве старшей части имени объекта, задается специальным регистром CURRENT SCHEMA. Начальное значение специального регистра CURRENT SCHEMA - это ID авторизации пользователя текущего сеанса. Если нужно изменить это значение во время текущего сеанса, можно при помощи оператора SET SCHEMA задать для этого специального регистра имя другой схемы.

При создании базы данных создаются некоторые объекты в определенных схемах, сохраняемых в системном каталоге.

В динамических операторах SQL в качестве спецификатора схемы для имен объектов, для которых не задана схема, неявно используется значение специального регистра CURRENT SCHEMA. В статических операторах SQL спецификатор схемы для имен объектов базы данных, для которых схема не задана, неявно задается опцией прекомпиляции-связывания QUALIFIER.

Перед созданием объектов нужно решить, создавать их в своей схеме или использовать другую схему для логической группировки объектов. При создании совместно используемых объектов использование другого имени схемы может давать значительные преимущества.

Понятия, связанные с данным:

- “Определение таблиц системного каталога” на стр. 75

Задачи, связанные с данной темой:

- “Создание схемы” на стр. 94

Ссылки, связанные с данной темой:

- “SET SCHEMA statement” в *SQL Reference, Том 2*
- “CURRENT SCHEMA special register” в *SQL Reference, Том 1*

Параллелизм

Чтобы использовать преимущества параллелизма для раздела базы данных или одноразделной базы данных, необходимо изменить параметры конфигурации. Например, внутриразделный параллелизм позволяет реализовать преимущества использования нескольких процессоров симметричного многопроцессорного компьютера (SMP).

Включение параллелизма запросов Sysplex

Процедура:

Межраздельный параллелизм применяется автоматически в зависимости от числа разделов базы данных и распределения данных между этими разделами.

Понятия, связанные с данным:

- “Среды с различным числом процессоров и разделов” в *Руководство администратора: Планирование*
- “Разделение данных” в *Руководство администратора: Планирование*
- “Проектирование групп разделов базы данных” в *Руководство администратора: Планирование*

- “Разделы базы данных” в *Руководство администратора: Производительность*

Задачи, связанные с данной темой:

- “Включение внутриразделького параллелизма для запросов” на стр. 10
- “Включение разделения данных в базе данных” на стр. 14
- “Перераспределение данных между разделами” в *Руководство администратора: Производительность*

Включение внутриразделького параллелизма для запросов

Процедура:

С помощью Центра управления можно узнать или изменить значения отдельных записей в файле конфигурации конкретной базы данных или менеджера баз данных.

Чтобы узнать значения отдельных записей в файле конфигурации конкретной базы данных или менеджера баз данных, можно также использовать команды **GET DATABASE CONFIGURATION** и **GET DATABASE MANAGER CONFIGURATION**. Чтобы изменить отдельные записи в файле конфигурации конкретной базы данных или менеджера баз данных, используйте соответственно команды **UPDATE DATABASE CONFIGURATION** и **UPDATE DATABASE MANAGER CONFIGURATION** .

На внутриразделький параллелизм влияют параметры конфигурации менеджера баз данных *max_querydegree* и *intra_parallel* и параметр конфигурации базы данных *dft_degree*.

Чтобы разрешить внутриразделький параллелизм запросов, необходимо изменить параметры конфигурации базы данных, параметры конфигурации менеджера баз данных, опции предкомпиляции и связывания или специальный регистр.

intra_parallel
: Данный параметр указывает, разрешено ли менеджеру баз данных выполнять параллельную обработку внутри раздела. По умолчанию такая обработка не поддерживается.

max_querydegree
: Данный параметр задает максимальную степень параллелизма внутри раздела для оператора SQL, выполняемого в данном экземпляре менеджера баз данных. При обработке оператора SQL число операций, выполняемых параллельно в одном разделе, не будет превосходить данного значения. Кроме того, параметр *intra_parallel* должен быть равен “YES”, если применяется значение *max_querydegree*. По умолчанию значение этого параметра конфигурации равно -1. Это

значение указывает, что система использует степень параллелизма, определенную оптимизатором; в противном случае используется заданное пользователем значение.

dft_degree
Параметр конфигурации базы данных. Задает значения по умолчанию для опции связывания DEGREE и специального регистра CURRENT DEGREE. Значение по умолчанию равно 1. Это значение указывает, что система использует степень параллелизма, определенную оптимизатором.

DEGREE
Опция прекомпиляции или связывания для статического SQL.

CURRENT DEGREE
Специальный регистр для динамического SQL.

Понятия, связанные с данным:

- "Параллельная обработка в прикладных программах" в *Руководство администратора: Производительность*
- "Информация о параллельной обработке" в *Руководство администратора: Производительность*

Задачи, связанные с данной темой:

- "Настройка DB2 с помощью параметров конфигурации" в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- "Параметр конфигурации Максимальная степень параллелизма запросов - max_querydegree" в *Руководство администратора: Производительность*
- "Параметр конфигурации Разрешение внутрираздельного параллелизма - intra_parallel" в *Руководство администратора: Производительность*
- "Параметр конфигурации Степень по умолчанию - dft_degree" в *Руководство администратора: Производительность*
- "BIND Command" в *Command Reference*
- "PRECOMPILE Command" в *Command Reference*
- "CURRENT DEGREE special register" в *SQL Reference, Том 1*

Включение внутрираздельного параллелизма для утилит

В этом разделе описано, как разрешить внутрираздельный параллелизм для следующих утилит:

- Загрузки
- Создания индекса
- Резервного копирования базы данных или табличного пространства

- Восстановления базы данных или табличного пространства

Межраздельный параллелизм для утилит применяется автоматически в зависимости от числа разделов базы данных.

Включение параллелизма при загрузке данных: Утилита загрузки использует параллелизм автоматически; однако вы можете задать следующие параметры команды **LOAD**:

- CPU_PARALLELISM
- DISK_PARALLELISM

В среде многораздельной базы данных межраздельный параллелизм при загрузке данных возникает автоматически, если целевая таблица определена на нескольких разделах. Межраздельный параллелизм при загрузке данных можно переопределить параметром OUTPUT_DBPARTNUMBS. Утилита загрузки также автоматически включает параллелизм разделения данных в зависимости от размера целевых разделов. Для управления максимальной степенью параллелизма может применяться параметр MAX_NUM_PART_AGENTS. Параллелизм разделения данных также можно переопределить параметром PARTITIONING_DBPARTNUMS, если указано ANYORDER.

Понятия, связанные с данным:

- “Load Overview” в *Data Movement Utilities Guide and Reference*
- “Loading Data in a Partitioned Database - Overview” в *Data Movement Utilities Guide and Reference*

Включение параллелизма при создании индексов: Чтобы разрешить параллелизм при создании индекса:

- Параметр конфигурации менеджера баз данных *intra_parallel* должен иметь значение ON
- Таблица должна быть достаточно большой, чтобы использовать преимущества параллелизма
- На компьютере SMP должны быть включены несколько процессоров.

Ссылки, связанные с данной темой:

- “Параметр конфигурации Разрешение внутрираздельного параллелизма - intra_parallel” в *Руководство администратора: Производительность*
- “CREATE INDEX statement” в *SQL Reference, Том 2*

Включение параллелизма операций ввода-вывода при создании резервной копии базы данных или табличного пространства: Чтобы разрешить параллелизм операций ввода-вывода при создании резервной копии базы данных или табличного пространства:

- Используйте несколько носителей назначения.

- Настройте параллельный ввод-вывод табличных пространств, определив несколько контейнеров или один контейнер с несколькими дисками, и задайте переменную реестра DB2_PARALLEL_IO. Если вы собираетесь реализовать параллелизм ввода-вывода, то перед определением контейнеров следует выполнить несколько предварительных действий. Эти действия нельзя выполнять по мере необходимости, их обязательно нужно предпринять до того, как потребуется создать резервную копию базы данных или табличного пространства.
- С помощью параметра PARALLELISM команды **BACKUP** задайте степень параллелизма.
- С помощью параметра WITH число-буферов BUFFERS команды **BACKUP** задайте достаточное число буферов для указанной степени параллелизма. Число буферов должно быть несколько больше, чем число носителей назначения плюс выбранная степень параллелизма.

  Для резервного копирования надо использовать буферы:

  – Максимального размера. Можно рекомендовать размер 4 Мбайта или 8 Мбайт (1024 или 2048 страниц).
  – Размер буферов должен быть не меньше размера (extentsize * число контейнеров) наибольшего табличного пространства, для которого создается резервная копия.

Ссылки, связанные с данной темой:

- “BACKUP DATABASE Command” в *Command Reference*

Включение параллелизма операций ввода-вывода при восстановлении базы данных или табличного пространства: Чтобы разрешить параллелизм операций ввода-вывода при восстановлении базы данных или табличного пространства:

- Используйте несколько исходных носителей.
- Сконфигурируйте табличные пространства для параллельных операций ввода-вывода. Это нужно сделать до того, как вы определите контейнер. Эти действия нельзя выполнять по мере необходимости, их обязательно нужно предпринять до того, как потребуется восстановить резервную копию базы данных или табличного пространства.
- С помощью параметра PARALLELISM команды **RESTORE** задайте степень параллелизма.
- С помощью параметра WITH число-буферов BUFFERS команды **RESTORE** задайте достаточное число буферов для указанной степени параллелизма. Число буферов должно быть несколько больше, чем число носителей назначения плюс выбранная степень параллелизма.

  Используйте для восстановления буферы:

  – Максимального размера. Можно рекомендовать размер 4 Мбайта или 8 Мбайт (1024 или 2048 страниц).

– Размер буферов должен быть не меньше размера (extentsize * число контейнеров) наибольшего восстанавливаемого табличного пространства.
– Размер буферов должен совпадать с размером буферов резервного копирования или быть кратен ему.

Ссылки, связанные с данной темой:

• “RESTORE DATABASE Command” в *Command Reference*

Включение разделения данных в базе данных

Решение о том, должна ли база данных работать в среде с несколькими разделами, следует принять до создания базы данных. Разбиение базы данных на разделы - это один из способов повышения производительности через усовершенствование структуры базы данных.

Некоторые условия, связанные с созданием базы данных с разделами, приведены здесь.

Процедура:

При работе в среде многораздельных баз данных можно создать базу данных с любого узла, описанного в файле `db2nodes.cfg`, используя команду **CREATE DATABASE** или функцию sqlecrea() интерфейса прикладного программирования (API).

Перед создание базы данных нужно также выбрать раздел базы данных, который будет узлом каталога для этой базы данных. Затем можно создать базу данных непосредственно из этого раздела или из удаленного клиента, подключенного к этому разделу. Раздел базы данных, к которому вы подключитесь и на котором выполните команду CREATE DATABASE, станет *узлом каталога* для этой конкретной базы данных.

Узел каталога - это раздел базы данных, на котором хранятся все таблицы системного каталога. Все обращения к системным таблицам идут через этот раздел базы данных. Все объекты базы данных объединения (оболочки, серверы, псевдонимы и т.д.) хранятся в таблицах системного каталога на этом узле.

Если это возможно, создавайте каждую базу данных в отдельном экземпляре. Если это невозможно (то есть необходимо создать несколько баз данных в одном экземпляре), разместите узлы каталогов на разных разделах баз данных. Это уменьшит число конфликтов при обращении к информации каталога на одном узле базы данных.

Примечание: Следует регулярно делать резервную копию узла каталога и по возможности избегать размещения на нем пользовательских данных, поскольку это увеличит время, необходимое для резервного копирования.

При создании базы данных она автоматически создается на всех разделах базы данных, определенных в файле `db2nodes.cfg`.

При создании в системе первой базы данных формируется системный каталог баз данных. В него добавляется информация о всех других создаваемых базах данных. При работе в UNIX системный каталог баз данных называется `sqldbdir` и расположен в подкаталоге `sqllib` вашего начального каталога. Этот каталог должен находиться в совместно используемой файловой системе (например, NFS на платформах UNIX), поскольку один и тот же системный каталог баз данных используется для всех разделов базы данных, входящих в многораздельную базу данных. При работе в Windows системный каталог баз данных находится в каталоге экземпляра.

В каталоге `sqldbdir` также находится файл системных значений. Он называется `sqldbins` и обеспечивает синхронизацию разделов базы данных. Этот файл также должен находиться в совместно используемой файловой системе, поскольку он используется для всех разделов базы данных. Все разделы базы данных обращаются к этому файлу.

Чтобы использовать преимущества разделения данных, необходимо изменить параметры конфигурации. Чтобы узнать значения отдельных записей в файле конфигурации конкретной базы данных или менеджера баз данных, можно также использовать команды **GET DATABASE CONFIGURATION** и **GET DATABASE MANAGER CONFIGURATION** . Чтобы изменить отдельные записи в файле конфигурации конкретной базы данных или менеджера баз данных, используйте соответственно команды **UPDATE DATABASE CONFIGURATION** и **UPDATE DATABASE MANAGER CONFIGURATION** .

На многораздельные базы данных влияют следующие параметры конфигурации менеджера баз данных: *conn_elapse*, *fcm_num_anchors*, *fcm_num_buffers*, *fcm_num_connect*, *fcm_num_rqb*, *max_connretries*, *max_coordagents*, *max_time_diff*, *num_poolagents* и *stop_start_time*.

Задачи, связанные с данной темой:

- “Настройка DB2 с помощью параметров конфигурации” в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- “sqlecrea - Create Database” в *Administrative API Reference*
- “CREATE DATABASE Command” в *Command Reference*

Остановка экземпляра DB2 в UNIX

Вам может потребоваться остановить текущий экземпляр менеджера баз данных (DB2).

Предварительные требования:

Чтобы остановить экземпляр DB2 в вашей системе, необходимо сделать следующее:

1. Зарегистрируйтесь или подключитесь к экземпляру с ID пользователя или именем пользователя, обладающим полномочиями SYSADM, SYSCTRL или SYSMAINT для этого экземпляра, или зарегистрируйтесь как владелец экземпляра.
2. Выведите список всех прикладных программ и пользователей, соединенных с конкретной базой данных, которую нужно остановить. Просмотрите список прикладных программ, чтобы убедиться, что не запущены особо важные или критические программы. Для этого требуются полномочия SYSADM, SYSCTRL или SYSMAINT.
3. Принудительно отсоедините от базы данных все прикладные программы и всех пользователей. Для принудительного отсоединения пользователей требуются полномочия SYSADM или SYSCTRL.

Ограничения:

Команду **db2stop** можно выполнить только на сервере.При выполнении этой команды не должно быть соединений с базами данных; если же какие-либо подключения к экземплярам есть, они будут принудительно отключены перед остановкой DB2.

Примечание: Если к экземпляру подключены сеансы процессора командной строки, перед командой **db2stop** для каждого такого сеанса нужно выполнить команду **terminate**, чтобы завершить этот сеанс. Команда **db2stop** останавливает экземпляр, задаваемый переменной среды DB2INSTANCE.

Процедура:

Для остановки экземпляра используйте один из следующих двух способов:

1. Чтобы остановить экземпляр с помощью Центра управления:

   a. Раскройте дерево объектов и найдите папку **Экземпляры**.
   b. Выберите все экземпляры, которые нужно остановить.
   c. Щелкните правой кнопкой мыши по выбранным экземплярам и выберите из всплывающего меню пункт **Остановить**.
   d. В окне Подтверждение остановки нажмите кнопку **OK**.

2. Чтобы остановить экземпляр из командной строки, введите команду:

   ```
   db2stop
   ```

Отдельные разделы в среде базы данных с несколькими разделами можно остановить с помощью команды db2stop. Если при работе в базе данных с разделами вы останавливаете логический раздел с помощью команды

```
db2stop drop nodenum <0>
```

то обязательно убедитесь, что другие пользователи не пытаются обратиться с базе данных. В противном случае, вам будет отправлено сообщение SQL6030N.

Ссылки, связанные с данной темой:

- “db2stop - Stop DB2 Command” в *Command Reference*
- “TERMINATE Command” в *Command Reference*

Остановка экземпляра DB2 в системе Windows

Может потребоваться остановка экземпляра менеджера баз данных (DB2).

Предварительные требования:

Чтобы остановить экземпляр DB2 в вашей системе, необходимо сделать следующее:

1. Пользователь, останавливающий службу DB2, должен обладать необходимыми привилегиями в операционной системе Windows. Этот пользователь может быть членом группы администраторов, операторов сервера или привилегированных пользователей.
2. Выведите список всех прикладных программ и пользователей, соединенных с конкретной базой данных, которую нужно остановить. Просмотрите список прикладных программ, чтобы убедиться, что не запущены особо важные или критические программы. Для этого требуются полномочия SYSADM, SYSCTRL или SYSMAINT.
3. Принудительно отсоедините от базы данных все прикладные программы и всех пользователей. Для принудительного отсоединения пользователей требуются полномочия SYSADM или SYSCTRL.

Ограничения:

Команду **db2stop** можно выполнить только на сервере.При выполнении этой команды не должно быть соединений с базами данных; если же какие-либо подключения к экземплярам есть, они будут принудительно отключены перед остановкой DB2.

Примечание: Если к экземпляру подключены сеансы процессора командной строки, перед командой **db2stop** для каждого такого сеанса нужно выполнить команду **terminate**, чтобы завершить этот сеанс. Команда **db2stop** останавливает экземпляр, задаваемый переменной среды DB2INSTANCE.

Процедура:

Для остановки экземпляра DB2 в системе воспользуйтесь одним из следующих способов:

- **db2stop**
- Из Центра управления

1. Раскройте дерево объектов и найдите папку **Экземпляры**.
2. Выберите все экземпляры, которые нужно остановить.
3. Щелкните правой кнопкой мыши по выбранным экземплярам и выберите из всплывающего меню пункт **Остановить**.
4. В окне Подтверждение остановки нажмите кнопку **OK**.

- Командой “NET STOP”
- Из приложения.

Учтите, что в многораздельной базе данных каждый сервер раздела запускается как отдельная служба. Необходимо остановить каждую из них.

Ссылки, связанные с данной темой:

- “db2stop - Stop DB2 Command” в *Command Reference*

Подготовка к созданию базы данных

Перед созданием базы данных вам придется выполнить несколько предварительных действий. В число этих действий входит разработка структуры базы данных, создание экземпляра, каталогов и других вспомогательных файлов. Все эти задачи рассмотрены в следующих разделах:

- Разработка логических или физических характеристик базы данных
- Создание экземпляра
- Переменные среды и реестра профиля

- Сервер администратора DB2
- Создание файла конфигурации узлов
- Создание файла конфигурации базы данных
- Соединения Менеджера быстрой связи (FCM)

Разработка логических или физических характеристик базы данных

Перед созданием базы данных нужно принять решения, касающиеся ее логической и физической структуры. Дополнительную информацию о логической и физической структуре базы данных смотрите в руководстве *Руководство администратора: Планирование*.

Создание экземпляра

Экземпляр - это логическая среда менеджера баз данных, в которой создается каталог баз данных и задаются параметры конфигурации. В зависимости от потребностей, можно создать несколько экземпляров. Несколько экземпляров позволяют:

- Использовать один экземпляр для среды разработки, а другой - для производственной среды.
- Настроить экземпляр для конкретной среды.
- Ограничить доступ к критической информации.
- Управлять назначением полномочий SYSADM, SYSCTRL и SYSMAINT для каждого экземпляра.
- Оптимизировать конфигурацию менеджера баз данных для каждого экземпляра.
- Ограничить последствия ошибок в работе экземпляра. При возникновении такой ошибки она повлияет на работу только одного экземпляра. Другие экземпляры смогут продолжать работать нормально.

Следует отметить, что использование нескольких экземпляров имеет некоторые недостатки:

- Для каждого экземпляра требуются дополнительные системные ресурсы (виртуальная память и дисковое пространство).
- Увеличиваются затраты на управление, поскольку нужно управлять дополнительными экземплярами.

Вся информация об экземпляре базы данных хранится в каталоге экземпляра. После создания каталога экземпляра его положение нельзя изменить. Этот каталог содержит:

- Файл конфигурации менеджера баз данных
- Системный каталог баз данных
- Каталог узлов
- Файл конфигурации узлов (`db2nodes.cfg`)

- Все другие файлы, содержащие отладочную информацию (например, дамп исключительных ситуаций и регистров или стек вызовов для процесса DB2®).

В операционных системах UNIX® каталог экземпляра расположен в каталоге `INSTHOME/sqllib`, где `INSTHOME` - начальный каталог владельца экземпляра.

В операционных системах Windows® каталог экземпляра расположен в подкаталоге `/sqllib` каталога, в котором установлен DB2.

В системе многораздельных баз данных каталог экземпляра совместно используется всеми серверами разделов баз данных, входящими в этот экземпляр. Поэтому каталог экземпляра должен быть создан на совместно используемом сетевом диске, к которому могут обращаться все компьютеры этого экземпляра.

В процессе установки создается исходный экземпляр DB2 с именем “DB2”. В системе UNIX исходному экземпляру можно дать любое имя, удовлетворяющее правилам именования. Это имя экземпляра используется для задания структуры каталогов.

Чтобы можно было сразу использовать этот экземпляр, в процессе установки задаются следующие параметры:

- Переменной среды DB2INSTANCE присваивается значение “DB2”.
- Переменной реестра DB2 DB2INSTDEF присваивается значение “DB2”.

В системе UNIX в качестве значения этих переменных можно использовать любое имя, удовлетворяющее правилам именования.

В Windows имя экземпляра совпадает с именем службы, поэтому конфликтов возникнуть не должно. Для создания службы вам потребуются специальные права доступа.

Эти параметры определяют экземпляр с именем “DB2” как экземпляр по умолчанию. Используемый по умолчанию экземпляр можно изменить, но сначала нужно создать дополнительный экземпляр.

Перед использованием DB2 необходимо изменить среду для каждого пользователя, чтобы он мог обращаться к экземпляру и выполнять программы DB2. Это относится ко всем пользователям (включая администраторов).

Для операционных систем UNIX поставляются примеры файлов сценариев, помогающие настроить среду базы данных. Это файлы `db2profile` для оболочек Bourne или Korn и `db2cshrc` для оболочки C. Эти сценарии находятся в подкаталоге `sqllib` начального каталога владельца экземпляра. Владелец экземпляра или любой пользователь из группы SYSADM этого экземпляра

может настроить этот сценарий для всех пользователей экземпляра. Можно также скопировать и настроить этот сценарий для каждого пользователя.

Пример сценария содержит операторы, которые:

- Изменяют значение PATH для пользователя, добавляя к существующим путям поиска следующие каталоги: подкаталоги `bin`, `adm` и `misc` в подкаталоге `sqllib` начального каталога владельца экземпляра.
- Присваивают переменной среды DB2INSTANCE имя этого экземпляра.

Понятия, связанные с данным:

- “Несколько экземпляров в операционной системе UNIX” на стр. 23
- “Несколько экземпляров в операционной системе Windows” на стр. 24

Задачи, связанные с данной темой:

- “Добавление экземпляра” на стр. 28
- “Подробности создания экземпляров в UNIX” на стр. 26
- “Подробности создания экземпляров в Windows” на стр. 27
- “Выбор текущего экземпляра” на стр. 29
- “Автоматический запуск экземпляров” на стр. 30
- “Одновременная работа нескольких экземпляров” на стр. 30
- “Просмотр списка экземпляров” на стр. 29
- “Создание дополнительных экземпляров” на стр. 25

Автоматическая настройка среды DB2 в UNIX

По умолчанию сценарии настройки базы данных при создании экземпляра влияют на пользовательскую среду только во время текущего сеанса. При использовании оболочек Bourne или Korn можно изменить файл `.profile`, чтобы задать в нем автоматическое выполнение сценария `db2profile` во время регистрации пользователя. Для пользователей, использующих оболочку C, можно изменить файл `.login` , чтобы он запускал файл сценария `db2shrc`.

Процедура:

Добавьте в файл сценария `.profile` или `.login` один из следующих операторов:

- Для пользователей, совместно использующих одну версию сценария, добавьте:

```
. INSTHOME/sqllib/db2profile    (для оболочек Bourne или Korn)
source INSTHOME/sqllib/db2cshrc (для оболочки C)
```

  где `INSTHOME` - начальный каталог нужного экземпляра.
- Для пользователей, в начальных каталогах которых есть настроенные для них версии сценария, добавьте:

```
. USERHOME/db2profile         (для оболочек Bourne или Korn)
source USERHOME/db2cshrc      (в оболочке C)
```

где USERHOME - начальный каталог этого пользователя.

Задачи, связанные с данной темой:

- “Настройка среды DB2 в UNIX вручную” на стр. 22

Настройка среды DB2 в UNIX вручную

Процедура:

Чтобы выбрать экземпляр, который нужно использовать, введите в командной строке один из следующих операторов. Точка (.) и пробел обязательны.

- Для пользователей, совместно использующих одну версию сценария, добавьте:

```
. INSTHOME/sqllib/db2profile    (для оболочек Bourne или Korn)
source INSTHOME/sqllib/db2cshrc (для оболочки C)
```

  где INSTHOME - начальный каталог нужного экземпляра.

- Для пользователей, в начальных каталогах которых есть настроенные для них версии сценария, добавьте:

```
. USERHOME/db2profile         (для оболочек Bourne или Korn)
source USERHOME/db2cshrc      (в оболочке C)
```

  где USERHOME - начальный каталог этого пользователя.

Если нужно работать с несколькими экземплярами одновременно, выполните этот сценарий для каждого такого экземпляра в отдельном окне. Например, предположим, что есть два экземпляра с именами test и prod, а их начальные каталоги - /u/test и /u/prod.

В окне 1:

- В оболочке Bourne или Korn введите:

```
. /u/test/sqllib/db2profile
```

- В оболочке C введите:

```
source /u/test/sqllib/db2cshrc
```

В окне 2:

- В оболочке Bourne или Korn введите:

```
. /u/prod/sqllib/db2profile
```

- В оболочке C введите:

```
source /u/prod/sqllib/db2cshrc
```

Используйте окно 1 для работы с экземпляром `test`, а окно 2 - для работы с экземпляром `prod`.

Примечание: Чтобы проверить, правильно ли задан путь поиска, введите команду **which db2**. Эта команда возвращает абсолютный путь исполняемого файла DB2 CLP. Проверьте, есть ли этот файл в каталоге `sqllib` этого экземпляра.

Задачи, связанные с данной темой:

- “Автоматическая настройка среды DB2 в UNIX” на стр. 21

Несколько экземпляров в операционной системе UNIX

В одной системе UNIX® может быть несколько экземпляров. Однако одновременно можно работать только с одним экземпляром DB2®.

Примечание: Для исключения конфликтов между экземплярами убедитесь, что в каждом экземпляре применяется собственная домашняя файловая система. При использовании общей файловой системы могут возникнуть ошибки.

С каждым экземпляром связаны владелец экземпляра и группа администраторов системы (SYSADM). Владелец экземпляра и группа SYSADM назначаются при создании экземпляра. Один ID пользователя или имя пользователя может использоваться только для одного экземпляра. Пользователь с этим ID или именем пользователя называется также *владельцем экземпляра*.

У каждого владельца экземпляра должен быть свой уникальный начальный каталог. Все файлы, необходимые для запуска экземпляра, создаются в начальном каталоге этого ID пользователя или имени пользователя владельца экземпляра.

Если возникает необходимость удалить из системы этот ID пользователя или имя пользователя владельца экземпляра, можно потерять файлы, связанные с этим экземпляром, и утратить доступ к данным, сохраненным в этом экземпляре базы данных. Поэтому рекомендуется использовать ID пользователя или имя пользователя владельца экземпляра только для работы с DB2.

Также важна первичная группа владельца экземпляра. Эта первичная группа автоматически становится группой администраторов для этого экземпляра и получает полномочия SYSADM для этого экземпляра. Другие ID пользователей или имена пользователей, входящие в первичную группу экземпляра, также получают этот уровень полномочий. Поэтому можно назначить ID пользователя или имя пользователя владельца экземпляра первичной группе, которая зарезервирована для управления экземплярами. (Убедитесь также, что

для ID пользователя или имени пользователя владельца экземпляра назначена первичная группа; в противном случае будет использоваться первичная группа системы по умолчанию.)

Если уже есть группа, которую нужно сделать группой администраторов системы для этого экземпляра, можно просто назначить эту группу в качестве первичной группы при создании ID пользователя или имени пользователя владельца экземпляра. Чтобы предоставить полномочия администратора другим пользователям, добавьте их к этой группе, назначенной в качестве группы администраторов системы.

Чтобы получить независимые полномочия SYSADM для разных экземпляров, каждый ID пользователя или имя пользователя владельца экземпляра должны использовать разные первичные группы. Однако если вы решили использовать общие полномочия SYSADM для нескольких экземпляров, можно использовать для них одну первичную группу.

Задачи, связанные с данной темой:

- “Подробности создания экземпляров в UNIX” на стр. 26

Несколько экземпляров в операционной системе Windows

В одной системе может работать несколько экземпляров DB2®. С каждым экземпляром DB2 связаны собственные базы данных и собственные параметры конфигурации менеджера баз данных.

Компоненты экземпляра DB2:

- Служба Windows®, соответствующая экземпляру. Имя службы совпадает с именем экземпляра. В панели служб перед этим именем будет добавлена строка “DB2 - ”. Например, для экземпляра с именем DB2 существует служба Windows с именем “DB2”, которое будет показано в панели служб как “DB2 - DB2”.

  Примечание: Служба Windows не создается в Windows 98, Windows ME и для экземплярах клиентов.

- Каталог экземпляра. Каталог содержит файлы конфигурации менеджера баз данных, системный каталог базы данных, каталог узла, каталог базы данных DCS и все файлы журналов диагностики и дампа, связанные с интерфейсом. Каталог экземпляра по умолчанию - это подкаталог каталога SQLLIB с тем же именем, что и экземпляр. Например, каталог экземпляра “DB2”: `C:\SQLLIB\DB2`, где `C:\SQLLIB` - каталог установки DB2. Каталог экземпляра по умолчанию можно изменить с помощью переменной реестра DB2INSTPROF. Если переменная реестра DB2INSTPROF имеет другое значение, подкаталог экземпляра будет создан в каталоге, на который указывает эта переменная. Например, если `DB2INSTPROF=D:\DB2PROFS`, каталогом экземпляра будет `D:\DB2PROFS\DB2`.

- Ключ реестра находится в разделе HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\PROFILES\<имя-экземпляра>. В этом разделе находятся все переменные реестра, связанные с экземпляром.

Несколько экземпляров DB2 могут работать одновременно. Для работы с экземпляром необходимо указать в переменной среды DB2INSTANCE имя экземпляра, к которому будут относиться команды.

Для предотвращения обращения одного экземпляра к базе данных другого файлы баз данных каждого экземпляра создаются в каталоге с тем же именем, что и имя экземпляра. Например, при создании базы данных на устройстве `C:` для экземпляра с именем DB2 файлы базы данных будут созданы в каталоге `C:\DB2`. Аналогично, при создании базы данных на устройстве `C:` для экземпляра TEST файлы базы данных будут созданы в каталоге `C:\TEST`.

Понятия, связанные с данным:

- "High Availability" в *Справочное руководство по восстановлению данных и высокой доступности*

Задачи, связанные с данной темой:

- "Подробности создания экземпляров в Windows" на стр. 27

Создание дополнительных экземпляров

Один экземпляр создается при установке DB2 UDB, но может потребоваться создание дополнительных экземпляров.

Предварительные требования:

Если у вас есть полномочия root в системе UNIX, или вы входите в группу администраторов в Windows NT, вы можете добавить дополнительные экземпляры DB2. Компьютер, на котором добавляется экземпляр, становится компьютером - владельцем экземпляра (нулевым узлом). Убедитесь в том, что добавление экземпляров происходит на сервере администратора DB2.

Процедура:

Чтобы добавить экземпляр из командной строки, введите команду:

```
db2icrt <имя_экземпляра>
```

Если для добавления еще одного экземпляра DB2 используется команда **db2icrt**, нужно задать имя регистрации владельца экземпляра; можно также задать тип аутентификации для этого экземпляра. Заданный тип аутентификации применяется для всех баз данных, созданных в этом экземпляре. Тип аутентификации определяет, где будет выполняться аутентификация пользователей.

Можно изменить положение каталога экземпляра в DB2PATH, используя переменную среды DB2INSTPROF. Необходимо обладать привилегиями записи для этого каталога экземпляра. Если нужно, чтобы каталоги создавались не в пути DB2PATH, необходимо задать переменную среды DB2INSTPROF *ДО* ввода команды **db2icrt**.

При работе с DB2 Universal Database Enterprise - Server Edition нужно также объявить, что добавляемый новый экземпляр является системой многораздельных баз данных.

Понятия, связанные с данным:

- “Способы аутентификации, поддерживаемые сервером” на стр. 237
- “Особенности аутентификации удаленных клиентов” на стр. 242

Ссылки, связанные с данной темой:

- “db2icrt - Create Instance Command” в *Command Reference*

Подробности создания экземпляров в UNIX

При работе в операционных системах UNIX у команды **db2icrt** можно задать следующие дополнительные параметры:

- –h или –?

  Этот параметр используется для вывода на экран меню справки для этой команды.

- –d

  Этот параметр задает режим отладки, используемый для поиска причин ошибок.

- –a тип_аутентификации

  Этот параметр задает тип аутентификации для этого экземпляра. Допустимые типы аутентификации: SERVER, SERVER_ENCRYPT и CLIENT. Если этот параметр не задан, при установке сервера DB2 по умолчанию задается тип аутентификации SERVER. В противном случае задается тип аутентификации CLIENT.

  Примечания:

  1. Тип аутентификации экземпляра применяется для всех баз данных этого экземпляра.
  2. В операционных системах UNIX нельзя задавать тип аутентификации DCE.

- –u FencedID

  Этот параметр задает пользователя, под именем которого будут выполняться изолированные пользовательские функции (UDF) и хранимые процедуры. Это

необязательный параметр при установке клиента DB2 или клиента разработки прикладных программ DB2. Для других продуктов DB2 это обязательный параметр.

Примечание: В качестве FencedID нельзя использовать “root” или “bin”.

- –p имя_порта

  Этот параметр задает имя используемой службы или номер порта TCP/IP. Это значение будет затем задано в файлах конфигурации всех баз данных экземпляра.

- –s тип_установки

  Позволяет создавать экземпляры разных типов. Допустимы следующие типы: `ese`, `wse`, `client` и `standalone`.

Примеры:

- Чтобы добавить экземпляр сервера DB2, можно использовать команду:

  ```
  db2icrt -u db2fenc1 db2inst1
  ```

- Если установлен только DB2 Connect Enterprise Edition, в качестве FencedID можно использовать имя экземпляра:

  ```
  db2icrt -u db2inst1 db2inst1
  ```

- Чтобы добавить экземпляр клиента DB2, можно использовать команду:

  ```
  db2icrt db2inst1 -s client -u fencedID
  ```

Экземпляры клиентов DB2 создаются, когда нужно, чтобы данная рабочая станция соединялась с другими серверами баз данных, и не требуется локальная база данных на этой рабочей станции.

Ссылки, связанные с данной темой:

- “db2icrt - Create Instance Command” в *Command Reference*

Подробности создания экземпляров в Windows

При работе в операционной системе Windows для команды **db2icrt** можно задать следующие дополнительные параметры:

- –s тип_установки

  Позволяет создавать экземпляры разных типов. Допустимы следующие типы: `ese`, `wse`, `client` и `standalone`.

- –p:путь_профиля

  Этот необязательный параметр задает другой путь для профиля экземпляра. Если этот путь не задан, каталог экземпляра создается в каталоге `SQLLIB`, а его имя совместного использования получается добавлением к `DB2` имени экземпляра. Все пользователи в домене автоматически получают права на чтение и запись для этого каталога. Эти права можно изменить, чтобы ограничить доступ в этот каталог.

Если задан другой путь профиля экземпляра, необходимо создать совместно используемый диск или каталог. Это даст всем пользователям в домене доступ к каталогу экземпляра, пока разрешения не будут изменены.

- –u:имя_пользователя,пароль

  При создании среды многораздельной базы данных нужно задать домен, регистрационное имя, учетную запись и пароль для этой службы DB2.

- –r:начальный_порт,конечный_порт

  Это необязательный параметр, задающий диапазон портов TCP/IP для менеджера FCM (Fast Communications Manager). Задавая диапазон портов TCP/IP, убедитесь, что этот диапазон портов доступен на всех компьютерах этой системы многораздельных баз данных.

Пример этой команды для DB2 for Windows Enterprise Server Edition:

```
db2icrt inst1 -s eee
   -p:\\machineA\db2mpp      -u:<имя пользователя>,<пароль> -r:9010,9015
```

Примечание: Если вы изменили имя пользователя службы, то есть больше не используете службу по умолчанию, созданную при создании первого экземпляра во время установки продукта, то предоставьте новому домену/пользователю следующие права доступа:

- Действовать как часть операционной системы
- Создавать объекты маркеров
- Увеличивать квоту
- Регистрироваться как служба
- Заменять маркер уровня процесса
- Блокировать страницы в памяти

Эти права требуются экземпляру для доступа к совместно используемому диску, аутентификации учетной записи пользователя и запуска DB2 в виде службы Windows. Права доступа ″Блокировать страницы в памяти″ нужны для поддержки Address Windowing Extensions (AWE).

Ссылки, связанные с данной темой:

- “db2icrt - Create Instance Command” в *Command Reference*

Добавление экземпляра

Процедура:

Создав дополнительный экземпляр, добавьте запись для него в Центр управления, чтобы с этим экземпляром можно было работать из Центра управления.

Чтобы добавить еще один экземпляр, выполните следующие шаги:

1. Зарегистрируйтесь под ID или именем пользователя, обладающим полномочиями администратора или входящим в группу локальных администраторов.
2. Чтобы добавить экземпляр, используйте один из следующих методов:

   При помощи Центра управления:

a. Раскройте дерево объектов и найдите папку **Экземпляры** нужной системы.
b. Щелкните правой кнопкой мыши по папке экземпляров и выберите из всплывающего меню пункт **Добавить**.
c. Введите необходимую информацию и нажмите кнопку **Применить**.

Просмотр списка экземпляров

Процедура:

Чтобы получить список всех доступных в системе экземпляров с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Экземпляры**.
2. Щелкните правой кнопкой мыши по папке Экземпляры и выберите из всплывающего меню пункт **Добавить**.
3. В окне Добавить экземпляр нажмите кнопку **Обновить**.
4. Щелкните по стрелке вниз, чтобы увидеть список экземпляров баз данных.
5. Нажмите кнопку **Отмена**, чтобы закрыть это окно.

Чтобы получить список всех доступных в системе экземпляров из командной строки, введите команду:

```
db2ilist
```

Для того чтобы узнать, какой экземпляр используется в текущем сеансе (на поддерживаемой платформе Windows), вызовите команду:

```
set db2instance
```

Ссылки, связанные с данной темой:

- “db2ilist - List Instances Command” в *Command Reference*

Выбор текущего экземпляра

Процедура:

Команды запуска или остановки менеджера баз данных экземпляра система DB2 выполняет для текущего экземпляра. DB2 определяет текущий экземпляр так:

- Если для текущего сеанса задана переменная среды DB2INSTANCE, текущий экземпляр определяется ее значением. Чтобы задать переменную среды DB2INSTANCE, введите команду:

  ```
  set db2instance=<новое_имя_экземпляра>
  ```

- Если переменная среды DB2INSTANCE не задана для текущего сеанса, DB2 использует значение переменной среды DB2INSTANCE из переменных среды системы. В Windows NT переменные среды системы задаются в среде системы. В Windows 9x они задаются в файле `autoexec.bat`.
- Если переменная среды DB2INSTANCE вообще не задана, DB2 использует значение переменной реестра DB2INSTDEF.

  Чтобы задать переменную реестра DB2INSTDEF на глобальном уровне реестра, введите команду:

  ```
  db2set db2instdef=<новое_имя_экземпляра> -g
  ```

Чтобы узнать, какой экземпляр используется в текущем сеансе, введите команду:

```
db2 get instance
```

Задачи, связанные с данной темой:

- “Объявление переменных реестра и среды” на стр. 33

Автоматический запуск экземпляров

Процедура:

В операционных системах Windows экземпляр DB2, создаваемый при установке, запускается автоматически по умолчанию. Экземпляр, созданный командой **db2icrt**, необходимо запускать вручную. Для изменения типа запуска экземпляра откройте панель Службы и измените на ней свойства службы DB2.

Чтобы в системе UNIX разрешить автоматический запуск экземпляра после каждого перезапуска системы, введите следующую команду:

```
db2iauto -on <имя экземпляра>
```

где `<имя экземпляра>` - имя, под которым зарегистрирован экземпляр

Чтобы запретить в системе UNIX автоматический запуск экземпляра после каждого перезапуска системы, введите следующую команду:

```
db2iauto -off <имя экземпляра>
```

где `<имя экземпляра>` - имя, под которым зарегистрирован экземпляр

Одновременная работа нескольких экземпляров

Процедура:

Чтобы одновременно запустить несколько экземпляров с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Базы данных**.
2. Щелкните правой кнопкой мыши по одному из экземпляров и выберите из всплывающего меню пункт **Запустить**.
3. Повторите шаг 2 для всех экземпляров, которые должны выполняться одновременно.

(Только в Windows:) Чтобы одновременно запустить несколько экземпляров из командной строки:

1. Задайте в качестве значения переменной DB2INSTANCE имя еще одного экземпляра, который нужно запустить, используя команду:

   ```
   set db2instance=<имя другого_экземпляра>
   ```

2. Запустите этот экземпляр, введя команду **db2start**.

Управление лицензиями

Для управления лицензиями на продукты DB2® используется в первую очередь Центр лицензий, входящий в Центр управления пользовательского интерфейса продукта. В Центре лицензий можно узнать информацию лицензии, статистическую информацию и информацию о зарегистрированных и текущих пользователях для каждого установленного продукта.

Если Центр управления недоступен, то основные функции работы с лицензиями можно выполнить с помощью команды **db2licm** Средства управления лицензиями. Эта команда позволяет добавлять, удалять, просматривать и изменять лицензии и правила, установленные в локальной системе.

Ссылки, связанные с данной темой:

- “db2licm - License Management Tool Command” в *Command Reference*

Переменные среды и реестра профиля

Переменные среды и переменные реестра определяют среду базы данных.

Для настройки параметров конфигурации и переменных реестра можно воспользоваться Помощником по настройке **db2ca**).

Если перед началом работы с реестром профиля DB2DB2® на рабочей станции (например, с системой WindowsDB2®) требуется изменить переменные среды, выполните это изменение и перезагрузите систему. Теперь ваша среда (за немногими исключениями) контролируется переменными реестра, хранящимися в реестрах профилей DB2. Пользователи UNIXDB2® с правами системного администратора (SYSADM) для данного экземпляра могут изменять значения

реестра для этого экземпляра. Пользователям Windows не нужны права SYSADM для изменения значений реестра. Чтобы изменить значения переменных реестра без перезагрузки, используйте команду **db2set**; заданные значения сразу сохраняются в реестрах профилей.Измененная информация в реестре будет действительна для экземпляров сервера DB2 и прикладных программ DB2, запущенных после внесения изменений.

При изменении реестра новые значения не применяются для активных в настоящее время прикладных программ или пользователей DB2. Для прикладных программ, запущенных после изменения реестра, используются новые значения.

Примечание: Переменные среды DB2 `DB2INSTANCE` и `DB2NODE` невозможно сохранить в реестре профилей DB2. В некоторых операционных системах для изменения переменных среды необходимо вызвать команду **set**. Эти изменения вступят в силу только после следующей перезагрузки системы. В системе UNIX вместо команды **set** можно использовать команду **export**.

Используя реестр профиля, можно централизованно управлять переменными среды. Различные уровни поддержки обеспечиваются в настоящее время разными профилями. При использовании сервера администратора DB2 возможно также удаленное управление переменными среды.

Существует четыре реестра профилей:
- Реестр профиля DB2 уровня экземпляра. В этом реестре размещается большинство переменных среды DB2. В нем хранятся значения переменных среды для конкретного экземпляра. Значения, определенные на этом уровне, переопределяют соответствующие значения на глобальном уровне.
- Реестр профиля DB2 глобального уровня. Если переменная среды не задана для конкретного экземпляра, используется этот реестр. Он содержит значения переменных среды, заданные для всего компьютера. В DB2 UDB ESE на каждом компьютере существует свой профиль глобального уровня.
- Реестр профиля DB2 уровня узла экземпляра. На этом уровне реестра хранятся переменные, применяемые на конкретном разделе (узле) многоразрядной среды. Значения из этого уровня переопределяют значения, заданные на уровне экземпляра или глобальном уровне.
- Реестр профиля экземпляра DB2. Этот реестр содержит список всех имен экземпляров, распознаваемых системой. Полный список экземпляров в системе можно получить с помощью команды `db2ilist`.

DB2 конфигурирует рабочую среду, проверяя значения реестра и переменные среды в следующем порядке:
1. Переменные среды, заданные при помощи команды `set`. (Или команды `export` в системах UNIX.)

2. Переменные среды, заданные в профиле уровня экземпляра (с помощью команды **db2set -i <экземпляр> <узел>**).
3. Значения реестра, заданные в профиле уровня экземпляра (при помощи команды **db2set -i**).
4. Значения реестра, заданные в профиле глобального уровня (при помощи команды **db2set -g**).

Понятия, связанные с данным:

- "Переменные реестра и среды DB2" в *Руководство администратора: Производительность*

Задачи, связанные с данной темой:

- "Объявление переменных реестра и среды" на стр. 33

Объявление переменных реестра и среды

Процедура:

Команда **db2set** позволяет задать значения локальных переменных реестра (и переменных среды).

Чтобы получить информацию справки для этой команды, используйте команду:

```
db2set ?
```

Чтобы получить полный список всех поддерживаемых переменных реестра, используйте команду:

```
db2set -lr
```

Чтобы получить список всех определенных переменных реестра для текущего экземпляра или экземпляра по умолчанию, используйте команду:

```
db2set
```

Чтобы получить список всех определенных переменных реестра в реестре профиля, используйте команду:

```
db2set -all
```

Чтобы узнать значение переменной реестра для текущего экземпляра или экземпляра по умолчанию, используйте команду:

```
db2set имя_переменной_реестра
```

Чтобы узнать значение переменной реестра на всех уровнях, используйте команду:

```
db2set имя_переменной_реестра -all
```

Чтобы изменить значение переменной реестра для текущего экземпляра или экземпляра по умолчанию, используйте команду:

```
db2set имя_переменной_реестра=новое_значение
```

Чтобы изменить значение по умолчанию переменной реестра для всех баз данных этого экземпляра, используйте команду:

```
db2set имя_переменной_реестра=новое_значение       -i имя_экземпляра
```

Чтобы изменить значение по умолчанию переменной реестра для конкретного раздела экземпляра, используйте команду:

```
db2set имя_переменной_реестра=новое_значение
   -i имя_экземпляра номер_узла
```

Чтобы изменить значение по умолчанию переменной реестра для всех экземпляров в системе, используйте команду:

```
db2set имя_переменной_реестра=новое_значение -g
```

При работе с протоколом LDAP вы можете задать переменные реестра в LDAP следующим образом:

- Для задания переменных реестра на уровне пользователя с помощью LDAP используйте команду:

  ```
  db2set -ul
  ```

- Для задания переменных реестра на глобальном уровне с помощью LDAP используйте команду:

  ```
  db2set -gl имя_пользователя
  ```

При работе в среде LDAP можно задать значение переменной реестра DB2 в LDAP, поскольку его область видимости - все компьютеры и всех пользователи, входящие в раздел каталога или в домен Windows NT. В настоящее время на глобальном уровне LDAP можно задать только переменную реестра DB2 DB2LDAP_SEARCH_SCOPE.

Для задания значения области поиска на глобальном уровне в LDAP, используйте команду:

```
db2set -gl db2ldap_search_scope = значение
```

где *значение* - “local”, “domain” или “global”.

Примечания:

1. Опция `-gl`, работающая на глобальном уровне LDAP, отличается от опции `-g`, используемой для задания переменных реестра DB2 на глобальном уровне компьютера.
2. Переменная реестра на уровне пользователя поддерживается LDAP только в среде Windows.

3. Значения переменных на уровне пользователя содержат значения, уникальные для каждого пользователя. Все изменения значений на уровне пользователя заносятся в каталог LDAP.
4. Параметры “-i”, “-g”, “-gl” и “-ul” нельзя использовать в одной команде.
5. Некоторые переменные всегда определяются в профиле глобального уровня. Они не могут быть заданы в профилях на уровне узла или экземпляра (например, переменные DB2SYSTEM and DB2INSTDEF).
6. Чтобы изменить значения реестра для экземпляра в системе UNIX, необходимо обладать полномочиями администратора системы (SYSADM). Только пользователи с полномочиями основного пользователя (root) могут изменять значения в реестрах глобального уровня.

Чтобы вернуть значение переменной реестра для экземпляра к ее значению по умолчанию из реестра глобального профиля, используйте команду:

```
db2set -r имя_переменной_реестра
```

Чтобы вернуть значение переменной реестра для узла в экземпляре к ее значению по умолчанию из реестра глобального профиля, используйте команду:

```
db2set -r имя_переменной_реестра номер_узла
```

Чтобы удалить значение переменной на определенном уровне, можно использовать тот же синтаксис, что и для задания значения переменной, но не задавать значение переменной. Например, чтобы удалить значение переменной на уровне узла, введите команду:

```
db2set имя_переменной_реестра= -i имя_экземпляра
номер_узла
```

Чтобы удалить значение переменной и ограничить ее использование, если она определена в профиле более высоком уровне, введите команду:

```
db2set имя_переменной_реестра= -null имя_экземпляра
```

Эта команда удалит значение указанной переменной и запретит использовать для изменения ее значения профили более высокого уровня (в данном случае профиль глобального уровня DB2). Однако для задания значения этой переменной при этом может использоваться профиль более низкого уровня (в данном случае профиль уровня узла DB2).

Понятия, связанные с данным:

- “Переменные реестра и среды DB2” в *Руководство администратора: Производительность*

Задачи, связанные с данной темой:

- “Задание переменных среды в Windows” на стр. 36
- “Задание переменных среды в системах UNIX” на стр. 38

- “Поиск в разделах каталога LDAP или доменах” на стр. 345
- “Настройка переменных реестра DB2 на уровне пользователя в среде LDAP” на стр. 347

Задание переменных среды в Windows

Процедура:

Все переменные реестра, относящиеся к DB2, настоятельно рекомендуется определять в реестре профиля DB2. Если переменные DB2 заданы вне реестра, будет невозможно удаленное управление этими переменными и для вступления новых значений переменных в силу необходима будет перезагрузка рабочей станции.

В операционных системах Windows есть одна системная переменная среды, DB2INSTANCE, которую можно задать только вне реестра профиля, однако задавать ее значение не обязательно. В профиле глобального уровня можно задать переменную реестра DB2 DB2INSTDEF, определяющую имя экземпляра, если не задана переменная среды DB2INSTANCE.

Для серверов DB2 Enterprise Server Edition в Windows есть две переменные среды системы, DB2INSTANCE и DB2NODE, которые можно задать только вне реестра профиля. Переменную среды DB2INSTANCE задавать не обязательно. В профиле глобального уровня можно задать переменную реестра DB2 DB2INSTDEF, определяющую имя экземпляра, если не задана переменная среды DB2INSTANCE.

Переменная среды DB2NODE используется для маршрутизации запросов на логический узел назначения в компьютере. Эта переменная среды должна быть задана в сеансе, в котором выполняются прикладная программа или команда, а не в реестре профиля DB2. Если эта переменная не задана, по умолчанию используется логический узел назначения, который определен с нулевым (0) портом на данном компьютере.

Чтобы узнать значение переменной среды, используйте команду **echo**. Например, чтобы узнать значение переменной среды DB2PATH, введите команду:

```
echo %db2path%
```

Чтобы задать переменные среды:

В системах Windows 9x: Отредактируйте файл `autoexec.bat` и перезагрузите систему, чтобы внесенные изменения вступили в силу.

В Windows: Переменные среды DB2 DB2INSTANCE и DB2NODE можно задать так (в этом описании - для DB2INSTANCE):

- (В Windows NT и Windows 2000) Выберите **Пуск**, **Настройка**, **Панель управления**. (В Windows XP и Windows .NET) Выберите **Пуск** —> **Панель управления**.
- (В Windows NT и Windows 2000) Дважды щелкните на значке **Система**. (В Windows XP и Windows .NET) В зависимости от выбранной темы Windows и типа просмотра, вам может потребоваться выбрать **Производительность** и **Обслуживание** перед выбором значка **Система**.
- (В Windows NT) В Панели управления выберите Система -> Переменные среды и выполните следующие действия: (В Windows 2000, Windows XP и Windows .NET) В окне Параметры системы выберите вкладку **Дополнительные**, нажмите кнопку Переменные среды и выполните следующие действия:
  1. Если переменная DB2INSTANCE не существует:
     a. (В Windows NT) Выберите любую переменную среды системы. (В Windows 2000, Windows XP и Windows .NET) Нажмите кнопку **Создать**.
     b. (В Windows NT) Измените имя в поле *Переменная* на DB2INSTANCE. (В Windows 2000, Windows XP и Windows .NET) Введите в поле *Имя переменной* значение DB2INSTANCE.
     c. (В Windows NT) Измените значение в поле *Значение* на имя экземпляра, например db2inst. (В Windows 2000, Windows XP и Windows .NET) Введите в поле *Значение переменной* имя экземпляра, например db2inst.
  2. Если переменная DB2INSTANCE уже существует, задайте новое значение:
     a. Выберите переменную среды DB2INSTANCE.
     b. Измените значение в поле *Значение* на имя экземпляра, например db2inst.
  3. (В Windows NT) Нажмите кнопку Задать. (В Windows 2000, Windows XP и Windows .NET) Нажмите OK.
  4. Нажмите кнопку OK.
  5. Перезагрузите систему, чтобы эти изменения вступили в силу.

Примечание: Переменную среды DB2INSTANCE можно также задать на уровне сеанса (процесса). Например, если нужно запустить второй экземпляр DB2 с именем TEST, введите в окне команд следующие команды:

```
set DB2INSTANCE=TEST
db2start
```

В оболочке C введите в командной строке следующие команды:

```
setenv DB2INSTANCE TEST
```

Реестры профилей располагаются в следующих местах:

- Реестр профиля DB2 уровня экземпляра - в реестре операционной системы Windows с путем:

  `\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\PROFILES\`*`имя_экземпляра`*

Примечание: *имя_экземпляра* - имя экземпляра DB2.

- Реестр профиля DB2 глобального уровня - в реестре Windows с путем:

  `\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\GLOBAL_PROFILE`

- Реестр профиля DB2 уровня узла экземпляра - в реестре Windows с путем:

  `...\SOFTWARE\IBM\DB2\PROFILES\`*`имя_экземпляра`*`\NODES\`*`номер_узла`*

  Примечание: *имя_экземпляра* и *номер_узла* - конкретные значения для используемого раздела базы данных.

- Реестр профиля экземпляра DB2 не требуется. Для каждого экземпляра DB2 в системе создается следующий ключ:

  `\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\PROFILES\`*`имя_экземпляра`*

  Список экземпляров можно получить, просмотрев ключи, расположенные под ключом PROFILES.

Понятия, связанные с данным:

- “Сервер администратора DB2” на стр. 45

Задачи, связанные с данной темой:

- “Задание переменных среды в системах UNIX” на стр. 38

Задание переменных среды в системах UNIX

Процедура:

Все переменные реестра, относящиеся к DB2, настоятельно рекомендуется определять в реестре профиля DB2. Если переменные DB2 заданы вне реестра, будет невозможно удаленное управление этими переменными.

В операционных системах UNIX необходимо задать переменную среды системы DB2INSTANCE.

Помогут вам настроить среду базы данных примеры сценариев `db2profile` (для оболочки Korn) и `db2cshrc` (для оболочек Bourne или C). Эти файлы можно найти в каталоге `insthome/sqllib`, где `insthome` - начальный каталог владельца экземпляра.

Эти сценарии содержат операторы, которые:

- Добавляют в переменную среды PATH пользователя следующие каталоги:
  - `insthome/sqllib/bin`
  - `insthome/sqllib/adm`
  - `insthome/sqllib/misc`

- Задают значение переменной среды DB2INSTANCE - используемое локальное имя_экземпляра по умолчанию.

Примечание: Все переменные, поддерживаемые DB2, кроме PATH и DB2INSTANCE, должны быть заданы в реестре профиля DB2. Чтобы задать переменные, которые не поддерживаются DB2, определите их в используемых файлах сценариев userprofile и usercshrc.

Владелец экземпляра или пользователь SYSADM может настроить эти сценарии для всех пользователей экземпляра. Можно также скопировать и настроить сценарий для каждого пользователя и затем вызывать его явно или добавить его вызов в файлы .profile или .login.

Чтобы изменить эту переменную среды для текущего сеанса, используйте команды, подобные приведенным ниже:

- Для оболочки Korn:

```
DB2INSTANCE=inst1
export DB2INSTANCE
```

- Для оболочки Bourne:

```
export DB2INSTANCE=<inst1>
```

- Для оболочки C:

```
setenv DB2INSTANCE <inst1>
```

Для правильного управления реестром профиля DB2 в операционных системах UNIX необходимо следовать следующим правилам принадлежности файлов.

- Файл реестра профиля DB2 уровня экземпляра:

  ```
  INSTHOME/sqllib/profile.env
  ```

  Разрешения на доступ и владелец для этого файла должны быть:

  ```
  -rw-rw-r-- <db2inst1> <db2iadm1> profile.env
  ```

  где <db2inst1> - владелец экземпляра, а <db2iadm1> - группа владельца.

  INSTHOME - начальный каталог владельца экземпляра.

- Реестр профиля DB2 глобального уровня:
  - /var/db2/<id_версии>/default.env в операционных системах AIX, Solaris и Linux (где <id-версии> текущая версия).
  - /var/opt/db2/<ID_версии>/default.env для операционных систем HP-UX (где <ID_версии> - текущая версия).

  Разрешения на доступ и владелец для этого файла должны быть:

  ```
  -rw-rw-r-- <Владелец_экземпляра> <Группа_владельца> default.env
  ```

Чтобы изменить переменные реестра глобального уровня, пользователь должен зарегистрироваться как root.

- Реестр профиля DB2 уровня узла экземпляра:

  ```
  INSTHOME/sqllib/nodes/<номер_узла>.env
  ```

  Разрешения на доступ и владелец для этого каталога и файла должны быть:

  ```
  drwxrwsr-w <Владелец_экземпляра> <Группа_владельца> nodes

  -rw-rw-r-- <Владелец_экземпляра> <Группа_владельца> <номер_узла>.env
  ```

  INSTHOME - начальный каталог владельца экземпляра.

- Реестр профиля экземпляров DB2:
  - /var/db2/<id_версии>/profiles.reg в операционных системах AIX, Solaris и Linux (где <id_версии> - текущая версия).
  - /var/opt/db2/<ID_версии>/profiles.reg для операционных систем HP-UX (где <ID_версии> - текущая версия).

  Разрешения на доступ и владелец для этого файла должны быть:

  ```
  -rw-r--r-- root system profiles.reg
  ```

Понятия, связанные с данным:

- “Сервер администратора DB2” на стр. 45

Задачи, связанные с данной темой:

- “Задание переменных среды в Windows” на стр. 36

Создание файла конфигурации узлов

Процедура:

Если база данных работает в среде многораздельных баз данных, необходимо создать файл конфигурации узлов с именем db2nodes.cfg. Чтобы можно было запустить менеджер баз данных с возможностями параллелизма на нескольких разделах, этот файл должен находиться в подкаталоге sqllib начального каталога экземпляра. Этот файл содержит информацию конфигурации для всех разделов баз данных этого экземпляра и совместно используется всеми разделами базы данных для этого экземпляра.

Особенности Windows

При работе с продуктом DB2 Enterprise - Server Edition в операционной системе Windows файл конфигурации узлов автоматически создается при создании экземпляра. Не следует пытаться создавать или изменять файл конфигурации узлов вручную. Для добавления сервера раздела базы данных к экземпляру служит команда **db2ncrt**. Для отбрасывания сервера раздела быза дынных из экземпляра служит команда **db2ndrop**.

Для изменения конфигурации сервера раздела базы данных, в том числе перемещения сервера с одного компьютера на другой, изменения имени хоста TCP/IP и изменения логического порта и сетевого имени, служит команда **db2nchg**.

Примечание: Чтобы избежать возможной потери данных при удалении экземпляра, не следует создавать в подкаталоге `sqllib` файлы или каталоги, кроме уже созданных системой DB2. Есть два исключения. Если используемая система поддерживает хранимые процедуры, помещайте программы хранимых процедур в подкаталог `function` подкаталога `sqllib`. Другое исключение составляет случай, когда созданы пользовательские функции (UDF). Выполняемые файлы пользовательских функций помещаются в этот же каталог.

Этот файл содержит по одной строке для каждого раздела базы данных, входящего в этот экземпляр. Формат каждой из этих строк:

```
номер-раздела  имя-хоста [логический-порт [сетевое-имя]]
```

Переменные разделяются пробелами. Переменные:

`номер-раздела`
Номер раздела базы данных, однозначно идентифицирующий узел. Допустимы значения от 0 до 999. Номера разделов базы данных должны быть заданы в порядке возрастания. В последовательности номеров могут быть пропуски.

После того как номер раздела базы данных был задан, его нельзя изменить. (В противном случае станет недостоверной информация карты разделения, задающей разделение данных.)

После отбрасывания узла его номер можно назначить новому узлу.

Номер раздела базы данных применяется для генерации имени узла в каталоге базы данных. Его формат:

```
NODEnnnn
```

nnnn - это номер раздела базы данных, дополненный слева нулями. Кроме того, номер раздела базы данных применяется в командах **CREATE DATABASE** и **DROP DATABASE**.

`имя_хоста`
Имя хоста IP-адреса для межраздельной связи. (Исключение - когда задан параметр `сетевое-имя`. В этом случае для установления соединений обычно применяется `сетевое-имя`, а `имя-хоста` используется только в командах **db2start**, **db2stop** и **db2_all**.)

`логический_порт`
Этот необязательный параметр задает номер логического порта для

этого узла. Этот номер применяется вместе с именем экземпляра менеджера баз данных при поиске имени службы TCP/IP в файле etc/services.

Сочетание IP-адреса и логического порта применяется в качестве стандартного адреса. Для поддержки соединений между узлами этот адрес должен быть уникальным среди всех прикладных программ.

Для каждого имени_хоста один логический_порт должен иметь значение 0 (ноль) или пустое значение (по умолчанию используется значение 0). Связанный с этим логическим_портом узел - это узел по умолчанию на хосте, с которым соединяются клиенты. Переопределить это можно, задав переменную среды DB2NODE в сценарии db2profile или при помощи функции API sqlesetc().

Если на одном хосте есть несколько узлов (то есть одному хосту соответствует несколько номеров-разделов), присвойте этим логическим узлам номера логических-портов в порядке возрастания, начиная с 0 и без пропусков.

сетевое_имя
Этот необязательный параметр используется для поддержки хоста с несколькими активными интерфейсами TCP/IP, каждый из которых имеет свое собственное имя хоста.

Ниже приведен пример файла конфигурации узла для системы RS/6000 SP, в которой у хоста SP2EN1 есть несколько интерфейсов TCP/IP и два логических раздела; сетевое имя SP2SW1 применяется для интерфейса DB2 Universal Database. Номера разделов заданы, начиная с 1 (а не с 0), и в последовательности номеров-разделов есть пропуск:

Таблица 1. Таблица примера номеров разделов базы данных.

номер-раздела	имя-хоста	логический-порт	сетевое-имя
1	SP2EN1	0	SP2SW1
2	SP2EN1	1	SP2SW1
4	SP2EN2	0	
5	SP2EN3		

Для изменения файла db2nodes.cfg можно использовать любой текстовый редактор. (Исключение: не следует использовать редактор в Windows.) Будьте осторожны, чтобы не нарушить целостность информации в этом файле - для распределения данных требуется, чтобы номера разделов базы данных не менялись. Файл конфигурации узлов блокируется после выполнения команды **db2start** и разблокируется после завершения работы менеджера баз данных командой **db2stop**. Для того чтобы изменить файл, когда он заблокирован,

можно воспользоваться командой **db2start**. Например, можно вызвать команду **db2start** с опцией RESTART или ADDNODE.

Примечание: Если при выполнении команды **db2stop** возникла ошибка, и файл конфигурации узлов не был разблокирован, вызовите команду **db2stop FORCE**, чтобы разблокировать его.

Понятия, связанные с данным:

- "Указания по работе с хранимыми процедурами" в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- "db2start - Start DB2 Command" в *Command Reference*
- "db2stop - Stop DB2 Command" в *Command Reference*
- "CREATE DATABASE Command" в *Command Reference*
- "DROP DATABASE Command" в *Command Reference*
- "db2nchg - Change Database Partition Server Configuration Command" в *Command Reference*
- "db2ncrt - Add Database Partition Server to an Instance Command" в *Command Reference*
- "db2ndrop - Drop Database Partition Server from an Instance Command" в *Command Reference*

Создание файла конфигурации базы данных

Процедура:

Файл конфигурации базы данных создается для каждой базы данных. Это выполняется автоматически. Файл содержит значения различных *параметров конфигурации*, влияющих на использование этой базы данных:

- Параметры, заданные и/или использованные при создании базы данных (например, кодовая страница базы данных, последовательность слияния, уровень выпуска DB2)
- Параметры, указывающие текущее состояние базы данных (например, флаг отложенного резервного копирования, флаг согласованности базы данных, флаг отложенного повтора)
- Параметры, определяющие количество системных ресурсов, которые могут использовать для работы базы данных (например, размер пула буферов, размер журналов базы данных, размер памяти сортировки).

Нельзя изменять эти параметры в конфигурационном файле вручную. Следует использовать только предлагаемый интерфейс.

Совет по улучшению производительности: У многих параметров конфигурации есть значения по умолчанию, но для оптимальной производительности базы данных может потребоваться их изменение.

Для нескольких разделов: Если база данных распределена по нескольким разделам, во всех разделах базы данных должен быть один и тот же файл конфигурации. Это требуется, потому что компилятор SQL компилирует распределенные операторы SQL, используя информацию из локального файла конфигурации, и создает план доступа в соответствии с потребностями этого оператора SQL. Если на разделах базы данных файлы конфигурации отличаются, это может привести к тому, что в зависимости от того, на каком разделе базы данных выполняется подготовка оператора, будут получаться разные планы доступа. Для синхронизации файлов конфигурации на всех разделах базы данных используйте команду **db2_all**.

Понятия, связанные с данным:

- “Вызов команд в среде многораздельной базы данных” на стр. 371

Задачи, связанные с данной темой:

- “Настройка DB2 с помощью параметров конфигурации” в *Руководство администратора: Производительность*

Соединения Менеджера быстрой связи (FCM)

В среде многораздельных баз данных большую часть связей между разделами базы данных обеспечивает менеджер FCM (Fast Communications Manager). Чтобы включить FCM на разделе базы данных и разрешить связь с другими разделами базы данных, в файле `services` этого раздела, находящемся в каталоге `etc`, необходимо создать запись службы, как это описано ниже.FCM использует для связи заданный порт. Если на одном хосте определено несколько разделов, нужно задать диапазон портов, как показано ниже.

Особенности Windows

При работе с продуктом DB2® Enterprise - Server Edition в операционной системе Windows диапазон портов TCP/IP автоматически добавляется в файл служб. Это делает:

- Программа установки, когда она создает экземпляр или добавляет новый узел
- Утилита **db2icrt** при создании нового экземпляра
- Утилита **db2ncrt** при добавлении первого узла на компьютере

Синтаксис записи службы:

```
DB2_экземпляр порт/tcp #комментарий
```

DB2_*экземпляр*

Значение поля *экземпляр* - это имя экземпляра менеджера баз данных.

Все символы имени должны быть введены в нижнем регистре. Например, для экземпляра с именем db2puser нужно задать DB2_db2puser

порт/**tcp**
Порт TCP/IP, который нужно зарезервировать для этого раздела базы данных.

#*комментарий*
Любой комментарий для этой записи. Перед комментарием должен стоять символ #.

Если файл services из каталога etc используется совместно, число заданных в нем портов должно быть не меньше максимального числа разделов базы данных этого экземпляра. При выделении портов не забудьте учесть все компьютеры, которые могут использоваться в качестве резервных.

Если файл services из каталога etc не используется совместно, применяются те же правила, и кроме того, заданные для этого экземпляра DB2 записи должны быть одинаковыми во всех файлах services из каталога etc (хотя другие записи, не относящиеся к этой многораздельной базе данных, могут не совпадать).

Если на одном хосте есть несколько разделов базы данных, нужно задать несколько портов для использования FCM. Для этого задайте в файле services из каталога etc две строки, определяющие выделенный диапазон портов. В первой строке задается первый порт, а во второй - последний порт в этом блоке портов. В следующем примере для экземпляра sales выделяются пять портов. Это означает, что ни один из компьютеров экземпляра не может содержать более пяти разделов базы данных. Например,

```
DB2_sales         9000/tcp
DB2_sales_END     9004/tcp
```

Примечание: Слово END должно задаваться только в верхнем регистре. Должны быть также заданы оба символа подчеркивания (_).

Сервер администратора DB2 (DAS)

Сервер администрирования DB2 (DAS) призван помочь вам при работе с сервером DB2.

Сервер администратора DB2

Сервер администратора DB2® (DAS) - это точка управления DB2, используемая только для задач управления серверами DB2. DAS следует обязательно запустить в том случае, если вы собираетесь работать с такими инструментами, как Помощник по настройке, Центр управления или Центр разработки. Сервер

администратора DB2 работает вместе с Центром управления и Помощником по настройке при выполнении следующих задач администратора:

- Разрешение удаленного управления серверами DB2.
- Обеспечение средств для управления заданиями, включая возможность вносить в расписание выполнение командных сценариев DB2 и командных сценариев операционной системы. Эти командные сценарии задаются пользователем.
- Определение расписания заданий, просмотра результатов выполненных заданий и выполнения других действий по управлению заданиями, расположенными на том же компьютере, что и DAS, или на другом компьютере.
- Обеспечение средств для поиска информации о конфигурации экземпляров DB2, баз данных и других серверов администраторов DB2 при использовании вместе с утилитой поиска DB2. Эта информация используется Помощником по настройке и Центром управления для упрощения и автоматизации процесса конфигурирования соединений клиента с базами данных DB2.

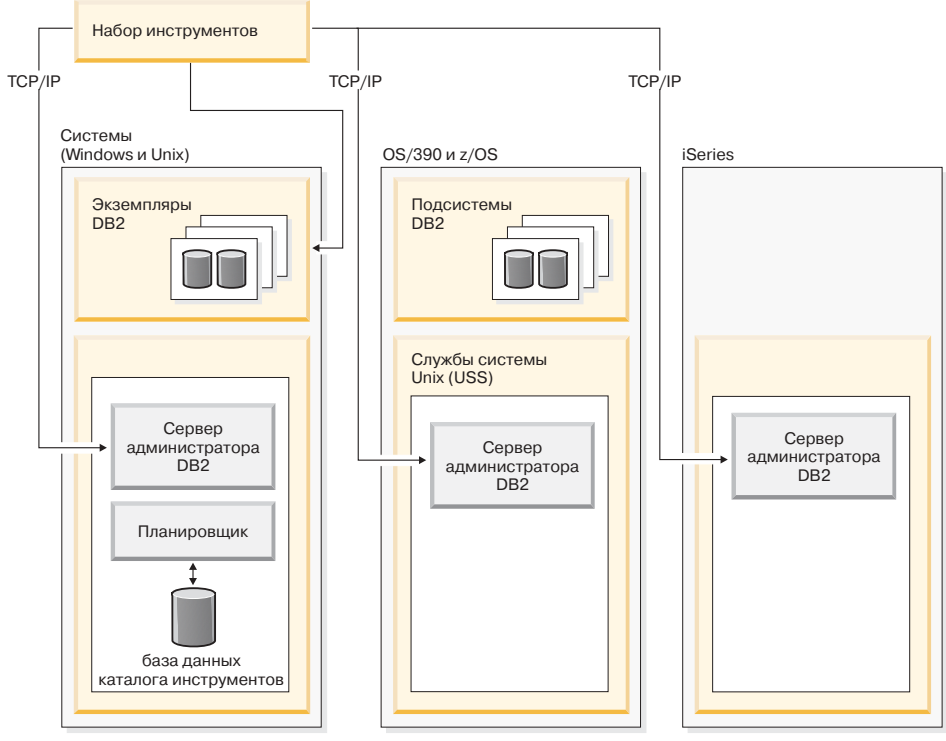


Рисунок 1. Где используется DAS

На компьютере может быть только один DAS. DAS конфигурируется во время установки и запускается при загрузке операционной системы.

DAS выполняет на удаленной системе хоста и сервера запросы клиентов, полученные от Центра управления и Помощника по настройке или других инструментов.

DAS может работать на всех поддерживаемых платформах Windows® и UNIX®, а также на zSeries™ (OS/390 и z/OS™) и iSeries™. DAS на zSeries и iSeries применяется для поддержки Центра управления, Центра разработки и Центра репликации при выполнении административных задач.

Центр администрирования DB2 в zSeries (только OS/390 и z/OS) поставляется как часть средства Клиенты управления DB2. DAS для iSeries также поставляется на отдельном компакт-диске вместе с продуктом Data Propagator for iSeries. Для работы продуктов, которым необходим DAS, таких как Центр управления, Центр разработки и Центр репликации, необходимо установить функцию DAS. Информацию о совместимости DAS с другими операционными системами можно получить в представительстве фирмы IBM®.

В состав DAS для Windows и UNIX входит планировщик задач (таких как сценарии DB2 и командной строки), определенных в Центре задач. Информация о задачах, включая команду, расписание, параметры уведомления и действия при завершении, а также результаты выполнения команды, хранятся в базе данных DB2, называемой Каталог инструментов. База данных Каталог инструментов создается при начальной настройке. Ее также можно создать и включить с помощью Центра управления или из командной строки (команда **CREATE TOOLS CATALOG**).

Хотя планировщик и не поставляется для zSeries (OS/390 и z/OS), с помощью функций Центра управления Построить JCL и Создать JCL можно создать JCL и сохранить его в многораздельных базах данных для запуска системным планировщиком заданий.

Понятия, связанные с данным:

- “Рекомендации по защите сервера администратора в Windows” на стр. 58
- “Настройка DAS в системах Enterprise Server Edition (ESE)” на стр. 63
- “Подключение Центра управления к серверу администратора: порты служб” на стр. 64
- “Управляющие соединения между узлами: Windows DB2 ESE” на стр. 64
- “Поиск серверов администратора, экземпляров и баз данных” на стр. 64

Задачи, связанные с данной темой:

- “Создание Сервера администрирования DB2” на стр. 48

Создание Сервера администрирования DB2

Сервер администрирования DB2 обеспечивает поддержку таких инструментов DB2, как Центр управления и Помощник по настройке.

Предварительные требования:

Для создания DAS вам потребуются права root на платформах UNIX, либо права доступа, позволяющие создавать службы.

В Windows создайте пользователя с полномочиями локального администратора. Введите **db2admin create**. (Чтобы задать конкретные имя пользователя и пароль, для команды **db2admin create** нужно задать опции “/USER:” и “/PASSWORD:”.)

Процедура:

Обычно программа установки во время установки DB2 создает DAS на компьютере - владельце экземпляра. Но если программа установки не создала DAS, его можно создать вручную.

Ниже представлен обзор задач установки, относящихся к DAS:

- На платформах Windows:

  Зарегистрируйтесь на компьютере, на котором нужно создать DAS, с именем пользователя, обладающим правами на создание службы.

  Создавая сервер администратора, можно (хотя и не обязательно) задать учетное имя пользователя и пароль. Если заданы правильные значения, эти

имя пользователя и пароль будут определять владельца сервера администратора. Не используйте ID пользователя или учетное имя, созданные для сервера администратора, в качестве учетной записи пользователя. Для пароля этой учетной записи задайте опцию “Пароль с неограниченным сроком действия”. После создания DAS можно задать или изменить его владельца, задав учетную запись пользователя и пароль при помощи команды **db2admin setid**.

- На платформах UNIX:
  1. Вы должны обладать полномочиями пользователя root.
  2. Введите следующую команду из подкаталога `instance` каталога установки DB2:

     ```
     dascrt -u <пользователь_DAS>
     ```

     где `<пользователь_DAS>` - имя пользователя DAS, созданного при создании пользователей и групп в DB2.

     – В AIX:

     ```
     /usr/opt/db2_08_01/instance/
     dascrt -u <пользователь_DAS>
     ```

     – В HP-UX, Solaris или Linux:

     ```
     /opt/IBM/db2/V8.1/instance/
     dascrt -u <пользователь_DAS>
     ```

Ссылки, связанные с данной темой:

- “db2admin - DB2 Administration Server Command” в *Command Reference*

Запуск и завершение работы DAS

Процедура:

Для того чтобы вручную запустить или завершить работу DAS в операционной системе Windows, необходимо войти в систему с учетной записью или ID пользователя, относящимся к группе Администраторы, Операторы сервера или Опытные пользователи. Для того чтобы вручную запустить или завершить работу DAS в операционной системе Unix необходимо, чтобы учетная запись или ID пользователя относился к группе *dasadm_group*. Значение *dasadm_group* задано в параметрах конфигурации DAS.

Для того чтобы запустить или завершить работу DAS в Windows, вызовите команду **db2admin start** или **db2admin stop**.

При работе с DB2 для любой из операционных систем UNIX необходимо сделать следующее:

- Чтобы запустить сервер администратора:
  1. Зарегистрируйтесь как владелец сервера администратора.

2. Запустите сценарий запуска одной из следующих команд:

```
. DASHOME/das/dasprofile    (в оболочке Bourne и Korn)
source DASHOME/das/dascshrc (в оболочке C)
```

   где DASHOME - это домашний каталог сервера администратора DB2.

3. Для запуска DAS вызовите команду **db2admin**:

```
db2admin start
```

Примечание: Сервер администратора автоматически запускается после каждой перезагрузки системы. Параметры автоматического запуска можно изменить с помощью команды **dasauto**.

- Чтобы остановить сервер администратора:
  1. Войдите в систему с помощью учетной записи или ID пользователя, относящегося к группе *dasadm_group*.
  2. Завершите работу DAS с помощью команды **db2admin**, как показано ниже:

```
db2admin stop
```

Примечание: В обоих случаях для выполнения этих команд в UNIX необходимо зарегистрироваться с ID авторизации владельца сервера администратора. Для применения команд **db2admin start** и **db2admin stop** пользователь должен входить в группу *dasadm_group*.

Ссылки, связанные с данной темой:

- “db2admin - DB2 Administration Server Command” в *Command Reference*
- “Параметр конфигурации Имя группы с правами администратора системы DAS - dasadm_group” в *Руководство администратора: Производительность*

Вывод имени сервера администратора

Процедура:

Для того чтобы узнать имя DAS на локальном компьютере, введите:

```
db2admin
```

Эта команда также применяется для запуска и завершения работы DAS, создания нового пользователя и пароля, отбрасывания DAS и создания и изменения учетной записи пользователя, связанной с DAS.

Ссылки, связанные с данной темой:

- “db2admin - DB2 Administration Server Command” в *Command Reference*

Конфигурирование сервера администратора

Процедура:

Для просмотра текущих значений параметров конфигурации сервера администратора DB2, относящихся к DAS, введите:

```
db2 get admin cfg
```

Будут показаны текущие значения, заданные в качестве значений по умолчанию при установке продукта или при предыдущих изменениях параметров конфигурации.

Для обновления отдельных записей в файле конфигурации DAS введите:

```
db2 update admin cfg using ...
```

Для того чтобы восстановить для параметров конфигурации рекомендуемые значения по умолчанию, введите:

```
db2 reset admin cfg
```

В некоторых случаях изменения, внесенные в файл конфигурации DAS, вступают в силу только после их загрузки в память (то есть после выполнения команд **db2admin stop** и **db2admin start**, либо после перезапуска службы на платформе Windows.) В остальных случаях параметры конфигурации можно настраивать динамически (то есть для применения изменений не требуется перезапускать DAS).

Задачи, связанные с данной темой:

- "Настройка DB2 с помощью параметров конфигурации" в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- "UPDATE ADMIN CONFIGURATION Command" в *Command Reference*

Установка и настройка базы данных каталога инструментов и планировщика DAS

База данных каталога инструментов содержит информацию задач, создаваемых Центром задач и Центром управления. Эти задачи выполняются планировщиком сервера управления DB2.

Предварительные требования:

Должен быть установлен сервер администратора DB2.

Процедура:

Цель состоит в установке и настройке базы данных каталога инструментов и планировщика DAS.

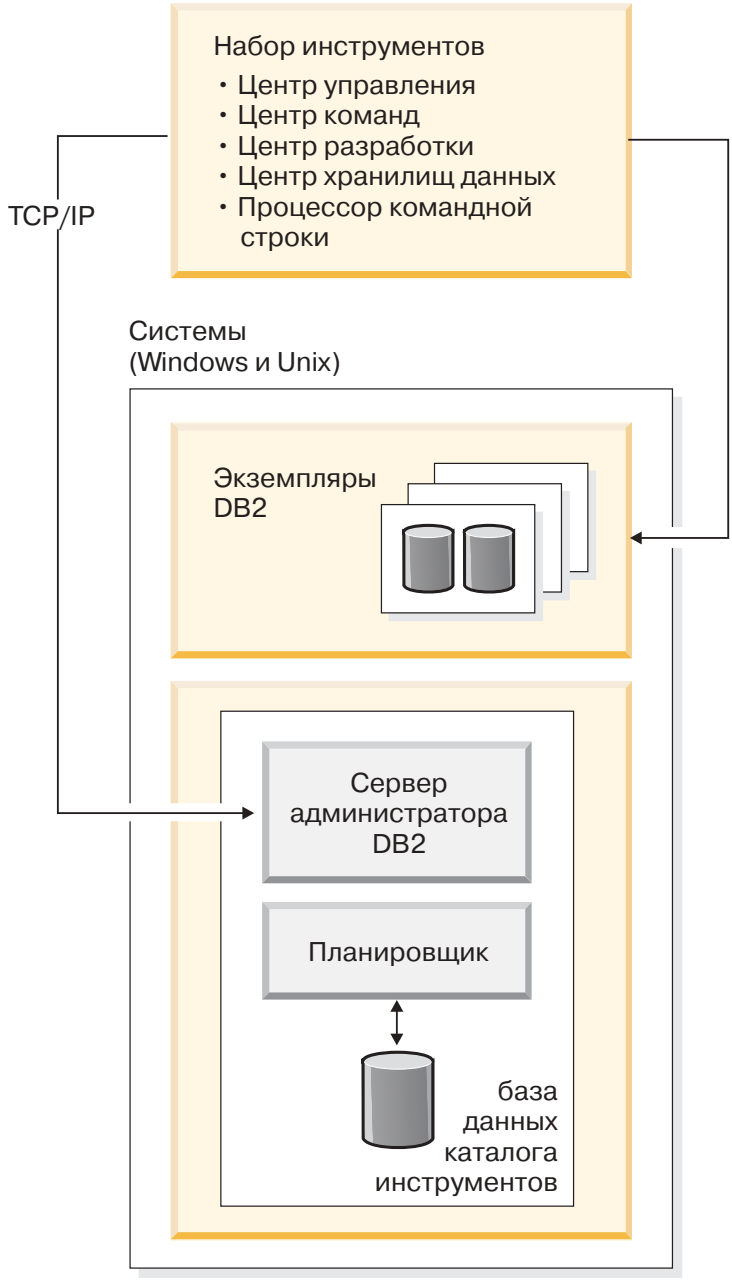


Рисунок 2. Как DAS связан с другими компонентами DB2

Планировщик DAS требует для доступа к информации каталога инструментов наличия Виртуальной машины Java (JVM). Информация JVM задается с помощью параметра конфигурации DAS `jdk_path` сервера администратора DB2.

Центр управления и Центр задач работают с базой данных каталога инструментов непосредственно из системы клиента. Поэтому база данных каталога инструментов должна быть каталогизирована в системе клиента, прежде чем ее сможет использовать Центр управления. Центр управления позволяет автоматически получать информацию о базе данных каталога инструментов и создавать нужные записи каталога в каталоге локального узла и каталоге базы данных. Автоматическая каталогизация поддерживается только для протокола связи TCP/IP.

Один из параметров конфигурации DAS имеет имя *exec_exp_task*. Этот параметр указывает, должен ли Планировщик выполнять задачи, запуск которых был назначен на момент в прошлом, но не произошел. Просроченные задачи проверяются Планировщиком только при запуске.

Например, если задание было запланировано на запуск каждую субботу, а в субботу Планировщик был выключен, при включении в понедельник задание, запланированное на субботу, будет считаться просроченным. Если параметр *exec_exp_task* равен “Yes”, задание будет запущено при запуске Планировщика.

Другие параметры конфигурации DAS, нужные планировщику, идентифицируют базу данных каталога инструментов и сервер SMTP, применяемый для отправки уведомлений.

Применение этих параметров показано на следующих примерах:

- Пример настройки сервера Windows.
  1. База данных каталога инструментов может иметь любое имя. В данном примере база данных каталога инструментов названа “CCMD” и создана в экземпляре DB2, находящемся в системе Host1 (имя хоста TCP/IP). Для идентификации каталога инструментов в базе данных применяется имя схемы. В данном примере схема имеет имя “CCADMIN”.
  2. Экземпляр DB2 настроен на работу по протоколу TCP/IP и использует порт 50000:

     ```
     db2set -i DB2 DB2COMM=TCPIP
     db2 update dbm cfg using svcename db2cDB2
     db2stop
     db2start
     ```

  3. Имя службы `db2cDB2` определено в файле `%SystemRoot%\system32\drivers\etc\services`. Файл `services` должен содержать строку:

     ```
     db2cDB2     50000/tcp     #порт связи для экземпляра DB2
     ```

4. Эту информацию нужно указать на сервере администратора DB2 командой:

   ```
   db2 update admin cfg using toolscat_inst DB2 toolscat_db CCMD
      toolscat_schema CCADMIN
   ```

   Эти параметры DAS можно задать при установке или из Центра управления.

5. Пусть сервер SMTP, применяемый для отправки уведомлений по электронной почте, находится в системе Host2 (имя хоста TCP/IP). Эту информацию нужно указать на сервере администратора DB2 командой:

   ```
   db2 update admin cfg using smtp_server Host2
   ```

   Это может быть сделано во время установки. Позже это можно сделать вручную с помощью команды CLP версии 8 DB2 UDB, показанной выше.

6. Java Development Kit (JDK) установлен в системе Windows в каталоге `%DB2PATH%\java\jdk`. Этот параметр должен быть уже задан для DAS. Его можно задать командой:

   ```
   db2 update admin cfg using jdk_path c:\SQLLIB\java\jdk
   ```

   Эта команда предполагает, что продукт DB2 установлен в каталоге `C:\SQLLIB`.

7. Наконец, необходимо включить планировщик:

   ```
   db2 update admin cfg using sched_enable on
   ```

Для вступления всех внесенных изменений в силу необходимо перезапустить сервер администратора DB2.

- Пример настройки клиента Windows.
  1. Предположим, что Сервер управления работает в клиентской системе C1 (имя хоста TCP/IP).
  2. DAS каталогизирован в каталоге локального узла в качестве узла сервера администратора с помощью Ассистента конфигурирования или Центра управления, либо следующей командой:

     ```
     db2 catalog admin tcpip node Host1 remote Host1 system Host1 ostype NT
     ```

  3. Если запущен Центр задач и выбрана система Host1, Центр задач пытается найти базу данных каталога инструментов в локальном каталоге. (Вместо Центра задач может применяться Центр управления.) Если база данных не будет найдена, Центр пытается каталогизировать узел и базу данных командой:

     ```
     db2 catalog admin tcpip node <уникальное-имя-узла>
       remote Host1 server 50000
       remote_instance DB2 system Host1 ostype NT
     db2 catalog db CCMD as <уникальный-алиас-базы-данных> at
   node <уникальное-имя-узла>
     ```

Если автоматическая каталогизация не будет выполнена, база данных может быть каталогизирована с помощью Ассистента конфигурирования или Центра управления. Это позволит Центру задач обнаружить и использовать базу данных.

- Пример настройки сервера AIX.
  1. База данных каталога инструментов может иметь любое имя. В данном примере база данных каталога инструментов названа “CCMD” и создана в экземпляре db2inst1, находящемся в системе Host1 (имя хоста TCP/IP). Для идентификации каталога инструментов в базе данных применяется имя схемы. В данном примере схема имеет имя “CCADMIN”.
  2. Экземпляр db2inst1 настроен на работу по протоколу TCP/IP и использует порт 50000:

     ```
     db2set -i DB2 DB2COMM=TCPIP
     db2 update dbm cfg using svcename xdb2inst
     db2stop
     db2start
     ```

  3. Имя службы `xdb2inst` определено в файле `%SystemRoot%/etc/services`. Файл `services` должен содержать строку:

     ```
     xdb2inst     50000/tcp     #порт связи для экземпляра DB2
     ```

  4. Эту информацию нужно указать на сервере администратора DB2 командой:

     ```
     db2 update admin cfg using toolscat_inst xdb2inst toolscat_db CCMD
        toolscat_schema CCADMIN
     ```

     Эти параметры DAS можно задать при установке или из Центра управления.
  5. Пусть сервер SMTP, применяемый для отправки уведомлений по электронной почте, находится в системе Host2 (имя хоста TCP/IP). Эту информацию нужно указать на сервере администратора DB2 командой:

     ```
     db2 update admin cfg using smtp_server Host2
     ```

     Это может быть сделано во время установки. Позже это можно сделать вручную с помощью команды CLP версии 8 DB2 UDB, показанной выше.
  6. Java Development Kit (JDK) установлен в системе AIX в каталоге `/usr/java130`. Этот параметр должен быть уже задан для DAS. Его можно задать командой:

     ```
     db2 update admin cfg using jdk_path /usr/java130
     ```

  7. Наконец, необходимо включить планировщик:

     ```
     db2 update admin cfg using sched_enable on
     ```

  Для вступления всех внесенных изменений в силу необходимо перезапустить сервер администратора DB2.
- Пример настройки клиента AIX.

1. Предположим, что Сервер управления работает в клиентской системе C1 (имя хоста TCP/IP).
2. DAS каталогизирован в каталоге локального узла в качестве узла сервера администратора с помощью Ассистента конфигурирования или Центра управления::

   ```
   db2 catalog admin tcpip node Host1 remote Host1 system Host1
     ostype AIX
   ```

3. Если запущен Центр задач и выбрана система Host1, Центр задач пытается найти базу данных каталога инструментов в локальном каталоге. (Вместо Центра задач может применяться Центр управления.) Если база данных не будет найдена, Центр пытается каталогизировать узел и базу данных командой:

   ```
   db2 catalog admin tcpip node <уникальное-имя-узла>
     remote Host1 server 50000
     remote_instance DB2 system Host1 ostype AIX
   db2 catalog db CCMD as <уникальный-алиас-базы-данных>
       at node <уникальное-имя-узла>
   ```

   Если автоматическая каталогизация не будет выполнена, база данных может быть каталогизирована с помощью Ассистента конфигурирования или Центра управления. Это позволит Центру задач обнаружить и использовать базу данных.

Ссылки, связанные с данной темой:

- “Параметр конфигурации Имя службы TCP/IP - svcename” в *Руководство администратора: Производительность*
- “Параметр конфигурации Режим планировщика - sched_enable” в *Руководство администратора: Производительность*
- “Параметр конфигурации Экземпляр базы данных каталога инструментов - toolscat_inst” в *Руководство администратора: Производительность*
- “Параметр конфигурации База данных каталога инструментов - toolscat_db” в *Руководство администратора: Производительность*
- “Параметр конфигурации Схема базы данных каталога инструментов - toolscat_schema” в *Руководство администратора: Производительность*
- “Параметр конфигурации Сервер SMTP - smtp_server” в *Руководство администратора: Производительность*
- “Параметр конфигурации DAS Путь установки Java Development Kit - jdk_path” в *Руководство администратора: Производительность*
- “Параметр конфигурации Исполнять просроченные задачи - exec_exp_task” в *Руководство администратора: Производительность*

Установка и настройка уведомлений и списка контактов

Процедура:

Для получения уведомлений от планировщика или Монитора работоспособности необходимо задать два параметра конфигурации DAS.

Параметр конфигурации DAS *smtp_server* задает сервер SMTP, применяемый для отправки по электронной почте и пейджинговой связи уведомлений о выполнении задач планировщиком и оповещений - Монитором работоспособности.

Параметр конфигурации DAS *contact_host* задает расположение контактной информации, применяемой планировщиком и монитором работоспособности для отправки уведомлений. Расположение задается в виде имени хоста сервера администратора DB2. Размещение списка контактов на удаленном сервере DAS позволяет совместно использовать его нескольким серверам администратора DB2. Эта конфигурация рекомендуется для многораздельных баз данных для обеспечения согласованности списка контактов. Список контактов хранится в текстовом файле в каталоге DAS. Если параметр *contact_host* не задан, контактная информация загружается из локального хоста.

Ссылки, связанные с данной темой:

- “Параметр конфигурации Сервер SMTP - smtp_server” в *Руководство администратора: Производительность*
- “Параметр конфигурации Положение списка адресатов - contact_host” в *Руководство администратора: Производительность*

Установка виртуальной машины Java DAS

Процедура:

В параметре конфигурации *jdk_path* хранится имя каталога установки продукта Java Development Kit (JDK), который будет применяться для выполнения функций сервера управления DB2. На его основе другие переменные среды, применяемые интерпретатором Java.

Планировщик требует для доступа к информации каталога инструментов наличия Виртуальной машины Java (JVM). Виртуальная машина должна быть настроена перед запуском планировщика.

На платформе UNIX для этого параметра не предусмотрено значение по умолчанию. Значение этого параметра должно быть указано при установке Java Development Kit.

Java Development Kit (JDK) установлен в системе Windows в каталоге `%DB2PATH%\java\jdk` (по умолчанию). Этот параметр должен быть уже задан для DAS. Значение параметра *jdk_path* можно проверить командой:

```
db2 get admin cfg
```

Эта команда показывает значения параметров из файла конфигурации сервера администратора DB2, одним из которых является *jdk_path*. При необходимости этот параметр может быть задан командой:

```
db2 update admin cfg using jdk_path 'C:\Program Files\IBM\SQLLIB'
```

Предполагается, что DB2 установлен в каталоге `'C:\Program Files\IBM\SQLLIB'`.

JDK в AIX устанавливается в каталоге `/usr/java130`. При необходимости этот параметр может быть задан командой:

```
db2 update admin cfg using jdk_path /usr/java130
```

Ссылки, связанные с данной темой:

- "GET ADMIN CONFIGURATION Command" в *Command Reference*
- "UPDATE ADMIN CONFIGURATION Command" в *Command Reference*
- "Параметр конфигурации DAS Путь установки Java Development Kit - jdk_path" в *Руководство администратора: Производительность*

Рекомендации по защите сервера администратора в Windows

Вы можете изменить ID пользователя, связанный со службой DAS в операционной системе Windows.

После создания сервера администратора можно задать или изменить учетную запись для входа в систему с помощью команды **db2admin**:

```
db2admin setid <имя_пользователя> <пароль>
```

где `<имя_пользователя>` и `<пароль>` - это имя пользователя и пароль учетной записи с полномочиями локального администратора. Перед вызовом этой команды необходимо войти в систему с помощью учетной записи или ID пользователя, у которого есть полномочия локального администратора.

Примечание: Обратите внимание, что в пароле учитывается регистр символов. Пароль может содержать как прописные, так и строчные буквы, поэтому при вводе пароля важно не перепутать регистр символов.

Примечание: В операционной системе Windows не следует изменять учетную запись DAS с помощью утилиты **Службы Панели управления**, так как в этом случае учетной записи не будут предоставлены некоторые необходимые права доступа. Для задания и изменения учетной записи сервера администратора DB2® (DAS) всегда используйте команду **db2admin**.

Ссылки, связанные с данной темой:

- "db2admin - DB2 Administration Server Command" в *Command Reference*

Обновление сервера администратора в UNIX

Процедура:

В операционной системе UNIX после обновления DB2 путем установки временного исправления программы (PTF) или исправления кода необходимо обновить все серверы администратора DB2 (DAS) и экземпляры. Для обновления DAS вызовите команду **dasupdt**, расположенную в каталоге `instance`. Расположение этого каталога зависит от установленной версии и выпуска DB2.

Войдите в систему под именем привилегированного пользователя (обычно это пользователь “root”).

Эта команда используется так:

```
dasupdt
```

В команде предусмотрены следующие необязательные параметры:

- –h или –?

  Выводит для этой команды меню справки.
- –d

  Задает режим отладки, используемый для анализа ошибок.

Примечание: В операционной системе Windows обновление DAS выполняется во время установки. Пользователю обновлять сервер не требуется.

Удаление сервера администратора

Процедура:

Чтобы удалить сервер администратора:

- В операционной системе Windows:
  1. Войдите в систему с помощью учетной записи или ID пользователя, у которого есть полномочия на удаление службы.
  2. Остановите сервер администратора, используя команду **db2admin stop**.
  3. Если нужно, сделайте резервную копию всех файлов из подкаталога `db2das00` подкаталога `sqllib`.

     Примечание: В этом примере подразумевается, что имя удаляемого сервера администратора - `db2das00`. Имя DAS может быть отлично от DB2DAS00, если пользователь создал экземпляр DB2 с именем DB2DAS00. В этом случае имя DAS будет равно DB2DAS01 (а если имя DB2DAS01 занято, то DB2DAS02, и так далее). В списке служб может

присутствовать несколько имен с префиксом “DB2DAS”, соответствующих разным серверам администратора. Вам нужно выбрать одно из них. Для просмотра списка всех DAS вызовите команду **db2admin** без параметров.

4. Отбросьте сервер администратора, используя команду **db2admin drop**.

- В операционных системах UNIX:
  1. Войдите в систему под именем пользователя с полномочиями DASADM.
  2. Запустите сценарий запуска, используя одну из следующих команд:

     ```
     . DASHOME/das/dasprofile    (в оболочке Bourne или Korn)
     source DASHOME/das/dascshrc (в оболочке C)
     ```

     где DASHOME - домашний каталог владельца DAS.
  3. Завершите работу DAS с помощью команды **db2admin**, как показано ниже:

     ```
     db2admin stop
     ```
  4. При необходимости создайте резервную копию всех файлов из подкаталога das, расположенного в домашнем каталоге DAS.
  5. Выйдите из системы.
  6. Войдите в систему как пользователь root и удалите DAS с помощью команды **dasdrop**:

     ```
     dasdrop
     ```

     Команда **dasdrop** находится в подкаталоге instance. Расположение этого подкаталога зависит от установленной версии и выпуска DB2.

     Примечание: Команда **dasdrop** удаляет подкаталог das в домашнем каталоге сервера администратора DB2 (DAS).

Ссылки, связанные с данной темой:

- “db2admin - DB2 Administration Server Command” в *Command Reference*

Установка DAS в системах Enterprise Server Edition (ESE)

Процедура:

Ниже описана пошаговая процедура настройки сервера DB2 ESE (Linux, Solaris, Windows NT, Windows 2000, Windows .NET, HP-UX и AIX) для удаленного администрирования. Эта процедура выполняется с помощью Центра управления.

Во время установки программа установки создает один сервер администратора на компьютере - владельце экземпляра. Для того чтобы у Центра управления или Ассистента конфигурирования был доступ к другим узлам координатора, создайте дополнительные серверы администратора на других компьютерах.

После этого можно будет распределить нагрузку, связанную с выполнением функций узла координатора, по нескольким разделам экземпляра. Программа установки создаст DAS на всех узлах, на которых она запущена. Создавать DAS вручную требуется только в том случае, если команда **db2setup** не применяется.

Приведенные ниже инструкции относятся только к многораздельной среде ESE. Если вы работаете с одноразделной системой ESE, не выполняйте эти инструкции.

Чтобы распределить функции координатора:

1. Создайте новые серверы администратора на выбранных дополнительных компьютерах системы многораздельных баз данных.
2. В Центре управления или Ассистенте конфигурирования добавьте каждый сервер администратора в каталог как отдельную систему.
3. Внесите в каталог этот экземпляр на каждой новой системе, каждый раз задавая имя компьютера, которое вы использовали для внесения в каталог сервера администратора.

Существует два аспекта задачи конфигурирования: То, что необходимо сделать для сервера администратора DB2, и то, что рекомендуется сделать для управляемого им экземпляра DB2. Этим двум темам посвящены два из последующих трех разделов. Перед ними в предварительном разделе описывается предполагаемая среда.

Пример среды

продукт/версия:
DB2 UDB ESE V8.1

путь установки:
путь_установки

файл служб TCP:
services

Экземпляр DB2:

имя: db2inst

ID владельца:
db2inst

путь экземпляра:
путь_экземпляра

узлы: 3 узла, db2nodes.cfg:

- 0 hostA 0 hostAswitch
- 1 hostA 1 hostAswitch

- 2 hostB 0 hostBswitch

Имя базы данных:
db2instDB

Сервер администратора:

имя: db2as00

ID владельца/пользователя:
db2as

путь экземпляра:
путь_das

хост установки/выполнения:
hostA

порт межузловой связи:
16000 (неиспользуемый порт для hostA и hostB)

Примечание: Подставьте в показанные выше поля значения для конкретных систем. Например, в следующей таблице приведены примеры путей для некоторых поддерживаемых платформ ESE:

Таблица 2. Примеры путей для поддерживаемых платформ ESE

Пути	DB2 UDB ESE for AIX	DB2 UDB ESE for Solaris	DB2 UDB ESE for Windows
путь_установки	/usr/opt/<v_r_ID>	/opt/IBM/db2/<v_r_ID>	C:\sqllib
путь_экземпляра	/home/db2inst/sqllib	/home/db2inst/sqllib	C:\profiles\db2inst
путь_das	/home/db2as/das	/home/db2as/das	C:\profiles\db2as
файл_служб_tcp	/etc/services	/etc/services	C:\winnt\system32 \drivers\etc\services

В этой таблице <v_r_ID> - это зависящий от платформы идентификатор версии и выпуска. Например, в DB2 UDB ESE for AIX версии 8 значение <v_r_ID> равно db2_08_01.

При установке DB2 UDB ESE программа установки создает сервер администратора на компьютере-владельце экземпляра. Сервер раздела базы данных находится на том же компьютере, что и сервер администратора, и этот сервер является точкой соединения для экземпляра. Это означает, что этот сервер раздела базы данных является узлом координатора для запросов к экземпляру, отправляемых из Центра управления или Ассистента конфигурирования.

Если сервер администратора установлен на всех физических компьютерах, то любой из них может играть роль узла координатора. В Центре управления и

Ассистенте конфигурирования каждый физический компьютер представлен в виде отдельной системы DB2SYSTEM. Если для подключения к серверу многораздельной базы данных различные клиенты применяют разные системы, то функции узла координатора и соединения с клиентами можно распределить между различными компьютерами.

Настройка DAS в системах Enterprise Server Edition (ESE)

Сервер администратора (DAS) - это точка управления, выполняющая определенные задачи при помощи инструментов. На физическом компьютере может быть не более одного сервера администратора. В случае экземпляра ESE, включающего несколько компьютеров, необходимо установить сервер администратора на всех компьютерах, для того чтобы можно было работать с экземпляром ESE в Центре управления. Сервер администратора (DB2DAS00) представлен в дереве объектов Центра управления в виде родительской системы управляемого экземпляра DB2® (db2inst).

В нашем примере db2inst состоит из трех узлов, размещенных на двух физических компьютерах или хостах. Для выполнения минимальных требований достаточно запустить **db2as** на хосте hostA **или** hostB.

Примечания:

1. Число разделов на хосте hostA не влияет на число серверов администратора, которые можно запустить на этом хосте. На хосте hostA можно запустить только одну копию **db2as**, независимо от того, что на нем настроено нескольких логических узлов (MLN).
2. ID сервера администратора, DB2DASxx, не обязательно создавать на всех хостах. Требуется только, чтобы он существовал на хосте, на котором он работает. На каждом компьютере, или физическом узле, должен быть запущен один сервер администратора, который можно создать независимо от остальных серверов с помощью команды **dascrt**. Наличие сервера администратора на каждом компьютере, или физическом узле, необходимо для правильной работы Центра задач и Центра управления. Домашний каталог DAS **не** следует монтировать на другом узле, если на нем применяется тот же ID владельца DAS. В рассматриваемом примере ID db2as должен существовать на хосте hostA, но не требуется на хосте hostB, и домашний каталог db2as не следует монтировать на хосте hostB.

Если при работе с DB2 UDB for Windows® Enterprise - Server Edition для автоматической настройки соединения с сервером DB2 применяется Ассистент конфигурации или Центр управления, то узлом координатора станет сервер раздела базы данных, расположенный на том же компьютере, что и DAS. Это означает, что все физические соединения, устанавливаемые между клиентами и базой данных, будут направляться узлу координатора, а лишь затем - перенаправляться на другой сервер раздела базы данных.

В DB2 UDB for Windows® Enterprise - Server Edition создание дополнительных серверов администратора DB2 на других компьютерах позволяет Ассистенту конфигурации или Центру управления настроить другие системы в качестве узлов координатора с помощью функции поиска DB2.

Задачи, связанные с данной темой:

- “Создание Сервера администрирования DB2” на стр. 48

Ссылки, связанные с данной темой:

- “db2admin - DB2 Administration Server Command” в *Command Reference*

Подключение Центра управления к серверу администратора: порты служб

Центр управления подключается к серверу администратора (DAS) через порт службы TCP с номером 523. Поскольку этот порт зарезервирован для DB2® UDB, то добавлять записи в файл служб TCP необязательно.

Управляющие соединения между узлами: Windows DB2 ESE

Служба удаленных команд DB2® (**db2rcmd.exe**) автоматически обслуживает управляющие соединения между узлами.

Поиск серверов администратора, экземпляров и баз данных

Соединение с удаленным компьютером можно настроить двумя способами: с помощью службы поиска, предусмотренной в Ассистенте конфигурирования, или с помощью существующей службы каталогов, например, LDAP.

Служба поиска - это встроенная функция Ассистента конфигурирования и сервера администратора DB2®. Для настройки соединения с удаленным компьютером пользователь должен войти в систему компьютера-клиента и запустить Ассистент конфигурирования (CA). CA отправит оповещающий сигнал всем компьютерам сети. Все компьютеры, на которых установлен сервер администратора с настроенной функцией поиска, отправят в ответ на сигнал CA пакет, содержащий полную информацию о локальном экземпляре и базе данных. С помощью полученной информации CA настроит соединение клиента. При работе с функцией поиска информация каталога для удаленного сервера может автоматически создаваться в локальной базе данных и каталоге узла.

Для применения функции поиска необходимо войти в систему и запустить CA на каждом компьютере-клиенте. При наличии большого числа клиентов это потребует много времени и сил. В этом случае лучше воспользоваться службой каталогов, например, LDAP.

Функция поиска KNOWN позволяет находить экземпляры и базы данных в системах, известных клиенту, и добавлять новые системы для поиска на них

экземпляров и баз данных. Функция поиска SEARCH обеспечивает все возможности поиска KNOWN, а также позволяет искать другие серверы DB2 в локальной сети.

Для того чтобы в системе поддерживалась функция поиска KNOWN, присвойте параметру *discover* в файле конфигурации DAS значение `KNOWN`. Для того чтобы в системе поддерживались обе функции поиска (KNOWN и SEARCH), присвойте параметру *discover* в файле конфигурации DAS значение `SEARCH` (это значение установлено по умолчанию). Для того чтобы другие компьютеры не могли получать информацию о системе, ее экземплярах и базах данных с помощью функции поиска, присвойте этому параметру значение `DISABLE`. Если параметру *discover* в файле конфигурации DAS будет присвоено значение `DISABLE`, то система не будет обнаружена функцией поиска.

Примечание: Функция поиска SEARCH возвращает клиенту то же имя TCP/IP хоста, которое сообщает система сервера DB2 при выполнении команды **hostname**. Чтобы узнать соответствующий этому имени хоста IP-адрес, клиент использует определенный на нем сервер имен доменов TCP/IP (DNS) или, если DNS не определен, запись отображения в файле *hosts* этого клиента. Если в системе сервера DB2 сконфигурировано несколько плат адаптеров, нужно убедиться, что конфигурация TCP/IP на этом сервере возвращает правильное имя хоста и что DNS или файл *hosts* клиента отображает это имя в правильный IP-адрес.

Чтобы разрешить функцию поиска на клиенте, используется также параметр *discover*, но в этом случае параметр задается для экземпляра клиента (или сервера, работающего как клиент) так:

- **KNOWN**

  Функция поиска KNOWN применяется Ассистентом конфигурирования и Центром управления для получения информации об экземплярах и базах данных тех систем, которые уже известны локальной системе. Для добавления новых систем в этих программах предусмотрена функция **Добавить системы**. Если параметру *discover* будет присвоено значение `KNOWN`, вы не сможете выполнять поиск в сети.

- **SEARCH**

  Предоставляет те же возможности, что и функция поиска KNOWN, и дополнительно позволяет выполнять поиск в локальной сети. Поиск будет выполняться только в локальной сети.

  Значок “Другие системы (искать в сети)” появляется, только если задано это значение параметра. Это значение применяется по умолчанию.

- **DISABLE**

  Запрещает функцию поиска. В этом случае в “Мастере по добавлению баз данных” не будет доступна функция **Поиск в сети**.

Примечание: По умолчанию параметр *discover* имеет значение SEARCH на всех экземплярах клиентов и серверов. Кроме того, параметр *discover* по умолчанию равен SEARCH на всех серверах администратора DB2 (DAS).

Понятия, связанные с данным:

- “Служба каталога протокола LDAP (Lightweight Directory Access Protocol)” на стр. 79

Задачи, связанные с данной темой:

- “Как скрыть экземпляры и базы данных сервера от функции поиска” на стр. 66
- “Настройка параметров функции поиска” на стр. 66

Как скрыть экземпляры и базы данных сервера от функции поиска

Процедура:

В системе сервера может присутствовать несколько экземпляров и несколько баз данных. Некоторые из них можно скрыть от функции поиска.

Чтобы разрешить клиентам обнаруживать экземпляры сервера в какой-либо системе, задайте для параметра конфигурации менеджера баз данных *discover_inst* в каждом экземпляре сервера в этой системе значение ENABLE (это значение по умолчанию). Задайте для этого параметра значение DISABLE, чтобы скрыть этот экземпляр и его базы данных от функции поиска.

Чтобы разрешить клиентам обнаруживать базу данных, задайте для ее параметра конфигурации *discover_db* значение ENABLE (это значение по умолчанию). Задайте для этого параметра значение DISABLE, чтобы скрыть эту базу данных от функции поиска.

Примечание: Для того чтобы функция поиска могла обнаружить экземпляр, в файле конфигурации DAS в качестве функции поиска должно быть указано значение KNOWN или SEARCH. Для того чтобы функция поиска могла обнаружить базу данных, в экземпляре сервера должен быть установлен параметр *discover_inst*.

Ссылки, связанные с данной темой:

- “Параметр конфигурации Экземпляр сервера Discover - discover_inst” в *Руководство администратора: Производительность*
- “Параметр конфигурации Поиск баз данных - discover_db” в *Руководство администратора: Производительность*

Настройка параметров функции поиска

Процедура:

Параметр *discover* задается в файле конфигурации DAS на сервере и в файле конфигурации менеджера баз данных на клиенте. С помощью Ассистента конфигурирования и Центра управления можно настроить следующие параметры конфигурации менеджера баз данных: *discover*, *discover_inst*, *discover_db*. Чтобы задать эти параметры:

- На сервере администратора:

  Измените параметр *discover* в файле конфигурации DAS (как показано в примере), введя в командной строке:

  ```
  update admin cfg using discover [ DISABLE | KNOWN |
    SEARCH ]
  ```

  Параметр конфигурации DAS *discover* можно изменять динамически, то есть для изменения его значения не нужно останавливать и снова запускать сервер администратора.

  Примечание: Функция поиска SEARCH поддерживает только протокол TCP/IP.

- С помощью Ассистента конфигурирования. Запустите Ассистент конфигурирования. Для этого введите в командной строке **db2ca** (на любой платформе) или откройте меню Пуск в Windows: **Пуск —> Программы —> IBM DB2 —> Ассистент конфигурирования**. С помощью Ассистента конфигурирования выполните следующие действия:

  1. Выберите в меню Настроить пункт **Конфигурация DBM**.
  2. Выберите ключевое слово, которое нужно изменить.
  3. Выберите значение ключевого слова в поле **Значение** и нажмите кнопку **OK**.
  4. Нажмите кнопку **OK** еще раз. После появления сообщения нажмите кнопку **Закрыть**. Будет снова показано главное окно.

Параметры *discover_inst* и *discover_db* можно настроить с помощью Центра управления.

Ассистент конфигурирования также позволяет обновить параметры конфигурации.

Ссылки, связанные с данной темой:

- “Параметр конфигурации Экземпляр сервера Discover - discover_inst” в *Руководство администратора: Производительность*
- “Параметр конфигурации Поиск баз данных - discover_db” в *Руководство администратора: Производительность*
- “UPDATE ADMIN CONFIGURATION Command” в *Command Reference*
- “Параметр конфигурации Режим поиска DAS - discover” в *Руководство администратора: Производительность*

Настройка DAS для работы с Ассистентом конфигурирования и Центром управления

Предварительные требования:

Настройте параметр **discover** для получения информации о системах из сети.

Ограничения:

DAS должен быть установлен во всех физических разделах. При создании DAS в разделе настраивается хост TCP/IP с именем DB2SYSTEM, а параметру **discover** присваивается значение по умолчанию SEARCH.

Процедура:

Функция поиска DB2 применяется Ассистентом конфигурирования и Центром управления. Для того чтобы функция поиска DB2 получала правильную информацию, может потребоваться изменить конфигурацию сервера администратора DB2 (DAS) и конфигурацию менеджера баз данных экземпляра.

На поисковый запрос клиента, отправленный с помощью Ассистента конфигурирования или Центра управления, отвечают все DAS с настроенной функцией поиска. В среде многораздельной базы данных каждый физический раздел отвечает как независимый хост DB2SYSTEM. Любой физический раздел может применяться для управления экземплярами, известными этому разделу. Если один экземпляр распределен по нескольким разделам, для управления этим экземпляром могут применяться разные системы. Такая возможность позволяет распределить нагрузку, связанную с экземпляром сервера. Например, если экземпляр “A” доступен в системах “S1” и “S2”, то некоторые пользователи могут добавлять базу данных в каталог с помощью системы “S1”, а некоторые - с помощью системы “S2”. При этом разные пользователи могут подключаться к серверу с помощью разных разделов координатора.

Ссылки, связанные с данной темой:

- “db2ilist - List Instances Command” в *Command Reference*
- “db2ncrt - Add Database Partition Server to an Instance Command” в *Command Reference*
- “Параметр конфигурации Режим поиска DAS - discover” в *Руководство администратора: Производительность*

Изменение конфигурации DAS для применения функции поиска

Ограничения:

DAS должен быть установлен во всех физических разделах. При создании DAS в разделе настраивается хост TCP/IP с именем DB2SYSTEM, а параметру *discover* присваивается значение по умолчанию SEARCH.

Процедура:

Получаемые функцией поиска имена систем - это имена систем, на которых находятся серверы администратора DB2 (DAS). При установлении соединений функция поиска использует эти системы в качестве узлов координаторов.

Если вы хотите, чтобы у вас была возможность выбирать узел координатора в списке систем DB2, то при обновлении конфигурации DAS укажите *discover*=SEARCH (это значение применяется по умолчанию) во всех файлах конфигурации сервера администратора DB2.

Если в среде сервера многораздельной базы данных создано несколько DAS, в интерфейсе Ассистента конфигурирования и Центра управления один экземпляр может содержаться в нескольких системах. Однако в каждой системе будет применяться свой путь доступа к экземпляру. Пользователи могут выбирать разные системы DB2 в качестве узлов координаторов для связи и таким образом перераспределять рабочую нагрузку.

Ссылки, связанные с данной темой:

- "Разные переменные" в *Руководство администратора: Производительность*

Сбор данных первого сбоя сервера администратора DB2

Сбор данных первого сбоя (FFDC) - это общий термин, относящийся к диагностической информации, автоматически собираемой и сохраняемой сервером администратора DB2® при возникновении ошибки. Эта информация позволяет избежать необходимости воспроизведения ошибок для получения диагностической информации. Вся диагностическая информация хранится в одном расположении.

FFDC сервера администратора DB2 собирает следующую информацию:

- Журналы уведомлений администратора.

  При возникновении события сервер администратора DB2 сохраняет информацию в файле журнала сервера администратора DB2 с именем `db2dasdiag.log`.
- Файлы дампа.

  Для некоторых ошибок собирается дополнительная информация, сохраняемая во внешних двоичных файлах, имя которых совпадает с идентификатором процесса, в котором был обнаружен сбой. Эти файлы также предназначены для использования сервисным представительством.
- Файлы дампа.

Если продолжение работы невозможно из-за прерывания, исключения или нарушения сегментации, сервер администратора DB2 создает файл прерываний. Файлы прерывания содержат информацию о последних действиях функции перед обнаружением ошибки.

Расположение собранных данных FFDC сервера администратора DB2

По умолчанию собранные данные FFDC сервера администратора DB2 помещаются в следующие расположения:

- В системах Windows®:

  Если переменная среды DB2INSTPROF не задана:

  ```
  каталог-db2\DB2DAS00\dump
  ```

  где `каталог-db2` - каталог, указанный в переменной среды DB2PATH, а `DB2DAS00` - имя службы DAS. Имя DAS можно просмотреть командой **db2admin** без аргументов.

  Если переменная среды DB2INSTPROF задана:

  ```
  x:\db2instprof\DB2DAS00\dump
  ```

  где `x:` - буква диска, указанного в переменной среды DB2PATH, `db2instprof` - каталог профиля экземпляра, а `DB2DAS00` - имя службы DAS.

- В системах UNIX:

  ```
  $DASHOME/das/dump
  ```

  где `$DASHOME` - домашний каталог пользователя DAS.

Примечание: Рекомендуется регулярно очищать каталог дампа, чтобы он не занимал слишком много места.

Интерпретация журнала сервера администратора DB2.

Формат файла журнала сервера администратора DB2 (`db2dasdiag.log`) аналогичен формату файла журнала FFDC DB2 `db2diag.log`. Для получения информации об интерпретации журналов администратора обратитесь к разделам по интерпретации файла db2dasdiag.log.

Глава 2. Создание базы данных

В этой главе кратко описаны все различные объекты, которые могут входить в структуру базы данных.

Предыдущая глава была посвящена тому, что нужно знать перед созданием базы данных. В ней также были рассмотрены некоторые операции, которые должны быть выполнены перед созданием базы данных.

В последней главе этой части книги описывается, что нужно учесть перед изменением базы данных. В ней также объясняется, как изменить или отбросить объекты базы данных.

Создание базы данных

Предварительные требования для установки:

Перед созданием базы данных обязательно уделите достаточно внимания изучению ее будущего содержимого, структуры, возможностей расширения и способов использования.

Процедура:

При создании базы данных автоматически выполняются следующие действия:

- Настраиваются все таблицы системного каталога, необходимые для этой базы данных
- Выделяется журнал восстановления базы данных
- Создается файл конфигурации базы данных и в нем задаются значения по умолчанию
- Выполняется связывание утилит баз данных с этой базой данных

В производных таблицах системного каталога пользователю PUBLIC автоматически предоставляются следующие привилегии для этой базы данных: CREATETAB, BINDADD, CONNECT, IMPLICIT_SCHEMA и SELECT.

Чтобы создать базу данных с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Базы данных**.
2. Щелкните правой кнопкой мыши по папке **Базы данных** и выберите из всплывающего меню пункт **Создать —> Базу данных при помощи мастера**.
3. Выполните шаги в этом мастере.

Следующая команда процессора командной строки создает в положении по умолчанию базу данных с именем `personl` и с комментарием ″База данных персонала для компании BSchiefer″.

```
CREATE DATABASE personl
   WITH "База данных персонала для компании BSchiefer"
```

Кроме того, при создании базы данных можно вместо принятия значений параметров по умолчанию воспользоваться Мастером настройки производительности. Для этого укажите опцию AUTOCONFIGURE команды **CREATE DATABASE**:

```
CREATE DATABASE <имя базы данных>
   AUTOCONFIGURE
```

У условия AUTOCONFIGURE есть несколько опций. Условие AUTOCONFIGURE нельзя использовать в среде с несколькими разделами.

Вы можете создать базу данных в другом, возможно, удаленном экземпляре менеджер баз данных. В этом случае вы можете выполнять любые задачи администратора уровня экземпляра для экземпляров, отличных от экземпляра по умолчанию, в том числе для удаленных экземпляров.

Понятия, связанные с данным:

- “Что хранить в базе данных” в *Руководство администратора: Планирование*
- “Несколько экземпляров менеджера баз данных” на стр. 6
- “Привилегии базы данных” на стр. 250
- “Дополнительные особенности структуры базы данных” в *Руководство администратора: Планирование*

Ссылки, связанные с данной темой:

- “CREATE DATABASE Command” в *Command Reference*

Определение начальных групп разделов базы данных

При создании базы данных создаются разделы базы данных для всех разделов, заданных в файле db2nodes.cfg. Другие разделы можно добавить или удалить с помощью команд **ADD DBPARTITIONNUM** и **DROP DBPARTITIONNUM VERIFY**.

Определяются три группы баз данных:

- IBMCATGROUP для табличного пространства SYSCATSPACE, в котором хранятся таблицы системного каталога
- IBMTEMPGROUP для табличного пространства TEMPSPACE1, в котором хранятся временные таблицы, созданные при обработке базы данных
- IBMDEFAULTGROUP для табличного пространства USERSPACE1, в котором по умолчанию хранятся пользовательские таблицы и индексы.

Понятия, связанные с данным:

- "Группы разделов базы данных" в *Руководство администратора: Планирование*

Ссылки, связанные с данной темой:

- "ADD DBPARTITIONNUM Command" в *Command Reference*
- "DROP DBPARTITIONNUM VERIFY Command" в *Command Reference*

Определение начальных табличных пространств

При создании базы данных определяются три табличных пространства:

- SYSCATSPACE для таблиц системного каталога
- TEMPSPACE1 для временных таблицы, создаваемых при обработке базы данных
- USERSPACE1 для пользовательских таблиц и индексов

Примечание: При создании базы данных не создаются пользовательские временные табличные пространства.

Если в команде **CREATE DATABASE** не заданы параметры табличных пространств, менеджер баз данных создает эти табличные пространства, используя контейнеры каталога управляемого системой хранения (SMS). Эти контейнеры каталога создаются в подкаталоге, созданном для этой базы данных. Размеру экстента для этих табличных пространств присваивается значение по умолчанию.

Предварительные требования для установки:

База данных уже должна существовать, а у вас должны быть полномочия на создание табличных пространств.

Процедура:

Чтобы определить исходные табличные пространства с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Базы данных**.
2. Щелкните правой кнопкой мыши по папке **Базы данных** и выберите из всплывающего меню пункт **Создать —> Базу данных при помощи мастера**.
3. Выполните шаги в этом мастере.

Чтобы определить исходные табличные пространства из командной строки, введите команду:

```
CREATE DATABASE <имя>
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('<путь>')
    EXTENTSIZE <значение> PREFETCHSIZE <значение>
  USER TABLESPACE
    MANAGED BY DATABASE USING (FILE'<путь>' 5000,
                               FILE'<путь>' 5000)
    EXTENTSIZE <значение> PREFETCHSIZE <значение>
  TEMPORARY TABLESPACE
    MANAGED BY SYSTEM USING ('<путь>')
  WITH "<комментарий>"
```

Если вы не хотите использовать для этих табличных пространств определение по умолчанию, их характеристики можно задать в команде **CREATE DATABASE**. Например, для создания базы данных в Windows может использоваться команда:

```
CREATE DATABASE PERSONL
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('d:\pcatalog','e:\pcatalog')
    EXTENTSIZE 16 PREFETCHSIZE 32
  USER TABLESPACE
    MANAGED BY DATABASE USING (FILE'd:\db2data\personl' 5000,
                               FILE'd:\db2data\personl' 5000)
    EXTENTSIZE 32 PREFETCHSIZE 64
  TEMPORARY TABLESPACE
    MANAGED BY SYSTEM USING ('f:\db2temp\personl')
  WITH "База данных персонала для компании BSchiefer"
```

В этом примере явно задано определение для каждого из исходных табличных пространств. Определения табличных пространств нужно задать только для тех табличных пространств, для которых не подходит определение по умолчанию.

Примечание: В среде многораздельных баз данных создавать и присваивать контейнеры конкретным разделам нельзя. Сначала вам нужно создать базу данных с пользователем по умолчанию и временными табличными пространствами. Затем вы можете создать необходимые табличные пространства с помощью выражения CREATE TABLESPACE. Затем удалите табличные пространства по умолчанию.

Условие MANAGED BY команды **CREATE DATABASE** имеет тот же формат, что и условие MANAGED BY команды CREATE TABLESPACE.

Понятия, связанные с данным:

- “Определение таблиц системного каталога” на стр. 75
- “Проектирование табличного пространства” в *Руководство администратора: Планирование*

Задачи, связанные с данной темой:

- “Создание табличного пространства” на стр. 85

Ссылки, связанные с данной темой:

- “CREATE DATABASE Command” в *Command Reference*

Определение таблиц системного каталога

Для каждой базы данных создается и поддерживается набор таблиц системного каталога. Эти таблицы содержат информацию об определениях объектов базы данных (например, таблиц, производных таблиц, индексов и пакетов) и информацию защиты о том, какой тип доступа разрешен пользователям для этих объектов. Эти таблицы хранятся в табличном пространстве SYSCATSPACE.

Эти таблицы обновляются при операциях с базой данных (например, при создании таблиц). Нельзя явно создать или отбросить эти таблицы, но можно запросить и просмотреть их содержимое. При создании базы данных кроме объектов таблиц системного каталога в системном каталоге определяются следующие объекты базы данных:

- Набор подпрограмм (функций и процедур) в схемах SYSIBM, SYSFUN и SYSPROC.
- В схеме SYSCAT для таблиц системного каталога создается набор производных таблиц только для чтения.
- В схеме SYSSTAT создается набор производных таблиц каталога, которые можно изменять. Эти производные таблицы позволяют изменять некоторые

виды статистической информации, чтобы исследовать производительность предполагаемой базы данных, или обновлять статистики без использования утилиты **RUNSTATS**.

После создания базы данных можно ограничить доступ к производным таблицам системного каталога.

Понятия, связанные с данным:

- "User-defined functions" в *SQL Reference, Том 1*
- "Catalog views" в *SQL Reference, Том 1*
- "Functions overview" в *SQL Reference, Том 1*

Задачи, связанные с данной темой:

- "Защита производных таблиц системного каталога" на стр. 278

Ссылки, связанные с данной темой:

- "Functions" в *SQL Reference, Том 1*

Определение каталогов баз данных

При создании или настройке новой базы данных используются три каталога.

- Каталог локальных баз данных
- Системный каталог баз данных
- Каталог узла

Каталог локальных баз данных

Файл *локального каталога баз данных* существует в каждом пути (называемом в Windows® "диском"), в котором определена база данных. Этот каталог содержит записи для всех баз данных, доступных из этой системы. Каждая запись содержит:

- Имя базы данных, заданное в команде **CREATE DATABASE**.
- Алиас базы данных (совпадающий с именем базы данных, если алиас не задан)
- Описывающий базу данных комментарий, заданный в команде **CREATE DATABASE**.
- Имя корневого каталога этой базы данных
- Другую системную информацию.

Ссылки, связанные с данной темой:

- "CREATE DATABASE Command" в *Command Reference*

Системный каталог баз данных

Для каждого экземпляра менеджера баз данных существует файл *системного каталога баз данных*, содержащий записи для всех баз данных, внесенных в каталог для этого экземпляра. Базы данных автоматически вносятся в этот каталог при выполнении команды **CREATE DATABASE**; их можно также явно внести в этот каталог при помощи команды **CATALOG DATABASE**.

Для каждой создаваемой базы данных в каталог добавляется запись, содержащая следующую информацию:

- Имя базы данных, заданное в команде **CREATE DATABASE**.
- Алиас базы данных (совпадающий с именем базы данных, если алиас не задан)
- Комментарий для базы данных, заданный в команде **CREATE DATABASE**.
- Положение `локального каталога баз данных`
- Индикатор, указывающий на то, что эта база данных является *косвенной* (то есть расположена на том же компьютере, что и этот файл системного каталога баз данных)
- Другую системную информацию.

В UNIX® в среде многораздельных баз данных все разделы базы данных должны обращаться к одному и тому же файлу системного каталога баз данных - файлу `sqldbdir`, расположенному в подкаталоге `sqldbdir` начального каталога экземпляра. Если для файла системного каталога баз или файла системных значений `sqldbins` в том же подкаталоге `sqldbdir` заданы символические ссылки на другой файл в совместно используемой файловой системе, это может вызвать непредсказуемые ошибки.

Задачи, связанные с данной темой:

- “Включение разделения данных в базе данных” на стр. 14
- “Занесение базы данных в каталог” на стр. 82

Ссылки, связанные с данной темой:

- “CREATE DATABASE Command” в *Command Reference*

Просмотр локальных файлов или файлов системного каталога баз данных

Иногда вам может потребоваться просмотреть информацию о расположенных в системе базах данных.

Предварительные требования:

Для просмотра локальных файлов или файлов системного каталога вам следует заранее создать базу данных и экземпляр.

Процедура:

Для того чтобы просмотреть содержимое локального файла каталога базы данных, вызовите следующую команду (где <положение> задает положение этой базы данных):

```
LIST DATABASE DIRECTORY ON <положение>
```

Для того чтобы просмотреть содержимое файла системного каталога баз данных, вызовите команду **LIST DATABASE DIRECTORY**, *не* указывая расположение файла.

Ссылки, связанные с данной темой:

- "LIST DATABASE DIRECTORY Command" в *Command Reference*

Каталог узла

При внесении в каталог первого раздела базы данных менеджер баз данных создает *каталог узлов*. Чтобы внести в этот каталог раздел базы данных, используйте команду **CATALOG NODE**. Чтобы вывести содержимое локального каталога узлов, используйте команду **LIST NODE DIRECTORY**. Каталог узлов создается и поддерживается для каждого клиента базы данных. Этот каталог содержит записи для всех удаленных рабочих станций, содержащих одну или несколько баз данных, к которым может обращаться этот клиент. При каждом соединении с базой данных или подключении к экземпляру клиент DB2® использует информацию из каталога узлов о конечной точке связи.

Записи в этом каталоге также содержат информацию о типе протокола связи, который должен использоваться для связи между клиентом и удаленным разделом базы данных. При внесении в этот каталог локального раздела базы данных создается алиас для экземпляра, расположенного на этом же компьютере.

Ссылки, связанные с данной темой:

- "CATALOG TCP/IP NODE Command" в *Command Reference*
- "LIST NODE DIRECTORY Command" в *Command Reference*
- "CATALOG NETBIOS NODE Command" в *Command Reference*
- "CATALOG LOCAL NODE Command" в *Command Reference*
- "CATALOG NAMED PIPE NODE Command" в *Command Reference*

Служба каталога протокола LDAP (Lightweight Directory Access Protocol)

Служба каталогов хранит информацию о ресурсах для множества систем и служб в распределенной среде и обеспечивает клиенту и серверу доступ к этим ресурсам. Клиенты и серверы используют службу каталогов для получения информации о расположении других ресурсов. В службе каталогов хранится информация о расположении других распределенных ресурсов.

Протокол LDAP (Lightweight Directory Access Protocol - протокол упрощенного доступа к каталогам) - это промышленный стандарт для метода доступа к службам каталогов. Каждый экземпляр сервера баз данных сообщает серверу LDAP о своем существовании, а при создании базы данных помещает информацию об этой базе данных в каталог LDAP. Когда клиент соединяется с базой данных, он может получить информацию о нужном сервере из каталога LDAP. При этом клиентам не нужно хранить такую информацию в локальных каталогах на каждом компьютере. Прикладные программы клиента используют каталог LDAP для поиска информации, необходимой для соединения с базой данных.

Администратор системы DB2® UDB может установить и поддерживать службу каталогов. При работе со службой каталогов можно применять Помощник по настройке и Центр управления. DB2 UDB получает доступ к службе каталогов с помощью протокола LDAP. Для применения службы каталогов LDAP необходим сервер LDAP, поддерживаемый DB2, в котором будет храниться информация о каталогах.

Примечание: В среде домена Windows® 2000 сервер LDAP существует всегда, так как он встроен в Активный каталог Windows 2000. Таким образом, LDAP может применять любой компьютер, работающий под управлением Windows 2000.

Каталоги LDAP часто применяются в больших компаниях, где поддерживать и обновлять содержимое локальных каталогов на каждом компьютере очень трудно. В таких случаях рекомендуется хранить все данные каталогов на сервере LDAP, позволяющем обновлять данные централизованно. Затраты на приобретение и обслуживание сервера LDAP могут быть довольно значительными, поэтому его целесообразно применять только в средах с большим числом клиентов.

Понятия, связанные с данным:

- “Поиск серверов администратора, экземпляров и баз данных” на стр. 64
- “Простой протокол доступа к каталогам (LDAP) - Введение” на стр. 331

Создание групп разделов баз данных (групп узлов)

Для создания группы узлов используется оператор CREATE DATABASE PARTITION GROUP. В этом операторе задается набор разделов баз данных, на которых располагаются контейнеры табличных пространств и данные таблиц. Этот оператор также:

- Создает карту разделения для этой группы узлов.
- Генерирует ID карты разделения.
- Вставляет записи в следующие таблицы каталога:
  - SYSCAT.DBPARTITIONGROUPS
  - SYSCAT.PARTITIONMAPS
  - SYSCAT.DBPARTITIONGROUPDEF

Предварительные требования для установки:

Все системы должны быть доступны и должны поддерживать многораздельную среду. Вы приобрели и установили DB2 Universal Database Enterprise - Server Edition. База данных должна существовать.

Процедура:

Для того чтобы создать раздел базы данных с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Группы разделов базы данных**.
2. Щелкните правой кнопкой мыши по папке **Группы разделов баз данных** и выберите из всплывающего меню пункт **Создать**.
3. В окне Создание групп разделов введите нужную информацию, используя кнопки со стрелками для перемещения узлов между списками **Доступные узлы** и **Выбранные разделы базы данных**, и нажмите кнопку **OK**.

Для того чтобы создать группу разделов базы данных из командной строки, введите команду:

```
CREATE DATABASE PARTITION GROUP <имя> ON PARTITIONS (<значение>,<значение>)
```

Предположим, нужно загрузить некоторые таблицы в подмножество разделов базы данных. С помощью следующей команды можно создать группу узлов из двух разделов (1 и 2) в базе данных, содержащей не менее трех (с 0 по 2) разделов:

```
CREATE DATABASE PARTITION GROUP mixng12 ON PARTITIONS (1,2)
```

Команда **CREATE DATABASE** и функция API sqlecrea() создают также группы узлов системы по умолчанию IBMDEFAULTGROUP, IBMCATGROUP и IBMTEMPGROUP.

Понятия, связанные с данным:

- “Группы разделов базы данных” в *Руководство администратора: Планирование*
- “Карты разделения” в *Руководство администратора: Планирование*

Ссылки, связанные с данной темой:

- “CREATE DATABASE PARTITION GROUP statement” в *SQL Reference, Том 2*
- “sqlecrea - Create Database” в *Administrative API Reference*
- “CREATE DATABASE Command” в *Command Reference*

Определение журнала восстановления базы данных

В *журнале восстановления базы данных* хранятся записи обо всех изменениях базы данных, включая добавление новых таблиц или изменение существующих.Этот журнал состоит из набора *экстентов журнала*, каждый из которых находится в отдельном файле (эти файлы называются *файлами журнала*).

Журнал восстановления базы данных может использоваться, чтобы в случае сбоя (например, отключения питания системы или ошибки прикладной программы) база данных не оказалась в несовместимом состоянии. В случае сбоя выполняется откат всех сделанных, но не принятых изменений, и заново выполняются все принятые транзакции, которые могли быть физически не записаны на диск. Эти действия гарантируют целостность базы данных.

Понятия, связанные с данным:

- “Understanding Recovery Logs” в *Справочное руководство по восстановлению данных и высокой доступности*

Связывание утилит с базой данных

Создав базу данных, менеджер баз данных пытается связать с ней утилиты, перечисленные в файле `db2ubind.lst`. Этот файл хранится в подкаталоге `bnd` каталога `sqllib`.

При связывании утилиты создается *пакет* - объект, содержащий всю информацию, необходимую для обработки конкретных операторов SQL из одного исходного файла.

Примечание: Чтобы использовать эти утилиты с клиента, необходимо явно выполнить их связывание.

Если нужно выполнить связывание или повторное связывание утилит с базой данных, введите в командной строке следующие команды:

```
connect to sample
bind @db2ubind.lst
```

Примечание: Для создания пакетов в базе данных `sample` нужно находиться в каталоге, где расположены эти файлы. Файлы связывания находятся в подкаталоге `bnd` каталога `sqllib`. В этом примере `sample` - это имя базы данных.

Занесение базы данных в каталог

При создании базы данных она автоматически вносится в файл системного каталога баз данных. Чтобы явно внести базу данных в файл системного каталога баз данных, можно также использовать команду **CATALOG DATABASE**. Команда **CATALOG DATABASE** позволяет внести в каталог базу данных с другим алиасом или внести в него запись для базы данных, ранее удаленную при помощи команды **UNCATALOG DATABASE**.

Предварительные требования:

Хотя базы данных и заносятся в каталог автоматически при создании базы данных, вам все же может потребоваться сделать это вручную. При этом база данных уже должна существовать.

Процедура:

Следующая команда процессора командной строки вносит в каталог базу данных `personl` под алиасом `humanres`:

```
CATALOG DATABASE personl AS humanres
   WITH "База данных персонала"
```

В этом случае в запись системного каталога баз данных вносится алиас базы данных `humanres`, отличающийся от имени базы данных (`personl`).

Можно также внести базу данных в каталог на экземпляре, отличающемся от экземпляра от умолчанию. В следующем примере соединения с базой данных `B` направляются на `INSTNC_C`. Перед вызовом этой команды экземпляр `instnc_c` уже должен быть занесен в каталог как локальный узел.

```
CATALOG DATABASE b as b_on_ic AT NODE instnc_c
```

Примечание: Команда **CATALOG DATABASE** используется также на узлах клиентов для внесения в каталог баз данных, расположенных на компьютерах серверов.

Примечание: По умолчанию файлы каталогов, включая каталог баз данных, кэшируются в памяти с помощью параметра настройки "Поддержка кэша каталогов (*dir_cache*)". Если включено кэширование каталогов, изменения, внесенные в каталог (например, при помощи команд **CATALOG DATABASE** или **UNCATALOG DATABASE**) другой прикладной программой, могут вступить в силу для данной программы только после ее перезапуска. Чтобы обновить кэш каталога, используемый сеансом процессора командной строки, используйте команду **db2 terminate**.

В среде многораздельной базы данных кэш файлов каталога создается на всех разделах базы данных.

Кроме кэша уровня прикладной программы, для внутреннего поиска менеджера баз данных используется кэш уровня менеджера баз данных. Для обновления содержимого этого "совместно используемого" кэша используйте команды **db2stop** и **db2start**.

Задачи, связанные с данной темой:

- "Добавление в каталоги информации об удаленных компьютерах сервера базы данных" на стр. 83

Ссылки, связанные с данной темой:

- "Параметр конфигурации Поддержка кэша каталогов - dir_cache" в *Руководство администратора: Производительность*
- "CATALOG DATABASE Command" в *Command Reference*
- "TERMINATE Command" в *Command Reference*
- "UNCATALOG DATABASE Command" в *Command Reference*

Добавление в каталоги информации об удаленных компьютерах сервера базы данных

Процедура:

Для создания записей каталога можно воспользоваться мастером добавления базы данных интерпретатора Ассистент конфигурирования (CA). Если в системе установлен продукт Клиент разработки программ DB2, то вы можете создать прикладную программу, добавляющую записи каталога.

Примечание: Пользователь может добавить базу данных в каталог, если у него есть полномочия SYSADM или SYSCTRL, либо если параметру конфигурации *catalog_noauth* присвоено значение YES.

Для обновления каталогов с помощью процессора командной строки выполните следующие действия:

1. Для обновления узла каталога вызовите одну из следующих команд:
   - Для узла с соединением APPC:

     ```
     db2 CATALOG APPC NODE <имя-узла>
       REMOTE <символьное-имя-целевого-компьютера> SECURITY <тип-защиты>
     ```

     Например:

     ```
     db2 CATALOG APPC NODE DB2NODE REMOTE DB2CPIC SECURITY PROGRAM
     ```

   - Для базы данных DB2 Universal Database for OS/390 and z/OS версии 5.1 (или выше) или DB2 Universal Database for AS/400 версии 4.2 (или выше) с соединением TCP/IP:

     ```
     db2 CATALOG TCPIP NODE <имя-узла>
        REMOTE <имя-хоста> или <IP-адрес>
        SERVER <имя-службы> или <номер-порта>
        SECURITY <тип-защиты>
     ```

     Например:

     ```
     db2 CATALOG TCPIP NODE MVSIPNOD REMOTE MVSHOST SERVER DB2INSTC
     ```

     В DB2 for OS/390 и z/OS порт TCP/IP по умолчанию равен 446.
2. Если применяется функция DB2 Connect, то можно обновить каталог DCS с помощью команды CATALOG DCS DATABASE.

Если с базой данных работают удаленные клиенты, дополнительно необходимо обновить каталоги на этих клиентах.

Понятия, связанные с данным:

- “Значения системного каталога баз данных” в *DB2 Connect. Руководство пользователя*
- “Значения каталога DCS” в *DB2 Connect. Руководство пользователя*

Ссылки, связанные с данной темой:

- “CATALOG DATABASE Command” в *Command Reference*
- “CATALOG DCS DATABASE Command” в *Command Reference*
- “CATALOG APPC NODE Command” в *Command Reference*
- “CATALOG TCP/IP NODE Command” в *Command Reference*
- “CATALOG NETBIOS NODE Command” в *Command Reference*
- “CATALOG APPN NODE Command” в *Command Reference*

Создание табличного пространства

Табличные пространства устанавливают связь между физическим устройствами, на которых хранятся данные, и логическими контейнерами, или таблицами.

Предварительные требования:

При создании табличного пространства вы должны знать имя устройства или файла для контейнеров, на которые вы будете ссылаться. Вам также потребуется информация о памяти на каждом устройстве или файле, которая будет выделена для табличного пространства.

Процедура:

При создании в базе данных табличного пространства назначаются контейнеры для этого табличного пространства и в системный каталог базы данных записываются определения и атрибуты этого табличного пространства. Затем в этом табличном пространстве можно создать таблицы.

Чтобы создать табличное пространство с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Табличные пространства**.
2. Щелкните правой кнопкой мыши по папке **Табличные пространства** и выберите из всплывающего меню пункт **Создать —> Табличное пространство при помощи мастера**.
3. Для завершения задания выполните шаги в мастере.

Чтобы создать табличное пространство SMS из командной строки, введите команду:

```
CREATE TABLESPACE <имя>
   MANAGED BY SYSTEM
   USING ('<путь>')
```

Чтобы создать табличное пространство DMS из командной строки, введите команду:

```
CREATE TABLESPACE <имя>
   MANAGED BY DATABASE
   USING (FILE'<путь>' <размер>)
```

Следующий оператор SQL создает в Windows табличное пространство SMS, использующее три каталога на трех отдельных дисках:

```
CREATE TABLESPACE RESOURCE
   MANAGED BY SYSTEM
   USING ('d:\acc_tbsp', 'e:\acc_tbsp', 'f:\acc_tbsp')
```

Следующий оператор SQL создает в Windows табличное пространство DMS, использующее два контейнера файлов (размером 5000 страниц каждый):

```
CREATE TABLESPACE RESOURCE
   MANAGED BY DATABASE
   USING (FILE'd:\db2data\acc_tbsp' 5000,
          FILE'e:\db2data\acc_tbsp' 5000)
```

В двух показанных выше примерах имена контейнеров заданы явно. Однако если задать для контейнеров относительные имена, контейнеры будут созданы в подкаталоге, созданном для этой базы данных.

Кроме этого, если часть заданного пути не существует, менеджер баз данных создает ее. Если подкаталог создан менеджером баз данных, этот подкаталог может также быть удален менеджером баз данных при отбрасывании табличного пространства.

В показанных выше примерах предполагается, что табличные пространства не связаны с конкретной группой разделов баз данных. Если в операторе не задан следующий параметр, используется группа разделов по умолчанию IBMDEFAULTGROUP:

```
IN группа_узлов
```

Следующий оператор SQL создает в операционной системе на основе UNIX табличное пространство DMS, использующее три логических тома (10000 страниц в каждом), и задает их характеристики ввода-вывода:

```
CREATE TABLESPACE RESOURCE
   MANAGED BY DATABASE
   USING (DEVICE '/dev/rdblv6' 10000,
          DEVICE '/dev/rdblv7' 10000,
          DEVICE '/dev/rdblv8' 10000)
   OVERHEAD 24.1
   TRANSFERRATE 0.9
```

Указанные в этом операторе SQL устройства UNIX должны уже существовать, а владелец экземпляра и группа SYSADM должны иметь возможность записывать в них данные.

В следующем примере в многораздельной базе данных в системе UNIX создается табличное пространство DMS в группе узлов ODDNODEGROUP. Группа узлов ODDNODEGROUP должна быть заранее создана при помощи оператора CREATE NODEGROUP. В этом примере предполагается, что в группу узлов ODDNODEGROUP входят разделы базы данных с номерами 1, 3 и 5. На всех разделах базы данных используется устройство `/dev/hdisk0` объемом 10000 4-Кбайтных страниц. Кроме этого для каждого раздела базы данных задается устройство объемом 40000 4-Кбайтных страниц.

```
CREATE TABLESPACE PLANS IN ODDNODEGROUP
   MANAGED BY DATABASE
   USING (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n1hd01' 40000) ON NODE 1
         (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n3hd03' 40000) ON NODE 3
         (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n5hd05' 40000) ON NODE 5
```

В системе UNIX существуют устройства двух типов: последовательные символьные устройства и блочные устройства. Для каждого блочного устройства (или устройства *с обработкой*) файловой системы есть соответствующее последовательное символьное устройство (или *непосредственное* устройство). Блочным устройствам обычно присваиваются имена, подобные “hd0” или “fd0”. Последовательным символьным устройствам обычно присваиваются имена, подобные “rhd0”, “rfd0” или “rmt0”. Эти последовательные символьные устройства имеют большую скорость доступа, чем блочные устройства. В команде CREATE TABLESPACE следует использовать имена последовательных символьных устройств, а не имена блочных устройств.

При компиляции оператора SQL для определения наилучшего пути доступа используется информация о затратах и скорости передачи.

DB2 может значительно улучшить производительность последовательного ввода-вывода при помощи последовательной предварительной выборки, которая использует параллельный ввод-вывод.

Можно также создать табличное пространство, использующее страницы, размер которых больше размера по умолчанию (4 Кбайта). Следующий оператор SQL создает в системе на основе UNIX табличное пространство SMS с размером страниц 8 Кбайт.

```
CREATE TABLESPACE SMS8K
   PAGESIZE 8192
   MANAGED BY SYSTEM
   USING ('FSMS_8K_1')
   BUFFERPOOL BUFFPOOL8K
```

Обратите внимание на то, что связанный пул буферов тоже должен иметь размер страницы 8 Кбайт.

Созданное табличное пространство можно использовать только после того, как будет активирован заданный для него пул буферов.

Для добавления, удаления или изменения контейнеров в табличном пространстве DMS или изменения параметров PREFETCHSIZE, OVERHEAD и TRANSFERRATE для табличного пространства можно использовать оператор SQL ALTER TABLESPACE. Следует как можно быстрее выполнить принятие транзакции, в которой выполнен этот оператор, чтобы предотвратить конфликты системного каталога.

Примечание: Значение PREFETCHSIZE должно быть кратно значению EXTENTSIZE. Например, если значение EXTENTSIZE равно 10, параметр PREFETCHSIZE может иметь значение 20 или 30.

Понятия, связанные с данным:

- “Проектирование табличного пространства” в *Руководство администратора: Планирование*
- “Пространство, управляемое системой” в *Руководство администратора: Планирование*
- “Пространство, управляемое базой данных” в *Руководство администратора: Планирование*
- “Последовательная предварительная выборка” в *Руководство администратора: Производительность*
- “Табличные пространства” в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- “ALTER TABLESPACE statement” в *SQL Reference, Том 2*
- “CREATE TABLESPACE statement” в *SQL Reference, Том 2*

Создание табличных пространств различных типов

Менеджер баз данных, приложения и пользователи могут работать с табличными пространствами нескольких типов.

Создание временного системного табличного пространства

Несмотря на то, что временное системное табличное пространство создается при создании базы данных автоматически, для работы с системными задачами сортировки вам может потребоваться создать отдельное пространство.

Предварительные требования:

Контейнеры, связанные с временным системным табличным пространством, уже должны существовать.

Ограничения:

В базе данных всегда должно быть по крайней мере одно системное временное табличное пространство, поскольку системные временные таблицы могут храниться только в таком табличном пространстве.

Процедура:

Системное временное табличное пространство используется для хранения системных временных таблиц. Одно из трех определяемых при создании базы данных табличных пространств по умолчанию - это системное временное табличное пространство с именем “TEMPSPACE1”.

Для создания другого системного временного табличного пространства можно использовать оператор CREATE TABLESPACE. Например,

```
CREATE SYSTEM TEMPORARY TABLESPACE tmp_tbsp
   MANAGED BY SYSTEM
   USING ('d:\tmp_tbsp','e:\tmp_tbsp')
```

Вам потребуется по крайней мере одно табличное пространство для каждого размера страницы.

При создании системного временного табличного пространства можно задавать только группу разделов базы данных IBMTEMPGROUP.

Задачи, связанные с данной темой:

- “Создание временного пользовательского табличного пространства” на стр. 89

Ссылки, связанные с данной темой:

- “CREATE TABLESPACE statement” в *SQL Reference, Том 2*

Создание временного пользовательского табличного пространства

Пользовательские временные табличные пространства не создаются по умолчанию при создании базы данных. Однако при работе с базой данных некоторым программам могут потребоваться временные таблицы. В этом случае создайте пользовательскую временную таблицу.

Процедура:

Пользовательское временное табличное пространство используется для хранения объявленных временных таблиц.

Для создания пользовательского временного табличного пространства можно использовать оператор CREATE TABLESPACE:

```
CREATE USER TEMPORARY TABLESPACE usr_tbsp
   MANAGED BY DATABASE
   USING (FILE 'd:\db2data\user_tbsp' 5000,
          FILE 'e:\db2data\user_tbsp' 5000)
```

Как и другие табличные пространства, пользовательские временные табличные пространства можно создавать в любых группах разделов базы данных, кроме

IBMTEMPGROUP. По умолчанию при создании пользовательского временного табличного пространства используется группа разделов IBMDEFAULTGROUP.

Оператор DECLARE GLOBAL TEMPORARY TABLE определяет объявленную временную таблицу, используемую внутри пользовательского временного табличного пространства.

Ссылки, связанные с данной темой:

- "CREATE TABLESPACE statement" в *SQL Reference, Том 2*
- "DECLARE GLOBAL TEMPORARY TABLE statement" в *SQL Reference, Том 2*

Создание табличных пространств в группах разделов базы данных

При размещении табличного пространства в группе узлов с несколькими разделами базы данных все таблицы в этом табличном пространстве распределяются по всем разделам базы данных в этой группе разделов. Табличное пространство создается в группе разделов. После того, как табличное пространство было определено в конкретной группе разделов, оно должно оставаться в этой группе - его нельзя перевести в другую группу разделов. Для задания группы разделов базы данных для табличного пространства используется оператор CREATE TABLESPACE.

Ссылки, связанные с данной темой:

- "CREATE TABLESPACE statement" в *SQL Reference, Том 2*

Использование прямого ввода-вывода

При работе с контейнерами DB2 Universal Database поддерживает прямой доступ к дискам (непосредственный ввод-вывод). Это позволяет подключать к любой системе DB2 Universal Database устройства прямого доступа к дискам (непосредственные устройства). (Единственным исключением является операционная система Windows 9x.)

Предварительные требования:

При создании табличного пространства вы должны знать имя устройства или файла для контейнеров, на которые вы будете ссылаться. Вам также потребуется информация о памяти на каждом устройстве или файле, которая будет выделена для табличного пространства.

Для чтения и записи данных в контейнер необходимы особые права доступа.

Процедура:

В следующем списке показаны физические и логические методы определения устройств с прямым доступом:

- В Windows для задания физического жесткого диска используйте следующий синтаксис:

  \\.\PhysicalDriveN

  где N - номер одного из физических дисков в системе. В данном случае вместо N может быть задано 0, 1, 2 или любое другое положительное целое число:

  \\.\PhysicalDrive5
- В Windows для задания логического диска (то есть неформатированного раздела) используйте следующий синтаксис:

  \\.\N:

  где N: - задает букву логического диска в системе. Например, вместо N: можно задать E: или любую другую букву диска. Вы можете избежать ограничения, появляющегося при использовании буквы диска, указав глобальный идентификатор (GUID) логического диска.
- **Примечание:** Чтобы была возможность записывать на устройство, должна быть установлена операционная система Windows NT Версии 4.0 с Service Pack 3 или более поздним.
- В системах на основе UNIX используйте символьное имя последовательного устройства, например `/dev/rhd0`

В Windows 2000 и более поздних версиях применяется новый способ указания контейнеров табличных пространств с прямым вводом-выводом. Каждому тому (то есть, основному разделу дисков и динамическому тому) при создании присваивается глобальный идентификатор (GUID). Этот GUID можно использовать как идентификатор устройства при задании контейнеров в определении табличного пространства. GUID не повторяются в разных системах, поэтому GUID разделов будут разными, даже если определения дисковых разделов совпадают.

Инструмент *db2listvolumes.exe* помогает выводить GUID всех томов дисков, определенных в системе Windows. Этот инструмент создает два файла в каталоге, в котором его запустили. В одном файле - `volumes.xml` - находится информация о каждом томе диска в формате XML. В другом файле - `tablespace.ddl` - находится необходимый синтаксис для задания контейнеров табличных пространств. Этот файл необходимо обновлять для внесения в него сведений, требуемых для определения табличных пространств. Инструмент *db2listvolumes* не требует аргументов командной строки.

Задачи, связанные с данной темой:

- “Настройка прямого ввода-вывода в Linux” на стр. 92

Настройка прямого ввода-вывода в Linux

При работе с контейнерами DB2 Universal Database поддерживает прямой доступ к дискам (непосредственный ввод-вывод). Это позволяет подключать к любой системе DB2 Universal Database устройства прямого доступа к дискам (непосредственные устройства). В среде Linux с этой возможностью связаны особые условия.

Предварительные требования:

При создании табличного пространства вы должны знать имя устройства или файла для контейнеров, на которые вы будете ссылаться. Вам также потребуется информация о памяти на каждом устройстве или файле, которая будет выделена для табличного пространства.

Прежде чем установить в Linux непосредственный ввод-вывод, вам понадобятся:

- Один или несколько свободных разделов диска IDE или SCSI
- Контроллер непосредственных устройств под именем `/dev/rawctl` или `/dev/raw`. Если используется другое имя, создайте символическую ссылку:

  ```
  # ln -s /dev/your_raw_dev_ctrl /dev/rawctl
  ```
- Утилита raw, которая обычно поставляется с дистрибутивом Linux

Примечание: В распространяемых в настоящее время дистрибутивах, поддерживающих непосредственный ввод-вывод, узлы непосредственных устройств называются по-разному:

Таблица 3. Дистрибутивы Linux, поддерживающие непосредственный ввод-вывод

Дистрибутив	Узлы с непосредственным доступом	Контроллер с непосредственным доступом
RedHat или TurboLinux	`/dev/raw/raw1 - 255`	`/dev/rawctl`
SuSE	`/dev/raw1 - 63`	`/dev/raw`

DB2 поддерживает все вышеперечисленные контроллеры непосредственных устройств и большинство других названий для узлов непосредственных устройств. DB2 не поддерживает непосредственных устройств в Linux/390.

Процедура:

В Linux есть пул узлов с устройствами непосредственного ввода-вывода, который нужно связать с блочным устройством, прежде чем выполнять операции непосредственного ввода-вывода на нем. Информация связывания непосредственных устройств с блочными устройствами хранится на контроллере

непосредственных устройств. Связывание выполняется при помощи утилиты под именем raw, которая обычно поставляется в дистрибутиве Linux.

Чтобы сконфигурировать непосредственный ввод-вывод в Linux:

В этом примере используемый раздел raw называется /dev/sda5. В нем не должно быть никакой существенной информации.

Шаг 1. Вычислите количество страниц по 4096 байт, округляя, при необходимости, в меньшую сторону. Например:

```
# fdisk /dev/sda
Command (m for help): p

Disk /dev/sda: 255 heads, 63 sectors, 1106 cylinders
Units = cylinders of 16065 * 512 bytes
```

Таблица 4. Сведения о непосредственном вводе-выводе в Linux

Загрузка устройств	Запуск	Завершение	Блоки	ID	Система
/dev/sda1	1	523	4200997	83	Linux
/dev/sda2	524	1106	4682947+	5	Extended
/dev/sda5	524	1106	4682947	83	Linux

```
Command (m for help): q
#
```

Число страниц в /dev/sda5 равно

```
num_pages = floor( ((1106-524+1)*16065*512)/4096 )
num_pages = 11170736
```

Шаг 2. Свяжите с этим разделом неиспользуемый узел непосредственного устройства. Это необходимо делать при каждой перезагрузке компьютера; для этого нужен доступ с полномочиями root. Чтобы посмотреть, какие узлы непосредственных устройств уже используются, введите команду raw -a:

```
# raw /dev/raw/raw1 /dev/sda5
/dev/raw/raw1: bound to major 8, minor 5
```

Шаг 3. Задайте необходимые права чтения на контроллере непосредственного устройства и разделе диска. Задайте необходимые права чтения и записи на непосредственном устройстве:

Шаг 4. Создайте табличное пространство в DB2, выбрав непосредственное устройство, а не раздел диска. Например:

```
CREATE TABLESPACE dms1
MANAGED BY DATABASE
USING (DEVICE '/dev/raw/raw1' 11170736)
```

Табличные пространства на непосредственных устройствах поддерживаются также для всех остальных размеров страниц, поддерживаемых DB2.

Задачи, связанные с данной темой:

- “Использование прямого ввода-вывода” на стр. 90

Создание схемы

Данные организуются в виде таблиц, но может быть удобно группировать таблицы и другие связанные объекты вместе. Для этого при помощи оператора CREATE SCHEMA определяется схема. Информация о схеме хранится в таблицах системного каталога базы данных, с которой установлено соединение. Созданные объекты можно поместить в эту схему.

Предварительные требования для установки:

Таблицы базы данных и связанные с ней объекты должны существовать.

Ограничения:

Для использования этого оператора пользователь должен обладать полномочиями DBADM.

Схемы могут также создаваться неявно, если пользователь обладает полномочиями IMPLICIT_SCHEMA. Для пользователей с этими полномочиями при создании объекта, для которого задано имя несуществующей схемы, эта схема создается неявно.

Если пользователь не обладает полномочиями IMPLICIT_SCHEMA, он может создать только схему с именем, совпадающим с его ID авторизации.

Неавторизованный доступ к объектам внутри схемы не допускается, поскольку схема используется для обеспечения уникальности в базе данных. Это становится понятно, если представить себе возможность создания двумя пользователями двух таблиц (или других объектов) с одинаковыми именами. При отсутствии схемы, обеспечивающей уникальность, если к такой таблице попытается обратиться третий пользователь, возникла бы неоднозначность Без дополнительной идентификации будет невозможно определить, какую именно таблицу использовать.

Задаваемое для новой схемы имя не должно существовать в системных каталогах и не может начинаться с ″SYS″.

Процедура:

Пользователь с полномочиями SYSADM или DBADM может создавать схемы с любыми допустимыми именами. При создании базы данных полномочия IMPLICIT_SCHEMA предоставляются пользователям группы PUBLIC (то есть всем).

Пользователь, определивший какие-либо объекты в операторе CREATE SCHEMA, является владельцем этой схемы. Он может предоставлять (GRANT) другим пользователям и отзывать (REVOKE) у них привилегии для этой схемы.

Чтобы позволить другому пользователю, обращаясь к таблице по имени, не вводить имя схемы, нужно задать для этого пользователя производную таблицу. В определении производной таблицы должно быть указано полное имя таблицы, включая и схему пользователя; тогда пользователю в запросе нужно просто указать имя производной таблицы. Спецификацию производной таблицы нужно сделать полной, обязательно указав в ее определении схему данного пользователя.

Чтобы создать схему с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Схема**.
2. Щелкните правой кнопкой мыши по папке **Схема** и выберите из всплывающего меню пункт **Создать**.
3. Введите информацию для новой схемы и нажмите кнопку **OK**.

Чтобы создать схему из командной строки, введите команду:

```
CREATE SCHEMA <имя> AUTHORIZATION <имя>
```

В следующем примере оператор CREATE SCHEMA создает схему для отдельного пользователя с ID авторизации ″joe″:

```
CREATE SCHEMA joeschma AUTHORIZATION joe
```

Понятия, связанные с данным:

- “Группировка объектов по схеме” на стр. 8
- “Особенности полномочий неявного задания схемы (IMPLICIT_SCHEMA)” на стр. 251
- “Привилегии схем” на стр. 252

Задачи, связанные с данной темой:

- “Настройка схемы” на стр. 96

Ссылки, связанные с данной темой:

- “CREATE SCHEMA statement” в *SQL Reference, Том 2*

Сведения о создании схемы

Схемы применяются для управления принадлежностью объектов в базе данных.

Настройка схемы

Создав несколько схем, вы можете задать схему по умолчанию, используемую для заданных без спецификаторов имен объектов в динамических операторах SQL в конкретном соединении DB2.

Процедура:

Специальному регистру CURRENT SCHEMA присваивается имя схемы, которая должна использоваться по умолчанию. Любой пользователь может задать значение этого специального регистра - для этого не требуются никакие особые полномочия.

Ниже показан пример задания специального регистра CURRENT SCHEMA:

```
SET CURRENT SCHEMA = 'SCHEMA01'
```

Этот оператор можно использовать внутри прикладной программы или ввести в интерактивном режиме. Когда задано значение специального регистра CURRENT SCHEMA, оно используется в качестве спецификатора (имени схемы) для имен объектов, для которых не задана схема, в динамических операторах SQL, за исключением оператора CREATE SCHEMA, в котором используется ссылка без спецификатора на существующий объект базы данных.

Начальное значение специального регистра CURRENT SCHEMA - ID авторизации пользователя текущего сеанса.

Понятия, связанные с данным:

- “Schemas” в *SQL Reference, Том 1*

Ссылки, связанные с данной темой:

- “SET SCHEMA statement” в *SQL Reference, Том 2*
- “Reserved schema names and reserved words” в *SQL Reference, Том 1*
- “CURRENT SCHEMA special register” в *SQL Reference, Том 1*

Создание и заполнение таблицы

Таблицы - это основные хранилища информации в базе данных. Таблицы создаются и заполняются данными при создании базы данных.

Предварительные требования:

Формат и структуру таблиц следует продумать заранее.

Процедура:

Решив, как нужно организовать данные в таблицы, создайте эти таблицы при помощи оператора CREATE TABLE. Описания таблиц сохраняются в системном каталоге базы данных, с которой установлено соединение.

Оператор CREATE TABLE задает для таблицы имя (идентификатор со спецификаторами или без) и определения для каждого из ее столбцов. Можно хранить каждую таблицу в отдельном табличном пространстве, чтобы в табличном пространстве была только одна таблица. Если таблица будет часто отбрасываться и создаваться, лучше хранить ее в отдельном табличном пространстве и отбрасывать это табличное пространство, а не таблицу. Можно также хранить несколько таблиц в одном табличном пространстве. В среде многораздельных баз данных выбираемое табличное пространство задает также группу разделов и разделы базы данных, на которых хранятся данные таблицы.

При создании таблица не содержит данных. Чтобы добавить в нее строки данных, используйте один из следующих методов:

- Оператор INSERT
- Команды LOAD или IMPORT
- Утилиту autoloader при работе в среде многораздельной базы данных

При добавлении данных в таблицу информация об изменениях может не записываться в журнал. Условие NOT LOGGED INITIALLY оператора CREATE TABLE предотвращает сохранение в журнал информации об изменениях для этой таблицы. В журнал не будет записываться информация о всех изменениях, сделанных в этой таблице операторами INSERT, DELETE, UPDATE, CREATE INDEX, DROP INDEX или ALTER TABLE в той же единице работы, в которой создана таблица. Запись в журнал начнется в следующих единицах работы.

Таблица состоит из одного или нескольких определений столбцов. Для таблицы можно определить максимум 500 столбцов. Столбцы представляют собой атрибуты записи. Все значения в одном столбце относятся к одному типу данных.

Примечание: Максимальное число столбцов при использовании страниц размером 4 Кбайта - 500. При использовании страниц размером 8 Кбайт, 16 Кбайт или 32 Кбайта максимальное число столбцов - 1012.

Определение столбца включает в себя *имя столбца, тип данных*, а также при необходимости *атрибут пустых значений* или значение по умолчанию (необязательно, по выбору пользователя).

Имя столбца описывает информацию, содержащуюся в этом столбце, поэтому столбцам следует давать содержательные имена. Имя столбца должно быть уникальным в таблице; однако то же имя может использоваться в других таблицах.

Тип данных столбца определяет длину его значения и тип данных, допустимых для этого столбца. Менеджер баз данных использует типы данных: символьная строка, числовое значение, дата, время и большой объект. Тип данных графическая строка разрешены только в средах баз данных, использующих наборы многобайтных символов. Кроме этого, столбцы могут быть определены как столбцы особых пользовательских типов данных.

Атрибут значения по умолчанию определяет значение, которое должно использоваться в случаях, когда не задано конкретное значение. Можно задать значение по умолчанию или же использовать значение по умолчанию, определенное системой. Значения по умолчанию можно задать как для столбцов, для которых задан атрибут пустого значения, так и для столбцов, для которых он не задан.

Атрибут допустимости пустых значений определяет, может ли столбец содержать пустые значения.

Чтобы создать таблицу с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой мыши по папке **Таблицы** и выберите из всплывающего меню пункт **Создать —> Таблицы при помощи мастера**.
3. Для завершения задания выполните шаги в мастере.

Чтобы создать таблицу из командной строки, введите команду:

```
CREATE TABLE <имя>
   (<имя_столбца>  <тип_данных> <атрибут_пустых_значений>)
   IN <имя_табличного_пространства)
```

Ниже показан пример оператора CREATE TABLE, создающего таблицу EMPLOYEE в табличном пространстве RESOURCE. Это таблица определяется в базе данных sample:

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME  VARCHAR(12)                     NOT NULL,
    MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
    LASTNAME  VARCHAR(15)                     NOT NULL,
    WORKDEPT  CHAR(3),
    PHONENO   CHAR(4),
    PHOTO     BLOB(10M)   NOT NULL)
IN RESOURCE
```

При создании таблицы можно задать, чтобы столбцы таблицы были созданы на основе атрибутов структурированного типа. Такая таблица называются “типизированной таблицей”.

Типизированную таблицу можно определить так, чтобы она наследовала некоторые свои столбцы от другой типизированной таблицы. Такая таблица называется “подтаблицей”, а таблица, столбцы которой она наследует, называется “надтаблицей”. Комбинация типизированной таблицы и всех ее подтаблиц называется “иерархией таблиц”. Самая верхняя таблица в иерархии таблиц (та, которая которая не имеет надтаблицы) называется “корневой таблицей” иерархии.

Для объявления глобальной временной таблицы предназначен оператор DECLARE GLOBAL TEMPORARY TABLE.

Можно также создать таблицу, определенную на основе результата запроса. Такая таблица называется *материализованной таблицей запроса*.

Понятия, связанные с данным:

- “Import Overview” в *Data Movement Utilities Guide and Reference*
- “Load Overview” в *Data Movement Utilities Guide and Reference*
- “Moving Data Across Platforms - File Format Considerations” в *Data Movement Utilities Guide and Reference*
- “Пользовательский тип (UDT)” на стр. 134

Задачи, связанные с данной темой:

- “Создание материализованной таблицы запроса” на стр. 141

Ссылки, связанные с данной темой:

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

- “IMPORT Command” в *Command Reference*
- “LOAD Command” в *Command Reference*

Сведения о создании и заполнении таблицы

Все ваши данные хранятся в таблицах. При создании таблиц и заполнении их данными следует учитывать множество факторов.

Сжатие таблиц - Введение

Пространство, занимаемое таблицами на диске, можно сократить в двух случаях:

- Если значение столбца равно NULL, не указывайте заданный фиксированный объем памяти.
- Если значение столбца можно легко определить (например, если оно совпадает со значением по умолчанию), и если это значение доступно менеджеру баз данных во время форматирования записи и извлечения столбца.

В DB2® UDB предусмотрен особый формат записи, позволяющий сократить объем занимаемой памяти в описанных случаях. Можно сократить как размер таблицы, так и размер столбца.

Понятия, связанные с данным:

- “Размер объектов базы данных” в *Руководство администратора: Планирование*
- “Сжатие новых таблиц” на стр. 100
- “Сжатие существующих таблиц” на стр. 191

Сжатие новых таблиц

При создании таблицы можно задать условие VALUE COMPRESSION. В этом случае на уровне таблицы и, возможно, на уровне столбца будет применяться сжатый формат строк.

Если задано условие VALUE COMPRESSION, то пустые значения и значения нулевой длины, присвоенные столбцам с типом данных переменной длины (VARCHAR, VARGRAPHICS, LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB и DBCLOB), не сохраняются на диске. На диске сохраняются только значимые данные этого типа.

Вместе с условием VALUE COMPRESSION можно задать необязательную опцию COMPRESS SYSTEM DEFAULT, позволяющую еще больше сократить объем занимаемой памяти на диске. Для хранения вставленных и обновленных значений, совпадающих с системным значением по умолчанию для типа данных столбца, затрачивается минимальный объем дисковой памяти. Значение по

умолчанию не сохраняется на диске. Опцию COMPRESS SYSTEM DEFAULT поддерживают все столбцы с числовым типом данных, символьные столбцы фиксированной длины и столбцы с графическими строками фиксированной длины. Это означает, что нулевые значения и пробелы не будут сохраняться при сжатии.

Ссылки, связанные с данной темой:

- "CREATE TABLE statement" в *SQL Reference, Том 2*

Сведения о столбцах больших объектов (LOB)

Перед созданием таблицы, содержащей столбцы больших объектов, нужно решить:

1. Хотите ли вы, чтобы в журнал записывалась информация об изменениях для столбцов больших объектов?

   Если не нужно записывать в журнал эту информацию об изменениях, необходимо отключить запись в журнал, задав при создании таблицы условие NOT LOGGED:

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME  VARCHAR(12)                     NOT NULL,
    MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
    LASTNAME  VARCHAR(15)                     NOT NULL,
    WORKDEPT  CHAR(3),
    PHONENO   CHAR(4),
    PHOTO     BLOB(10M)   NOT NULL  NOT LOGGED)
IN RESOURCE
```

   Если размер столбца больших объектов больше 1 Гбайта, запись в журнал должна быть отключена. (Рекомендуется не записывать в журнал информацию об изменениях для столбцов больших объектов, больших 10 Мбайт.) Как и для других опций, заданных в определении таблицы, изменить значение опции записи в журнал можно, только заново создав эту таблицу.

   Даже если не задана запись в журнал информации об изменениях, для столбцов больших объектов будут сохраняться *теневые копии*, позволяющие выполнить откат изменений (откат может быть вызван системной ошибкой или требованием прикладной программы). Сохранение теневых копий - это техника восстановления, при которой содержимое текущих страниц хранения никогда не перезаписывается, то есть старые, неизмененные страницы сохраняются как "теневые" копии. Эти копии отбрасываются, когда они более не нужны для поддержки отката транзакции.

   Примечание: При восстановлении базы данных с помощью команд RESTORE и ROLLFORWARD данные больших объектов, информация об изменениях которых не была записана в журнал (столбцы которых определены как "NOT LOGGED") и

которые были записаны после последнего резервного копирования, будут *замещены двоичными нулями*.

2. Хотите ли вы минимизировать объем, необходимый для столбца больших объектов?

   Можно уменьшить, насколько это возможно, занимаемое столбцом больших объектов пространство, используя условие COMPACT оператора CREATE TABLE. Например:

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME  VARCHAR(12)                    NOT NULL,
    MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
    LASTNAME  VARCHAR(15)                    NOT NULL,
    WORKDEPT  CHAR(3),
    PHONENO   CHAR(4),
    PHOTO     BLOB(10M)   NOT NULL  NOT LOGGED  COMPACT)
IN RESOURCE
```

   При этом возможны некоторые *потери производительности* при добавлении данные в таблицу с компактными столбцами больших объектов, особенно если увеличивается размер значений больших объектов (поскольку придется изменять их хранение).

   На платформах, где не поддерживается фрагментарное выделение файлов и большие объекты размещаются в табличных пространствах SMS, возможно, следует использовать условие COMPACT. Фрагментарное выделение файлов определяет тип использования операционной системой физического дискового пространства. В операционных системах, поддерживающих фрагментарное выделение файлов, для хранения больших объектов используется меньший объем физического дискового пространства, чем в операционных системах, не поддерживающих фрагментарное выделение файлов. Даже при поддержке фрагментарного выделения файлов опция COMPACT позволяет увеличить “экономию” физического дискового пространства. Поскольку опция COMPACT может дать некоторую экономию физического дискового пространства, использование этой опции может быть выгодно, если операционная система не поддерживает фрагментарное выделение файлов.

   Примечание: Системные каталоги DB2® используют столбцы больших объектов и могут занимать больший объем пространства, чем в предыдущих версиях.

3. Хотите ли вы улучшить производительность работы со столбцами больших объектов, включив их в системные каталоги DB2?

   В таблицах каталога есть столбцы больших объектов. Данные больших объектов не хранятся в пуле буферов вместе с другими данными, при каждом обращении к таким данным они считываются с диска. Операции чтения с диска ухудшают производительность DB2 при работе со столбцами больших

объектов каталогов. Поскольку файловая система обычно имеет собственное место временного хранения (кэширования) данных, при использовании табличного пространства SMS или табличного пространства DMS, созданного на основе контейнеров файлов, можно избежать повторных операций чтения, если большой объект уже был ранее прочитан.

Понятия, связанные с данным:

- "Размер данных больших объектов" в *Руководство администратора: Планирование*

Ссылки, связанные с данной темой:

- "CREATE TABLE statement" в *SQL Reference, Том 2*
- "Large objects (LOBs)" в *SQL Reference, Том 1*

Определение ограничений

В этом разделе описывается, как определить ограничения:

- "Определение ограничения уникальности"
- "Определение реляционных ограничений" на стр. 105
- "Определение проверочного ограничения таблицы" на стр. 109
- "Определение информационного ограничения" на стр. 110.

Дополнительную информацию об ограничениях смотрите в разделе, посвященном планированию ограничений, в руководстве *Руководство администратора: Планирование*, а также в руководстве *SQL Reference*.

Определение ограничения уникальности

Ограничение уникальности обеспечивает уникальность значений указанного ключа. Таблица может содержать любое число ограничений уникальности, но не больше одного ограничения уникальности, определенного в качестве первичного ключа.

Ограничения:

Нельзя определить ограничение уникальности для подтаблицы.

У таблицы может быть только один первичный ключ.

Процедура:

Для определения ограничения уникальности используется условие UNIQUE операторов CREATE TABLE или ALTER TABLE. Уникальный ключ может состоять из нескольких столбцов. Для таблицы можно определить несколько ограничений уникальности.

После того, как ограничение уникальности задано, оно автоматически применяется менеджером баз данных, когда операторы INSERT или UPDATE изменяют данные в этой таблице. Ограничение уникальности обеспечивается при помощи индекса уникальности.

Если ограничение уникальности определено в операторе ALTER TABLE и существует индекс на том же наборе столбцов, что и ключ уникальности, этот индекс становится индексом уникальности и используется этим ограничением.

Одно из ограничений уникальности можно использовать в качестве *первичного ключа*. Первичный ключ может использоваться в качестве родительского ключа в реляционном ограничении (вместе с другими ограничениями уникальности). Первичный ключ определяется условием PRIMARY KEY оператора CREATE TABLE или ALTER TABLE. Первичный ключ может состоять из нескольких столбцов.

Первичный индекс требует уникальности значений первичного ключа. Если таблица создается с первичным ключом, менеджер баз данных создает для этого ключа первичный индекс.

Некоторые советы по улучшению производительности для индексов, использующихся в качестве ограничений уникальности:

- При выполнении исходной загрузки данных в пустую таблицу с индексами команда LOAD дает лучшую производительность, чем IMPORT. При этом неважно, какой из режимов команды LOAD вы используете - INSERT или REPLACE.
- При добавлении значительного объема данных в существующую таблицу с индексами (с помощью IMPORT INSERT или LOAD INSERT), команда LOAD дает немного лучшую производительность, чем IMPORT.
- Если для исходной загрузки большого объема данных используется команда IMPORT, создайте ключ уникальности после импорта или загрузки данных. Это позволит избежать дополнительных затрат на поддержку индекса во время загрузки таблицы. Это также уменьшит используемый индексом объем хранения.
- Если утилита загрузки используется в режиме REPLACE, создайте ключ уникальности до загрузки данных. В этом случае создание индекса во время загрузки более эффективно, чем использование оператора CREATE INDEX после загрузки.

Понятия, связанные с данным:

- “Keys” в *SQL Reference, Том 1*
- “Constraints” в *SQL Reference, Том 1*

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*
- “CREATE TABLE statement” в *SQL Reference, Том 2*

Определение реляционных ограничений

Для обеспечения реляционной целостности в определения таблицы и столбцов добавляются реляционные ограничения. После создания реляционного ограничения в менеджере базы данных все изменения данных будут проверены на соответствие этому ограничению. Возможность завершения действия будет зависеть от результата проверки ограничения.

Процедура:

Для задания реляционных ограничений используются условия FOREIGN KEY и REFERENCES операторов CREATE TABLE или ALTER TABLE. Перед созданием реляционного ограничения следует учесть его влияние на типизированные таблицы.

Когда заданы внешние ключи, накладываются ограничения на значения в строках таблицы или двух таблиц. Менеджер баз данных проверяет ограничения, заданные в определении таблицы, и в соответствии с ними поддерживает отношения между значениями. Целью является поддержание целостности в случаях, когда один объект базы данных ссылается на другой.

Например, пусть и первичный, и внешний ключи содержат столбец с номером отдела фирмы. В таблице сотрудников EMPLOYEE этот столбец называется WORKDEPT (отдел), а в таблице отделов DEPARTMENT - DEPTNO (номер отдела). Связь между двумя этими таблицами задается следующими ограничениями:

- Для каждого работника в таблице EMPLOYEE существует только один номер отдела и этот номер отдела существует в таблице DEPARTMENT.
- Каждая строка таблицы EMPLOYEE связана не более чем с одной строкой таблицы DEPARTMENT. Между таблицами существует отношение.
- Каждая строка таблицы EMPLOYEE, содержащая непустое значение WORKDEPT, связана со строкой с столбце DEPTNO таблицы DEPARTMENT.
- Таблица DEPARTMENT является родительской таблицей, а таблица EMPLOYEE - зависимой таблицей.

Оператор SQL, определяющий родительскую таблицу DEPARTMENT:

```
CREATE TABLE DEPARTMENT
   (DEPTNO    CHAR(3)           NOT NULL,
    DEPTNAME  VARCHAR(29) NOT NULL,
    MGRNO     CHAR(6),
```

```
    ADMRDEPT  CHAR(3)           NOT NULL,
    LOCATION  CHAR(16)                  ,
       PRIMARY KEY (DEPTNO)               )
IN RESOURCE
```

Оператор SQL, определяющий зависимую таблицу EMPLOYEE:

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME  VARCHAR(12)                      NOT NULL,
    LASTNAME  VARCHAR(15)                      NOT NULL,
    WORKDEPT  CHAR(3),
    PHONENO   CHAR(4),
    PHOTO     BLOB(10m)   NOT NULL,
       FOREIGN KEY DEPT (WORKDEPT)
       REFERENCES DEPARTMENT ON DELETE NO ACTION)
IN RESOURCE
```

Задав столбец DEPTNO в качестве первичного ключа таблицы DEPARTMENT и столбец WORKDEPT в качестве внешнего ключа таблицы EMPLOYEE, вы определяете реляционное ограничение на значения WORKDEPT. Это ограничение обеспечивает реляционную целостность значений в этих двух таблицах. В данном случае для всех работников, добавляемых в таблицу EMPLOYEE, должны задаваться номера отделов, существующие в таблице DEPARTMENT.

Для этого реляционного ограничения в таблице EMPLOYEE задано правило удаления NO ACTION, что означает, что отдел не будет удаляться из таблицы DEPARTMENT, если в этом отделе есть сотрудники.

Хотя в предыдущих примерах для добавления реляционного ограничения использовался оператор CREATE TABLE, для этого можно также использовать оператор ALTER TABLE.

Другой пример: Используются те же определения таблиц, что и в предыдущем примере. Таблица DEPARTMENT создается раньше таблицы EMPLOYEE. Каждый отдел имеет руководителя и эти руководители указаны в таблице EMPLOYEE. Столбец MGRNO (номер руководителя) в таблице DEPARTMENT используется как внешний ключ таблицы EMPLOYEE. Реляционные связи образуют цикл, что порождает некоторые проблемы с этим ограничением. Внешний ключ можно добавить позднее. Можно также использовать оператор CREATE SCHEMA, чтобы создать обе таблицы EMPLOYEE и DEPARTMENT одновременно.

Понятия, связанные с данным:

- “Условие FOREIGN KEY” на стр. 107
- “Условие REFERENCES” на стр. 108

Задачи, связанные с данной темой:

- “Добавление внешних ключей” на стр. 198

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*
- “CREATE SCHEMA statement” в *SQL Reference, Том 2*
- “CREATE TABLE statement” в *SQL Reference, Том 2*

Условие FOREIGN KEY

Внешний ключ ссылается на первичный ключ или ключ уникальности в той же или другой таблице. Задание внешнего ключа указывает, что реляционная целостность должна поддерживаться в соответствии с заданными реляционными ограничениями. Для определения внешнего ключа используется условие FOREIGN KEY оператора CREATE TABLE или ALTER TABLE.

Число столбцов внешнего ключа должно совпадать с числом столбцов соответствующего первичного или уникального ограничения (называемого родительским ключом) родительской таблицы. Кроме того, соответствующие компоненты определений столбцов ключей должны иметь совпадающие типы данных и длины. Внешнему ключу можно присвоить *имя ограничения*. Если это имя не задано, оно будет присвоено автоматически. Для упрощения работы рекомендуется задавать *имя ограничения*, а не использовать имя, сгенерированное системой.

Значение составного внешнего ключа совпадает со значением родительского ключа, **если** значение каждого столбца внешнего ключа совпадает со значением соответствующего столбца родительского ключа. Внешний ключ, содержащий пустые значения, не может совпадать со значениями родительского ключа, поскольку родительский ключ по определению не может содержать пустые значения. Однако пустое значение внешнего ключа всегда допустимо, независимо от значений его непустых компонентов.

Для определений внешних ключей применяются следующие правила:

- Таблица может иметь много внешних ключей
- У внешнего ключа допускается пустое значение, если какому-либо из его компонентов разрешено иметь пустые значения
- Внешний ключ имеет пустое значение, если любой из его компонентов имеет пустое значение.

Задачи, связанные с данной темой:

- “Определение ограничения уникальности” на стр. 103
- “Определение реляционных ограничений” на стр. 105

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*
- “CREATE TABLE statement” в *SQL Reference, Том 2*

Условие REFERENCES

Условие REFERENCES задает родительскую таблицу для отношения и определяет необходимые ограничения. Его можно включить в определение столбца или задать как отдельное условие, сопровождающее условие FOREIGN KEY, в операторах CREATE TABLE или ALTER TABLE.

Если условие REFERENCES задано в качестве ограничения для столбца, из перечисленных имен столбцов составляется неявный список столбцов. Учтите, что у нескольких столбцов могут быть отдельные условия REFERENCES, а у одного столбца может быть несколько таких условий.

Условие REFERENCES содержит правило удаления. В нашем примере используется правило удаления ON DELETE NO ACTION, которое указывает, что отделы нельзя удалять, если в них есть сотрудники. Другие правила удаления: ON DELETE CASCADE, ON DELETE SET NULL и ON DELETE RESTRICT.

Понятия, связанные с данным:

- “Условие FOREIGN KEY” на стр. 107

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*
- “CREATE TABLE statement” в *SQL Reference, Том 2*

Влияние на операции утилит

Утилита загрузки отключает проверку ограничений для автореферентных и зависимых таблиц и переводит эти таблицы в состояние отложенной проверки. После завершения работы утилиты загрузки нужно включить проверку ограничений для всех таблиц, для которых она была отключена. Например, если в состояние отложенной проверки были переведены только таблицы DEPARTMENT и EMPLOYEE, можно выполнить следующую команду:

```
SET INTEGRITY FOR DEPARTMENT, EMPLOYEE IMMEDIATE CHECKED
```

Реляционные ограничения влияют на утилиту импорта так:

- Функции REPLACE и REPLACE CREATE не разрешены, если от таблицы объектов зависят какие-либо другие таблицы.

Чтобы использовать эти функции, сначала отбросьте все внешние ключи, в которых эта таблица является родительской. Завершив импорт, заново создайте эти внешние ключи с помощью оператора ALTER TABLE.

- Успех импорта в таблицу с автореферентными ограничениями зависит от порядка, в котором импортируются столбцы.

Понятия, связанные с данным:

- "Import Overview" в *Data Movement Utilities Guide and Reference*
- "Load Overview" в *Data Movement Utilities Guide and Reference*
- "Checking for Integrity Violations" в *Data Movement Utilities Guide and Reference*

Ссылки, связанные с данной темой:

- "SET INTEGRITY statement" в *SQL Reference, Том 2*

Определение проверочного ограничения таблицы

Проверочное ограничение таблицы задает условие поиска, применяемое для каждой строки таблицы, для которой определено это проверочное ограничение таблицы. После создания проверочного ограничения ограничения в менеджере базы данных все изменения данных будут проверены на соответствие этому ограничению. Возможность завершения действия будет зависеть от результата проверки ограничения.

Процедура:

Для создания проверочного ограничения таблицы при создании или изменении этой таблицы с ней связывается определение проверочного ограничения. Это ограничение автоматически активируется, когда операторы INSERT или UPDATE изменяют данные в этой таблице. Проверочное ограничение таблицы не влияет на операторы DELETE или SELECT. Проверочное ограничение может быть связано с типизированной таблицей.

Имя ограничения не может совпадать с именами других ограничений, заданных в том же операторе CREATE TABLE. Если имя ограничения не задано, система сгенерирует 18-символьный уникальный идентификатор для этого ограничения.

Проверочное ограничение таблицы применяется для задания правил целостности данных, не обеспечиваемых уникальностью ключей или реляционными ограничениями. В некоторых случаях проверочное ограничение таблицы может использоваться для выполнения проверки домена. Следующее ограничение, заданное в операторе CREATE TABLE, гарантирует, что дата начала некоторого действия не будет больше даты его завершения:

```
CREATE TABLE EMP_ACT
   (EMPNO   CHAR(6) NOT NULL,
    PROJNO    CHAR(6)                        NOT NULL,
```

```
    ACTNO     SMALLINT                        NOT NULL,
    EMPTIME    DECIMAL(5,2),
    EMSTDATE   DATE,
    EMENDATE   DATE,
    CONSTRAINT ACTDATES CHECK(EMSTDATE <= EMENDATE) )
IN RESOURCE
```

Хотя в предыдущих примерах для добавления проверочного ограничения таблицы использовался оператор CREATE TABLE, для этого можно также использовать оператор ALTER TABLE.

Понятия, связанные с данным:

- “Constraints” в *SQL Reference, Том 1*

Задачи, связанные с данной темой:

- “Добавление проверочного ограничения для таблицы” на стр. 199

Ссылки, связанные с данной темой:

- “CREATE TABLE statement” в *SQL Reference, Том 2*
- “ALTER SERVER statement” в *SQL Reference, Том 2*

Определение информационного ограничения

Информационное ограничение - это правило, которое может применяться компилятором SQL, однако не применяется принудительно менеджером баз данных. Работа компилятора SQL включает в себя этап перезаписи запросов, на котором операторы SQL преобразуются в форму, упрощающую оптимизацию и позволяющую выбрать более эффективный путь доступа к данным. Назначение этого ограничения заключается не в том, чтобы задать очередное условие для данных, проверяемое менеджером баз данных, а в том, чтобы повысить производительность обработки запроса.

Процедура:

Информационное ограничение можно определить с помощью оператора CREATE TABLE или ALTER TABLE. Эти операторы применяются для добавления ограничений реляционной целостности и проверочных ограничений. Для таких ограничений можно задать атрибуты, указывающие, должны ли они принудительно применяться менеджером баз данных, и следует ли применять эти ограничения для оптимизации запроса.

Понятия, связанные с данным:

- “Constraints” в *SQL Reference, Том 1*
- “Работа компилятора SQL” в *Руководство администратора: Производительность*

- “Методы и примеры перезаписи запросов” в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*
- “CREATE TABLE statement” в *SQL Reference, Том 2*

Определение генерируемых столбцов в новых таблицах

Генерируемый столбец определяется в базовой таблице, если сохраняемое значение не задается операциями вставки или изменения, а вычисляется по некоторой формуле.

Процедура:

Если при создании таблицы известно, что для нее все время будут использоваться определенные выражения или предикаты, можно добавить в эту таблицу один или несколько генерируемых столбцов. Генерируемые столбцы позволяют улучшить производительность выполнения запросов для данных этой таблицы.

Например, вычисления выражений могут значительно влиять на производительность в следующих двух случаях:

1. Вычисление выражения должно выполняться в запросе много раз.
2. Требуются сложные вычисления.

Чтобы улучшить производительность запроса, можно определить дополнительный столбец, который будет содержать результаты вычислений для данного выражения. Затем в запросе, в котором используется это же выражение, можно непосредственно использовать этот генерируемый столбец; этот генерируемый столбец также может подставить вместо выражения оптимизатор.

Для генерируемого столбца можно также создать неуникальный индекс.

Для запросов, объединяющих данные двух или нескольких таблиц, добавление генерируемого столбца может позволить оптимизатору выбрать лучшую возможную стратегию объединения.

Ниже показан пример определения генерируемого столбца в операторе CREATE TABLE:

```
CREATE TABLE t1 (c1 INT,
                 c2 DOUBLE,
                 c3 DOUBLE GENERATED ALWAYS AS (c1 + c2)
                 c4 GENERATED ALWAYS AS
                    (CASE WHEN c1 > c2 THEN 1 ELSE NULL END))
```

Создав таблицу, можно использовать эти генерируемые столбцы для создания индексов. Например,

```
CREATE INDEX i1 ON t1(c4)
```

Генерируемые столбцы может быть выгодно использовать в запросах. Например,

```
SELECT COUNT(*) FROM t1 WHERE c1 > c2
```

можно переписать как

```
SELECT COUNT(*) FROM t1 WHERE c4 IS NOT NULL
```

Другой пример:

```
SELECT c1 + c2 FROM t1 WHERE (c1 + c2) * c1 > 100
```

можно переписать как

```
SELECT c3 FROM t1 WHERE c3 * c1 > 100
```

Генерируемые столбцы будут использоваться для улучшения производительности запросов. Полученные генерируемые столбцы будут такими же, как если бы они были добавлены после создания и заполнения таблицы.

Задачи, связанные с данной темой:

- “Определение генерируемого столбца в существующей таблице” на стр. 203

Ссылки, связанные с данной темой:

- “CREATE INDEX statement” в *SQL Reference, Том 2*
- “CREATE TABLE statement” в *SQL Reference, Том 2*
- “SELECT statement” в *SQL Reference, Том 2*

Создание временной пользовательской таблицы

Приложениям, работающим с базами данных, могут потребоваться временные пользовательские таблицы. В этих таблицах обычно хранятся промежуточные результаты вычислений.

Ограничения:

Описание этой таблицы не появляется в системном каталоге, поэтому она недоступна для других прикладных программам и не может ими использоваться.

Когда использующая эту таблицу прикладная программа завершает работу и отсоединяется от базы данных, из таблицы удаляются все данные и она неявно отбрасывается.

Пользовательские временные таблицы не поддерживают:

- Столбцы с типом большой объект (или с пользовательским типом на основе большого объекта)
- Столбцы с пользовательскими типами
- Столбцы LONG VARCHAR
- Столбцы DATALINK

Процедура:

Для определения временной таблицы используется оператор DECLARE GLOBAL TEMPORARY TABLE. Этот оператор используется в прикладной программе. Эта пользовательская временная таблица существует, только пока прикладная программа соединена с базой данных.

Пример определения временной таблицы:

```
DECLARE GLOBAL TEMPORARY TABLE gbl_temp
   LIKE empltabl
   ON COMMIT DELETE ROWS
   NOT LOGGED
   IN usr_tbsp
```

Этот оператор создает пользовательскую временную таблицу с именем `gbl_temp`. Эта пользовательская временная таблица определяется со столбцами, имеющими в точности те же имена и описания, что и столбцы таблицы `empltabl`. Такое неявное определение включает только такие атрибуты столбцов, как имя, тип данных, допустимость пустых значений и значение по умолчанию. Все остальные атрибуты столбцов (ограничения уникальности, ограничения внешних ключей, триггеры и индексы) не определяются. Если для этой таблицы не открыт указатель WITH HOLD, при выполнении операции принятия (COMMIT) из нее удаляются все данные. Информация об изменениях для пользовательской временной таблицы не записывается в журнал. Пользовательская временная таблица помещается в указанное пользовательское временное табличное пространство. Это табличное пространство должно существовать, иначе объявление этой таблицы будет ошибочным.

Если при создании таблицы указано ROLLBACK или ROLLBACK TO SAVEPOINT, то можно либо удалить все строки таблицы (DELETE ROWS, по умолчанию), либо сохранить их (PRESERVE ROWS).

Ссылки, связанные с данной темой:

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

Определение столбцов идентификации в новых таблицах

Столбец идентификации позволяет DB2 автоматически генерировать гарантированно уникальное числовое значение для каждой строки, добавляемой к таблице. Создавая таблицу, в которой нужно уникально идентифицировать каждую строку, можно добавить в нее столбец идентификации.

Ограничения:

После создания таблицы нельзя изменить описание таблицы, включив в нее столбец идентификации.

Столбцы идентификации поддерживаются только в одноразделных базах данных.

Если строки вставляются в таблицу с явно заданными значениями столбцов идентификации, следующее внутреннее генерируемое значение не изменяется, и может возникнуть конфликт с существующими значениями в таблице. Одинаковые значения приведут к появлению сообщения об ошибке.

Процедура:

Для задания столбца идентификации используется условие AS IDENTITY оператора CREATE TABLE.

Ниже показан пример определения столбца идентификации в операторе CREATE TABLE:

```
CREATE TABLE table (col1 INT,
                    col2 DOUBLE,
                    col3 INT NOT NULL GENERATED ALWAYS AS IDENTITY
                                      (START WITH 100, INCREMENT BY 5))
```

В этом примере третий столбец является столбцом идентификации. Можно также задать значение, используемое для создания уникального идентификатора для каждой добавляемой строки. В этом примере столбец идентификации первой строки будет содержать значение “100”; для каждой последующей добавляемой строки значение этого столбца будет увеличиваться на пять.

Другие примеры использования столбца идентификации: номер заказа, номер работника, номер фонда или номер события. Значения для столбца идентификации могут генерироваться DB2 двумя способами: ALWAYS (всегда) или BY DEFAULT (по умолчанию).

Значения столбца идентификации, определенного как GENERATED ALWAYS, всегда будут созданы DB2. Прикладным программам не разрешается задавать для него явные значения. Для столбца идентификации, определенного как

GENERATED BY DEFAULT, прикладные программы могут явно задавать его значения. Если значение не задано прикладной программой, DB2 сгенерирует это значение. Поскольку это значение может задавать прикладная программа, DB2 не может гарантировать его уникальность. Условие GENERATED BY DEFAULT предназначено для использования при распространении данных, когда нужно скопировать содержимое существующей таблицы, или для операций выгрузки и повторной загрузки таблицы.

Понятия, связанные с данным:

- “Сравнение столбцов идентификации (IDENTITY) и последовательностей” на стр. 117

Ссылки, связанные с данной темой:

- “CREATE TABLE statement” в *SQL Reference, Том 2*

Создание последовательности

Последовательность - это объект базы данных, поддерживающий автоматическую генерацию значений. Последовательности идеально подходят для генерации уникальных значений ключей. Прикладные программы могут использовать последовательности для предотвращения конфликтов и проблем производительности из-за генерации счетчика уникальности вне базы данных.

Ограничения:

В отличие от атрибута столбца идентификации, последовательность не привязана к конкретному столбцу таблицы и не связана с уникальным столбцом таблицы, и только доступ к ней происходит через этот столбец таблицы.

Последовательности поддерживаются только в одноразделных базах данных.

Если база данных, содержащая одну или несколько последовательностей, восстанавливается до более раннего момента времени, возникает опасность генерации повторных значений для некоторых последовательностей. Чтобы исключить появление повторяющихся значений, не следует подвергать базу данных с последовательностями восстановлению до более раннего момента времени.

Выражения NEXTVAL и PREVVAL можно использовать с определенными ограничениями.

Процедура:

При создании или изменении последовательности можно задать, чтобы генерация значений происходила по одному из следующих методов:

- Монотонное увеличение или уменьшение без ограничений

- Монотонное увеличение или уменьшение до определяемого пользователем предела и остановка
- Монотонное увеличение или уменьшение до определяемого пользователем предела и циклическое повторение

Ниже приводится пример создания объекта последовательности:

```
CREATE SEQUENCE order_seq
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE
   CACHE 24
```

В этом примере последовательность названа `order_seq`. Она начинается с 1 и будет увеличиваться на 1 без ограничений сверху. Поскольку верхний предел не задан, нет смысла задавать возврат к 1 и повторение сначала. Число, связанное с параметром `CACHE`, задает максимальное число значений последовательности, которые менеджер баз данных разместит и будет сохранять в памяти.

Генерируемые числовые последовательности обладают следующими свойствами:

- Значения могут быть любого точного числового типа данных с масштабом 0. К таким типам данных относятся: SMALLINT, BIGINT, INTEGER и DECIMAL.
- Последовательные значения могут отличаться на любое указанное целочисленное приращение. Значение приращения по умолчанию - 1.
- Значение счетчика допускает восстановление. Когда требуется восстановление, значение счетчика воссоздается по журналам.
- Значения могут помещаться в кэш для повышения производительности. Размещение и хранение значений в кэше сокращает синхронный ввод-вывод в журнал при генерации значений для последовательности. При системном сбое все значения в кэше, которые не были приняты, больше не используются и считаются потерянными. Заданное значение `CACHE` - это максимальное число значений последовательности, которое может быть потеряно при этом.

Для работы с последовательностями используются два выражения.

Выражение PREVVAL возвращает последнее значение, сгенерированное для данной последовательности предыдущим оператором в текущем процессе приложения.

Выражение NEXTVAL возвращает следующее значение данной последовательности. Когда имя последовательности задано в выражении NEXTVAL, для нее генерируется новое значение. Однако если в одном запросе есть несколько экземпляров выражения NEXTVAL с одним и тем же именем

последовательности, приращение счетчика последовательности производится только один раз для каждой строки результата.

Один и тот же член числовой последовательности может использоваться как значение ключа уникальности в двух отдельных таблицах, если в первой строке ссылка на член числовой последовательности делается при помощи выражения NEXTVAL, а во всех остальных строках - при помощи выражения PREVVAL.

Например:

```
INSERT INTO order (orderno, custno)
   VALUES (NEXTVAL FOR order_seq, 123456);
INSERT INTO line_item (orderno, partno, quantity)
   VALUES (PREVVAL FOR order_seq, 987654, 1)
```

Понятия, связанные с данным:

- “Сравнение столбцов идентификации (IDENTITY) и последовательностей” на стр. 117

Ссылки, связанные с данной темой:

- “CREATE SEQUENCE statement” в *SQL Reference, Том 2*

Сравнение столбцов идентификации (IDENTITY) и последовательностей

При всем сходстве столбцов идентификации и последовательностей между ними есть и различия. Особенности этих двух подходов могут использоваться при проектировании баз данных и прикладных программ.

Особенности столбца идентификации:

- Столбец идентификации может быть определен как часть таблицы только в момент создания таблицы. После того, как таблица создана, добавить к ней столбец идентификации нельзя. (Однако можно изменить характеристики существующего столбца идентификации.)
- Столбец идентификации автоматически генерирует значения для одной таблицы.
- Когда столбец идентификации определен как GENERATED ALWAYS, используемые значения всегда генерируются менеджером баз данных. Прикладным программам не разрешено задавать собственные значения при изменении содержимого таблицы.

Объект последовательности имеет следующие особенности:

- Объект последовательности - это объект базы данных, не привязанный к какой-либо одной таблице.
- Объект последовательности генерирует последовательные значения, которые можно использовать в любом операторе SQL.

- Поскольку объект последовательности может использоваться любой прикладной программой, есть два выражения, управляющие получением либо очередного значения в данной последовательности, либо значения, сгенерированного перед выполнением данного оператора. Выражение PREVVAL возвращает последнее значение, сгенерированное для данной последовательности предыдущим оператором в текущем сеансе. Выражение NEXTVAL возвращает следующее значение данной последовательности. Использование этих выражений допускает, чтобы одно и то же значение использовалось для разных операторов SQL в разных таблицах.

Хотя это и не все особенности этих двух подходов, изложенное поможет вам определить, какой из них использовать с учетом структуры вашей базы данных и использующих базу данных прикладных программ.

Задачи, связанные с данной темой:

- “Определение генерируемых столбцов в новых таблицах” на стр. 111
- “Создание последовательности” на стр. 115
- “Определение генерируемого столбца в существующей таблице” на стр. 203

Определение измерений таблицы

Измерение представляет собой ключ кластеризации для таблицы. Для таблицы можно выбрать одно или несколько измерений. Если для таблицы выбрано несколько измерений, то таблица называется таблицей с многомерной кластеризацией. Такая таблица создается с помощью оператора CREATE TABLE с условием ORGANIZE BY DIMENSIONS.

Ограничения:

Набор столбцов, заданный в условии ORGANIZE BY [DIMENSIONS], должен отвечать требованиям, предъявляемым оператором CREATE INDEX. Столбцы рассматриваются как ключи, применяемые для физического хранения данных в определенном порядке.

Процедура:

Каждое измерение задается в условии ORGANIZE BY [DIMENSIONS] оператора CREATE TABLE с помощью списка столбцов. Столбцы, относящиеся к одному измерению, заключаются в круглые скобки.

Данные одновременно физически кластеризуются по указанному количеству измерений. Они размещаются в виде экстентов или “блоков” вдоль линий измерений. Если в запросе задан предикат измерения, то просмотр выполняется только в тех экстентах таблицы, которые содержат значения соответствующего измерения. Поскольку экстент представляет собой группу страниц,

последовательно расположенных на диске, эффективность предварительной выборки во время такого просмотра будет очень высокой.

Степень кластеризации таблицы с одним индексом кластеризации уменьшается в процессе заполнения таблицы, однако кластеризация таблицы с несколькими измерениями автоматически поддерживается по всем измерениям. В результате не требуется выполнять реорганизацию таблицы, для того чтобы заново упорядочить данные.

Для каждого указанного измерения автоматически создается блочный индекс измерения. Этот индекс применяется для доступа к данным в измерении. Блочный индекс измерения содержит ссылки на экстенты, а не на отдельные строки, поэтому он значительно меньше обычных индексов. Блочные индексы измерения позволяют очень быстро получить доступ к тем экстентам таблицы, которые содержат значения из определенных измерений.

Кроме того, автоматически создается составной блочный индекс, содержащий все столбцы ключей измерения. Такой индекс применяется для поддержания кластеризации данных при выполнении операций вставки и обновления. Составной блочный индекс применяется при обработке запросов для обращения к данным таблицы, содержащим определенные значения измерений.

Примечание: При выборе составного блочного индекса для обработки запроса учитывается порядок его ключевых компонентов. Порядок ключевых компонентов определяется порядком столбцов, заданных во всем условии ORGANIZE BY [DIMENSIONS] при создании таблицы MDC. Например, если таблица была создана с помощью оператора:

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c4, (c3,c1), c2)
```

то составной блочный индекс будет создан для столбцов (c1,c4,c3,c2). Хотя столбец c1 дважды указан в определении измерений, он только один раз используется как ключевой компонент составного блочного индекса. При этом применяется первое вхождение имени столбца в определение измерений. Порядок ключевых компонентов составного блочного индекса не влияет на обработку операций вставки, однако может влиять на обработку запросов. Следовательно, лучше создать составной блочный индекс, в котором столбцы будут упорядочены следующим образом: (c1,c2,c3,c4). Для этого таблицу нужно создать с помощью следующего оператора:

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c2, (c3,c1), c4)
```

Составной блочный индекс не создается в том случае, если одно из измерений уже содержит все те столбцы, которые были бы добавлены в составной блочный индекс. Например, составной блочный индекс не был бы создан для следующей таблицы:

```
CREATE TABLE t1 (c1 int, c2 int)
   ORGANIZE BY DIMENSIONS (c1,(c2,c1))
```

Понятия, связанные с данным:

- “Многомерная кластеризация” в *Руководство администратора: Планирование*

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*
- “CREATE TABLE statement” в *SQL Reference, Том 2*

Создание типизированной таблицы

При создании иерархии структурированных типов вам потребуется создать типизированные таблицы. Типизированные таблицы применяются для хранения экземпляров объектов, характеристики которых определены выражением CREATE TYPE.

Предварительные требования:

Тип, на котором основана типизированная таблица, должен существовать.

Процедура:

Для создания типизированной таблицы можно использовать разновидность оператора CREATE TABLE.

Понятия, связанные с данным:

- “Typed Tables” в *Application Development Guide: Programming Server Applications*

Задачи, связанные с данной темой:

- “Заполнение типизированной таблицы” на стр. 121
- “Создание типизированной производной таблицы” на стр. 141
- “Dropping Typed Tables” в *Application Development Guide: Programming Server Applications*
- “Creating Typed Tables” в *Application Development Guide: Programming Server Applications*

Ссылки, связанные с данной темой:

- “CREATE TABLE statement” в *SQL Reference, Том 2*
- “CREATE TYPE (Structured) statement” в *SQL Reference, Том 2*

Заполнение типизированной таблицы

При создании иерархии структурированных типов вам потребуется создать типизированные таблицы. Типизированные таблицы применяются для хранения экземпляров объектов, характеристики которых определены выражением CREATE TYPE. После создания типизированной таблицы ее потребуется заполнить данными.

Предварительные требования для установки:

Типизированная таблица уже должна существовать.

Процедура:

После того, как созданы структурированные типы, созданы соответствующие таблицы и подтаблицы, можно заполнить типизированную таблицу данными.

Понятия, связанные с данным:

- “Substitutability in Typed Tables” в *Application Development Guide: Programming Server Applications*
- “Typed Tables” в *Application Development Guide: Programming Server Applications*

Задачи, связанные с данной темой:

- “Создание типизированной таблицы” на стр. 120
- “Storing Objects in Typed Table Rows” в *Application Development Guide: Programming Server Applications*
- “Dropping Typed Tables” в *Application Development Guide: Programming Server Applications*
- “Creating Typed Tables” в *Application Development Guide: Programming Server Applications*

Ссылки, связанные с данной темой:

- “CREATE TYPE (Structured) statement” в *SQL Reference, Том 2*

Таблица иерархии

Таблица иерархии - это таблица, связанная с реализацией иерархии типизированных таблиц. Она создается одновременно с корневой таблицей иерархии.

Задачи, связанные с данной темой:

- “Creating a Structured Type Hierarchy” в *Application Development Guide: Programming Server Applications*

Создание таблицы в нескольких табличных пространствах

Данные таблицы могут храниться в том же табличном пространстве, что и индекс этой таблицы и все данные длинных столбцов, связанные с этой таблицей. Можно также разместить индекс в отдельном табличном пространстве, а данные длинных столбцов - в своем табличном пространстве, отдельно от табличного пространства для остальных данных таблицы.

Предварительные требования для установки:

Все табличные пространства должны существовать перед выполнением оператора CREATE TABLE.

Ограничения:

Раздельное размещение частей таблицы возможно только при использовании табличных пространств DMS.

Процедура:

Чтобы создать таблицу в нескольких табличных пространствах с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой мыши по папке **Таблицы** и выберите из всплывающего меню пункт **Создать —> Таблицы при помощи мастера**.
3. Введите имя таблицы и нажмите кнопку **Следующий**.
4. Выберите столбцы для таблицы.
5. На странице **Табличное пространство** выберите **Использовать отдельное индексное пространство** и **Использовать отдельное пространство длинных данных**, задайте информацию и нажмите кнопку **Завершить**.

Чтобы создать таблицу в нескольких табличных пространствах из командной строки, введите команду:

```
CREATE TABLE <имя>
   (<имя_столбца>  <тип_данных>  <атрибут_пустого_значения>)
    IN <имя_табличного_пространства>
    INDEX IN <имя_индексного_пространства>
    LONG  IN <имя_пространства_длинных_данных>
```

В следующем примере показано, как создать таблицу EMP_PHOTO, различные части которой хранятся в разных табличных пространствах:

```
CREATE TABLE EMP_PHOTO
   (EMPNO   CHAR(6) NOT NULL,
    PHOTO_FORMAT VARCHAR(10)  NOT NULL,
```

```
    PICTURE      BLOB(100K) )
IN RESOURCE
INDEX IN RESOURCE_INDEXES
LONG  IN RESOURCE_PHOTO
```

Данные созданной в этом примере таблицы EMP_PHOTO будут храниться так:

- Индексы, созданные для таблицы EMP_PHOTO, будут храниться в табличном пространстве RESOURCES_INDEXES
- Данные столбца PICTURE будут храниться в табличном пространстве RESOURCE_PHOTO
- Данные столбцов EMPNO и PHOTO_FORMAT будут храниться в табличном пространстве RESOURCE.

Ссылки, связанные с данной темой:

- “CREATE TABLE statement” в *SQL Reference, Том 2*

Создание таблицы в многораздельной базе данных

Создание таблицы в нескольких разделах базы данных дает определенный выигрыш в производительности. Операции, связанные с получением данных, будут распределены между несколькими разделами базы данных.

Предварительные требования:

Перед созданием таблицы, которая будет физически распределена или разбита на разделы, нужно учесть следующее:

- Табличное пространство может располагаться на нескольких разделах базы данных. Число этих разделов зависит от числа разделов в группе разделов базы данных.
- Таблицы могут быть размещены совместно. Для этого они должны быть расположены в одном табличном пространстве или в разных табличных пространствах, относящихся к одной группе разделов базы данных.

Ограничения:

Важно правильно выбрать ключ разделения, так как *в дальнейшем его нельзя будет изменить*. Более того, ключ разделения должен входить во все индексы уникальности (и, следовательно, в ключи уникальности и в первичный ключ). То есть, если определен ключ разделения, то ключи уникальности и первичные ключи должны включать в себя все столбцы, входящие в ключ разделения (но они могут содержать и дополнительные столбцы).

Максимальный размер одного раздела таблицы - 64 Гбайта (или доступный объем дискового пространства, если он меньше 64 Гбайт). (Здесь подразумевается, что для табличного пространства используются 4-Кбайтные страницы.) Размер таблицы может быть равен 64 Гбайтам (или доступному

дисковому пространству), умноженным на число разделов базы данных. Если размер страницы для табличного пространства равен 8 Кбайтам, размер таблицы может быть равен 128 Гбайтам (или доступному дисковому пространству), умноженным на число разделов базы данных. Если размер страницы для табличного пространства равен 16 Кбайтам, размер таблицы может быть равен 256 Гбайтам (или доступному дисковому пространству), умноженным на число разделов базы данных.Если размер страницы для табличного пространства равен 32 Кбайтам, размер таблицы может быть равен 512 Гбайтам (или доступному дисковому пространству), умноженным на число разделов базы данных.

Процедура:

При создании таблицы можно указать, что таблица будет содержаться в нескольких разделах базы данных. В случае создания таблицы в многораздельной базе данных, кроме всего прочего, следует указать *ключ разделения*. Ключ разделения - это ключ, являющийся частью определения таблицы. Он определяет раздел, где хранится данная строка таблицы.

Если ключ разделения не задан явно, используются следующие умолчания. *Убедитесь, что этот ключ разделения по умолчанию вам подходит.*

- Если в операторе CREATE TABLE задан первичный ключ, в качестве ключа разделения используется первый столбец первичного ключа.
- Если первичный ключ отсутствует, используется первый столбец, не являющийся длинным полем.
- Если нет столбцов, удовлетворяющих требованиям к ключу разделения по умолчанию, таблица создается без ключа разделения (это разрешено только в одноразделных группах узлов).

Пример:

```
CREATE TABLE MIXREC (MIX_CNTL INTEGER NOT NULL,
                     MIX_DESC CHAR(20) NOT NULL,
                     MIX_CHR  CHAR(9) NOT NULL,
                     MIX_INT INTEGER NOT NULL,
                     MIX_INTS SMALLINT NOT NULL,
                     MIX_DEC DECIMAL NOT NULL,
                     MIX_FLT FLOAT NOT NULL,
                     MIX_DATE DATE NOT NULL,
                     MIX_TIME TIME NOT NULL,
                     MIX_TMSTMP TIMESTAMP NOT NULL)
                     IN MIXTS12
                     PARTITIONING KEY (MIX_INT) USING HASHING
```

В предыдущем примере задано табличное пространство `MIXTS12` и определен ключ разделения `MIX_INT`. Если бы ключ разделения не был задан явно, был бы

использован столбец MIX_CNTL. (Если не задан первичный ключ и не определен ключ разделения, ключом разделения будет первый столбец в списке, не содержащий длинных данных.)

Строка таблицы и вся информация для этой строки всегда располагаются в одном разделе базы данных.

Понятия, связанные с данным:

- "Группы разделов базы данных" в *Руководство администратора: Планирование*
- "Проектирование групп разделов базы данных" в *Руководство администратора: Планирование*
- "Совместное размещение таблиц" в *Руководство администратора: Планирование*

Ссылки, связанные с данной темой:

- "CREATE TABLE statement" в *SQL Reference, Том 2*

Создание триггера

Триггер определяет набор действий, выполняемых вместе с операцией (то есть запускаемых операцией) INSERT, UPDATE или DELETE для заданной базовой или типизированной таблицы. Например, триггеры можно использовать для:

- Проверки входных данных
- Генерации значения для только что вставленной строки
- Чтения из других таблиц для задания перекрестных ссылок
- Записи в другие таблицы для контроля процесса

Триггеры можно использовать для поддержки общих требований целостности или выполнения логических правил. Например, перед принятием заказа или обновлением сводной таблицы данных триггер может проверять, не превышен ли предел кредита для данного покупателя.

Применение триггера дает следующие преимущества:

- Ускорение разработки прикладных программ: Поскольку триггер хранится в базе данных, выполняемые им действия не требуется указывать в исходном коде каждой прикладной программы.
- Упрощение обслуживания: После того, как триггер определен, он автоматически вызывается при обращениях к таблице, для которой он создан.
- Глобальное задание логических правил: При изменении логических правил потребуется изменить только триггер, а не все прикладные программы.

Ограничения:

Нельзя использовать триггеры с псевдонимами.

Для триггера типа BEFORE в действии триггера нельзя задавать имя генерируемого столбца, если только это не столбец идентификации. Это значит, что в триггере BEFORE можно использовать значение генерируемого столбца идентификации.

При создании элементарного триггера будьте осторожны с символом окончания оператора. Менеджер баз данных в качестве символа по умолчанию предполагает ";". При создании элементарного триггера вам следует вручную отредактировать символ окончания оператора в своем сценарии, чтобы использовать символ, отличающийся от ";". ";" можно заменить другим специальным символом, например "#".

После этого нужно выполнить одно из двух действий:

- Изменить разделитель в меню Инструменты—>Параметры инструментов на вкладке Сеанс в Командном центре, а затем запустить данный сценарий,
- Ввести в командной строке:

  ```
  db2 -td <разделитель> -vf <сценарий>
  ```

  где разделитель - это альтернативный символ окончания оператора, а <сценарий> - измененный сценарий с новым разделителем.

Процедура:

Чтобы создать триггер с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Триггеры**.
2. Щелкните правой кнопкой мыши по папке **Триггеры** и выберите из всплывающего меню пункт **Создать**.
3. Задайте информацию об этом триггере.
4. Задайте действие, вызываемое этим триггером, и нажмите кнопку **OK**.

Чтобы создать триггер из командной строки, введите команду:

```
CREATE TRIGGER <имя>
   <действие> ON <имя_таблицы>
   <операция>
   <действие_триггера>
```

Следующий оператор SQL создает триггер, увеличивающий общее число сотрудников при приеме на работу нового человека, для чего он при каждом

добавлении строки в таблицу EMPLOYEE прибавляет 1 к значению числа сотрудников в столбце NBEMP таблицы COMPANY_STATS.

```
CREATE TRIGGER NEW_HIRED
   AFTER INSERT ON EMPLOYEE
   FOR EACH ROW MODE DB2SQL
   UPDATE COMPANY_STATS SET NBEMP = NBEMP+1;
```

Тело триггера может содержать один или несколько операторов SQL следующих типов: INSERT, UPDATE с поиском, DELETE с поиском, полная выборка, SET, задающий значение временных переменных, и SIGNAL SQLSTATE. Триггер может активироваться до (BEFORE) или после (AFTER) выполнения оператора INSERT, UPDATE или DELETE, для которого он определен.

Понятия, связанные с данным:

- “Зависимости триггера” на стр. 127
- “INSERT, UPDATE, and DELETE Triggers” в *Application Development Guide: Programming Server Applications*
- “Triggers in Application Development” в *Application Development Guide: Programming Server Applications*
- “Trigger Creation Guidelines” в *Application Development Guide: Programming Server Applications*
- “Применение триггеров для обновления производных таблиц” на стр. 128

Задачи, связанные с данной темой:

- “Отбрасывание триггера” на стр. 218
- “Creating Triggers” в *Application Development Guide: Programming Server Applications*
- “Defining Business Rules Using Triggers” в *Application Development Guide: Programming Server Applications*
- “Defining Actions Using Triggers” в *Application Development Guide: Programming Server Applications*

Ссылки, связанные с данной темой:

- “CREATE TRIGGER statement” в *SQL Reference, Том 2*

Зависимости триггера

Все зависимости триггера от других объектов записываются в каталоге SYSCAT.TRIGDEP. Триггер может зависеть от многих объектов. Подробная информация об этих объектах и зависимых от них триггерах приведена в описании оператора DROP.

Если один из этих объектов отброшен, триггер становится неработоспособным, но его определение сохраняется в каталоге. Чтобы восстановить работоспособность этого триггера, необходимо получить из каталога его описание и ввести новый оператор CREATE TRIGGER.

Если триггер отброшен, его описание удаляется из производной таблицы каталога SYSCAT.TRIGGERS и все его зависимости удаляются из производной таблицы каталога SYSCAT.TRIGDEP. Все пакеты, имеющие зависимости операторов UPDATE, INSERT или DELETE от этого триггера, становятся недействительными.

Если зависимый объект - это производная таблица, и она стала неработоспособной, триггер также отмечается как неработоспособный. Все пакеты, зависящие от триггера, отмеченного как неработоспособный, становятся недействительными.

Понятия, связанные с данным:

- “Применение триггеров для обновления производных таблиц” на стр. 128

Задачи, связанные с данной темой:

- “Создание триггера” на стр. 125
- “Отбрасывание триггера” на стр. 218

Ссылки, связанные с данной темой:

- “CREATE TRIGGER statement” в *SQL Reference, Том 2*
- “DROP statement” в *SQL Reference, Том 2*

Применение триггеров для обновления производных таблиц

С помощью триггеров INSTEAD OF можно выполнять операции удаления, вставки и обновления для производной таблицы, которая не унаследовала атрибут, разрешающий обновление. В прикладных программах эти триггеры позволяют выполнять операции обновления для производных таблиц так же, как и для обычных таблиц.

Например, можно создать производную таблицу с помощью следующих операторов SQL:

```
CREATE VIEW EMPV(EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO, HIREDATE,
      DEPTNAME)
   AS SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO, HIREDATE, DEPTNAME
      FROM EMPLOYEE, DEPARTMENT WHERE EMPLOYEE.WORKDEPT = DEPARTMENT.DEPTNO
```

Так как производная таблица EMPV формируется путем выполнения операции объединения, с ее помощью нельзя обновлять данные в базовых таблицах, если не будут добавлены следующие операторы:

```
CREATE TRIGGER EMPV_INSERT INSTEAD OF INSERT ON EMPV
   REFERENCING NEW AS NEWEMP DEFAULTS NULL FOR EACH ROW MODE DB2SQL
   INSERT INTO EMPLOYEE (EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT,
      PHONENO, HIREDATE)
   VALUES(EMPNO, FIRSTNME, MIDINIT, LASTNAME,
       COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
       WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
       RAISE_ERROR('70001', 'Unknown department name')),
       PHONENO, HIREDATE)
```

Этот оператор CREATE TRIGGER позволит выполнять запросы INSERT для производной таблицы EMPV.

```
CREATE TRIGGER EMPV_DELETE INSTEAD OF DELETE ON EMPV
   REFERENCING OLD AS OLDEMP FOR EACH ROW MODE DB2SQL
   DELETE FROM EMPLOYEE AS E WHERE E.EMPNO = OLDEMP.EMPNO
```

Этот оператор CREATE TRIGGER позволит выполнять запросы DELETE для производной таблицы EMPV.

```
CREATE TRIGGER EMPV_UPDATE INSTEAD OF UPDATE ON EMPV
   REFERENCING NEW AS NEWEMP
               OLD AS OLDEMP
   DEFAULTS NULL FOR EACH ROW MODE DB2SQL
   BEGIN ATOMIC
      VALUES(CASE WHEN NEWEMP.EMPNO = OLDEMP.EMPNO THEN 0
                  ELSE RAISE_ERROR('70002', 'Must not change EMPNO') END);
      UPDATE EMPLOYEE AS E
         SET (FIRSTNME, MIDINIT, LASTNAME, WORKDEPT, PHONENO, HIREDATE) =
             (NEWEMP.FIRSTNME, NEWEMP.MIDINIT, NEWEMP.LASTNAME,
              COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
                          WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
              RAISE_ERROR('70001', 'Unknown department name')),
              NEWEMP.PHONENO, NEWEMP.HIREDATE)
         WHERE NEWEMP.EMPNO = E.EMPNO;
   END
```

Этот оператор CREATE TRIGGER позволит выполнять запросы UPDATE для производной таблицы EMPV.

Задачи, связанные с данной темой:

- “Создание триггера” на стр. 125

Ссылки, связанные с данной темой:

- “CREATE TRIGGER statement” в *SQL Reference, Том 2*

Создание пользовательской функции (UDF) или метода

Пользовательские функции расширяют возможности, предоставляемые встроенными функциями SQL, и их можно использовать везде, где могут использоваться встроенные функции. Пользовательскую функцию можно создать как:

- Внешнюю функцию, которая написана на одном из языков программирования
- Функцию с источником, реализация которой наследуется от какой-либо другой существующей функции

Существует три типа пользовательских функций:

Скалярная

При каждом вызове возвращает одно значение. Пример скалярной функции - встроенная функция SUBSTR(). Скалярные пользовательские функции могут быть внешними функциями или функциями с источником.

Функция столбца

Возвращает одно значение для набора однородных значений (столбца). В DB2 иногда также называется сводной функцией. Пример функции столбца - встроенная функция AVG(). В DB2 нельзя определить внешнюю пользовательскую функцию столбца, но ее можно определить на основе одной из встроенных функций столбца. Это удобно для пользовательских типов.

Например, если имеется пользовательский тип SHOESIZE, определенный на основе типа INTEGER, можно определить пользовательскую функцию столбца AVG(SHOESIZE), источником которой будет встроенная функция AVG(INTEGER).

Табличная

Возвращает вызвавшему ее оператору SQL таблицу. Табличные функции можно вызывать только из условия FROM оператора SELECT. Такие функции позволяют применять возможности языка SQL для обработки данных, отличных от данных DB2®, и преобразовывать эти данные в таблицу DB2.

Например, табличная функция может преобразовывать файл в таблицу, вносить в таблицу примеры данных из WWW или возвращать информацию о почтовых сообщениях (дату, от кого получено письмо и текст сообщения), хранящуюся в базе данных Lotus® Notes. Эту информацию можно затем объединить с другими таблицами базы данных.

Табличная функция может быть только внешней функцией. Она не может быть функцией с источником.

Информация о существующих пользовательских функций записывается в производные таблицы каталога SYSCAT.FUNCTIONS и SYSCAT.FUNCPARMS. Системный каталог *не* содержит выполняемого кода пользовательской функции. (Поэтому создавая планы резервного копирования и восстановления, нужно принять решения насчет выполняемых файлов пользовательских функций.)

При компиляции операторов SQL важна статистическая информация о производительности пользовательских функций.

Понятия, связанные с данным:

- "Scalar functions" в *SQL Reference, Том 1*
- "User-defined functions" в *SQL Reference, Том 1*
- "Table functions" в *SQL Reference, Том 1*
- "Создание отображения функции" на стр. 131
- "Статистика для пользовательских функций" в *Руководство администратора: Производительность*
- "Общие правила обновления статистики каталогов вручную" в *Руководство администратора: Производительность*
- "DB2 User-Defined Functions and Methods" в *Application Development Guide: Programming Client Applications*

Задачи, связанные с данной темой:

- "Создание шаблона функции" на стр. 132

Ссылки, связанные с данной темой:

- "Functions" в *SQL Reference, Том 1*
- "CREATE FUNCTION statement" в *SQL Reference, Том 2*

Сведения о создании пользовательской функции (UDF) или метода

В этом разделе приведены общие сведения о создании пользовательских функций и методов.

Создание отображения функции

В базе данных объединения отображение функции применяется для отображения локальной функции или локального шаблона функции в функцию одного или нескольких источников данных. Для многих функций источников данных существуют отображения функций по умолчанию.

Отображения функций удобны, когда:

- На источнике данных становятся доступны новые встроенные функции.

- Нужно отобразить пользовательскую функцию источника данных на локальную функцию.
- Для прикладной программы требуется поведение, отличное от поведения по умолчанию, определяемого отображением по умолчанию.

Отображения функций, определенные с помощью операторов CREATE FUNCTION MAPPING, хранятся в базе данных объединения.

Функции (или шаблоны функций) должны иметь то же число входных параметров, что и функция источника данных. Кроме того, типы данных входных параметров на стороне объединения должны быть совместимы с типами данных входных параметров на стороне источника данных. Эти же требования применяются и для возвращаемых значений.

Для создания отображения функции используется оператор CREATE FUNCTION MAPPING. Например, для создания отображения функции Oracle AVGNEW в аналогичную функцию DB2® на сервере ORACLE1:

```
CREATE FUNCTION MAPPING ORAVGNEW FOR SYSIBM.AVG(INT) SERVER ORACLE1
OPTIONS (REMOTE_NAME 'AVGNEW')
```

Чтобы использовать этот оператор, необходимо обладать полномочиями SYSADM или DBADM для базы данных объединения. Атрибуты отображения функции хранятся в SYSCAT.FUNCMAPPINGS.

Сервер объединения не будет связывать входные переменные хоста или получать результаты типов большой объект, LONG VARCHAR/VARGRAPHIC, DATALINK, пользовательских типов и структурированных типов. Нельзя создать отображение функции, если входные параметры или возвращаемое значение относятся к одному из этих типов.

Понятия, связанные с данным:

- “Host Language Program Mappings with Transform Functions” в *Application Development Guide: Programming Server Applications*

Задачи, связанные с данной темой:

- “Создание шаблона функции” на стр. 132

Ссылки, связанные с данной темой:

- “CREATE FUNCTION MAPPING statement” в *SQL Reference, Том 2*

Создание шаблона функции

В системе объединения для “привязки” отображений функций используются шаблоны функций. Они используются, чтобы разрешить отображение функции источника данных, когда соответствующая функция DB2 не существует на

сервере объединения. Для отображения функции требуется наличие шаблона функции или существование подобной функции в DB2.

Шаблон представляет собой лишь оболочку функции: имя, входные параметры и возвращаемое значение. Для этой функции нет локального выполняемого файла.

Ограничения:

Поскольку в локальной системе нет исполняемого файла функции, вызов шаблона функции может завершиться неудачно, даже если функция доступна в источнике данных. Например, рассмотрим такой запрос:

```
SELECT myfunc(C1)
   FROM nick1
   WHERE C2 < 'A'
```

Если DB2 и источник данных, содержащий объект nick1, имеют разные последовательности слияния, этот запрос вызовет ошибку, поскольку сравнение должно выполняться на DB2, а функция - на источнике данных. Если последовательности слияния совпадают, операция сравнения может быть выполнена на источнике данных, содержащем базовую функцию `myfunc`.

Функции (или шаблоны функций) должны иметь то же число входных параметров, что и функция источника данных. Типы данных входных параметров на стороне объединения должны быть совместимы с типами данных входных параметров на стороне источника данных. Эти же требования применяются и для возвращаемых значений.

Процедура:

Для создания шаблона функции используется оператор CREATE FUNCTION с ключевым словом AS TEMPLATE. Когда шаблон создан, с помощью оператора CREATE FUNCTION MAPPING для него задается отображение на функцию источника данных.

Например, чтобы создать шаблон функции и отображение функции для функции MYS1FUNC на сервере S1:

```
CREATE FUNCTION MYFUNC(INT) RETURNS INT AS TEMPLATE

CREATE FUNCTION MAPPING S1_MYFUNC FOR MYFUNC(INT) SERVER S1 OPTIONS
(REMOTE_NAME 'MYS1FUNC')
```

Понятия, связанные с данным:

- “Создание отображения функции” на стр. 131

Ссылки, связанные с данной темой:

- “CREATE FUNCTION (Sourced or Template) statement” в *SQL Reference, Том 2*

Пользовательский тип (UDT)

Пользовательский тип - это именованный тип данных, созданный в базе данных пользователем. Этот тип может быть пользовательским типом с тем же представлением данных, что встроенный тип данных, или структурированным типом, содержащим последовательность поименованных атрибутов, каждый из которых имеет тип. Структурированный тип может быть подтипом другого структурированного типа (называемого надтипом), образуя иерархию типов.

Пользовательские типы поддерживают строгую типизацию, то есть, если этот пользовательский тип использует то же представление данных, что и другие типы, значения этого типа считаются совместимыми только со значениями этого же типа или типа в той же иерархии типов.

В производной таблице каталога SYSCAT.DATATYPES можно увидеть пользовательские типы, определенные для этой базы данных. В этой производной таблице каталога можно также увидеть типы данных, определенные менеджером баз данных при создании базы данных.

Пользовательские типы нельзя использовать как типы аргументов для большинства поставляемых с системой (встроенных) функций. Для работы с этими типами можно создать пользовательские функции.

Пользовательский тип можно отбросить, только если:

- Он *не* используется в определении столбца существующей таблицы.
- Он *не* используется в качестве типа существующей типизированной таблицы или типизированной производной таблицы.
- Он *не* используется в пользовательской функции, которую нельзя отбросить. Пользовательскую функцию нельзя отбросить, если от нее зависит производная таблица, триггер, проверочное ограничение таблицы или другая пользовательская функция.

При отбрасывании пользовательского типа отбрасываются также все зависящие от него функции.

Понятия, связанные с данным:

- “Создание пользовательского структурированного типа” на стр. 136

Задачи, связанные с данной темой:

- “Создание пользовательского особого типа” на стр. 135

Ссылки, связанные с данной темой:

- “User-defined types” в *SQL Reference, Том 1*
- “Data types” в *SQL Reference, Том 1*

Сведения о создании пользовательских типов (UDT)

В этом разделе обсуждаются особые типы и определения структурированных типов.

Создание пользовательского особого типа

Пользовательский особый тип - это тип данных, созданный на основе одного из существующих типов данных (таких как целый, десятичный или символьный). Для создания особого типа используется оператор CREATE DISTINCT TYPE.

Ограничения:

Если в операторе CREATE DISTINCT TYPE задано условие WITH COMPARISONS (как в этом примере), экземпляры одного особого типа могут сравниваться друг с другом. Условие WITH COMPARISONS нельзя задавать, если исходный тип данных - это большой объект, DATALINK, LONG VARCHAR или LONG VARGRAPHIC.

Экземпляры особых типов нельзя использовать в качестве аргументов функций или операндов операций, определенных для исходного типа. Аналогично и исходный тип нельзя использовать в аргументах или операндах, для которых определено использование особого типа.

Процедура:

Следующий оператор SQL создает особый тип `t_educ` на основе типа smallint:

```
CREATE DISTINCT TYPE T_EDUC AS SMALLINT WITH COMPARISONS
```

Создав особый тип, можно использовать его для определения столбцов в операторе CREATE TABLE:

```
CREATE TABLE EMPLOYEE
   (EMPNO   CHAR(6) NOT NULL,
    FIRSTNME  VARCHAR(12)                    NOT NULL,
    LASTNAME  VARCHAR(15)                    NOT NULL,
    WORKDEPT   CHAR(3),
    PHONENO    CHAR(4),
    PHOTO      BLOB(10M)    NOT NULL,
    EDLEVEL    T_EDUC)
IN RESOURCE
```

При создании особого типа также генерируется поддержка преобразования типов между особым типом и исходным типом. Поэтому значение типа T_EDUC можно преобразовать в значение типа SMALLINT и наоборот.

Используя функции преобразований, можно преобразовать пользовательские типы в базовые типы данных и базовые типы данных в пользовательские. Для создания функции преобразования используется оператор CREATE TRANSFORM.

Для поддержки преобразований могут также использоваться оператор CREATE METHOD и расширения оператора CREATE FUNCTION.

Понятия, связанные с данным:

- “Пользовательский тип (UDT)” на стр. 134

Ссылки, связанные с данной темой:

- “CREATE DISTINCT TYPE statement” в *SQL Reference, Том 2*
- “CREATE TRANSFORM statement” в *SQL Reference, Том 2*
- “CREATE METHOD statement” в *SQL Reference, Том 2*
- “CREATE FUNCTION (Sourced or Template) statement” в *SQL Reference, Том 2*
- “User-defined types” в *SQL Reference, Том 1*
- “Data types” в *SQL Reference, Том 1*

Создание пользовательского структурированного типа

Структурированный тип - это пользовательский тип, содержащий один или несколько атрибутов, каждый из которых имеет собственные имя и тип данных. Структурированный тип может служить в качестве типа таблицы, каждый столбец которой получает свое имя и тип данных от одного из атрибутов этого структурированного типа.

Понятия, связанные с данным:

- “User-Defined Structured Types” в *Application Development Guide: Programming Server Applications*
- “Structured Type Hierarchies” в *Application Development Guide: Programming Server Applications*

Задачи, связанные с данной темой:

- “Defining Structured Types” в *Application Development Guide: Programming Server Applications*
- “Creating a Structured Type Hierarchy” в *Application Development Guide: Programming Server Applications*

Ссылки, связанные с данной темой:

- “CREATE TYPE (Structured) statement” в *SQL Reference, Том 2*
- “User-defined types” в *SQL Reference, Том 1*

Создание отображения типа

В системе объединения отображения типов позволяют отобразить конкретные типы данных в таблицах и производных таблицах источника данных на особые типы данных DB2. Отображение типов применяется для одного источника данных или же для диапазона (тип, версия) источников данных.

Для встроенных типов источников данных и встроенных типов DB2 существуют отображения типов по умолчанию. Новые (созданные вами) отображения типов будут перечислены в производной таблице SYSCAT.TYPEMAPPINGS.

Ограничения:

Нельзя создать отображение типов для типов данных большой объект, LONG VARCHAR/VARGRAPHIC, DATALINK, структурированных или особых типов.

Процедура:

Для создания отображений типов используется оператор CREATE TYPE MAPPING. Чтобы использовать этот оператор, необходимо обладать полномочиями SYSADM или DBADM для базы данных объединения.

Пример оператора, создающего отображение типа:

```
CREATE TYPE MAPPING MY_ORACLE_DEC FROM SYSIBM.DECIMAL(10,2)
TO SERVER ORACLE1 TYPE NUMBER([10..38],2)
```

Ссылки, связанные с данной темой:

- “CREATE TYPE MAPPING statement” в *SQL Reference, Том 2*
- “Data Type Mappings between DB2 and OLE DB” в *Application Development Guide: Programming Client Applications*

Создание производной таблицы

Производные таблицы создаются на основе одной или нескольких базовых таблиц, псевдонимов или производных таблиц и могут использоваться для получения данных вместо базовых таблиц. При изменении данных в производной таблице изменяются данные в самих базовых таблицах.

С помощью производной таблицы можно ограничить доступ к конфиденциальным данным, предоставив свободный доступ ко всем остальным данным.

При операциях вставки в производную таблицу, в которых в список выборки (SELECT) для производной таблицы прямо или косвенно входит имя столбца

идентификации базовой таблицы, применяются те же правила, что и для оператора INSERT, в котором прямо указывается столбец идентификации базовой таблицы.

Кроме описанного выше применения производных таблиц, их можно также использовать, чтобы:

- Менять таблицу, не меняя прикладные программы. Для этого на основе базовой таблицы создается производная таблица. Создание этой новой производной таблицы не влияет на работу прикладных программ, использующих базовую таблицу. Новые прикладные программы могут использовать созданную производную таблицу для задач, отличающихся от задач прикладных программ, использующих базовую таблицу.
- Суммировать значения в столбце, выбирать максимальные значения или вычислять средние значения.
- Обеспечить доступ к информации в одном или нескольких источниках данных. В операторе CREATE VIEW можно задать псевдонимы и создать глобальную производную таблицу, которая может объединять информацию из нескольких источников данных, расположенных в разных системах.

  Если для создания производной таблицы задаются псевдонимы и используется обычный синтаксис команды CREATE VIEW, будет выдано предупреждение о том, что для доступа к базовым объектам (объектам на источниках данных) будут использоваться ID аутентификации пользователей этой производной таблицы, а не ID аутентификации ее создателя. Чтобы это предупреждение не выдавалось, используйте ключевое слово FEDERATED.

Вместо создания производной таблицы можно использовать вложенное или общее табличное выражение для уменьшения затрат на поиск в каталоге и улучшения производительности.

Предварительные требования:

Производную таблицу можно создать только на основе существующей базовой таблицы, псевдонима или производной таблицы.

Ограничения:

Можно создать производную таблицу, используя в ее определении пользовательскую функцию. Однако для обновления такой производной таблицы после изменения функции потребуется отбросить эту таблицу и создать ее заново. Если от пользовательской функции зависит производная таблица, эту функцию нельзя отбросить.

Следующий оператор SQL создает производную таблицу с функцией в определении:

```
CREATE VIEW EMPLOYEE_PENSION (NAME, PENSION)
  AS SELECT NAME, PENSION(HIREDATE,BIRTHDATE,SALARY,BONUS)
  FROM EMPLOYEE
```

Пользовательская функция PENSION вычисляет пенсию, на которую имеет право работник, используя формулу, в которую входят его дата приема на работу (HIREDATE), дата рождения (BIRTHDATE), заработная плата (SALARY) и премия (BONUS).

Процедура:

Чтобы создать производную таблицу с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Производные таблицы**.
2. Щелкните правой кнопкой мыши по папке **Производные таблицы** и выберите из всплывающего меню пункт **Создать**.
3. Введите необходимую информацию и нажмите кнопку **OK**.

Чтобы создать производную таблицу из командной строки, введите команду:

```
CREATE VIEW <имя> (<столбец>, <столбец>, <столбец>)
   SELECT <имена_столбцов> FROM <имя_таблицы>
   WITH CHECK OPTION
```

Например, таблица сотрудников EMPLOYEE может содержать информацию о заработной плате, которая не должна быть общедоступной. Однако номер телефона работника должен быть доступен всем пользователям. В этом случае можно создать производную таблицу, содержащую только столбцы LASTNAME (фамилия) и PHONENO (номер телефона). Права на доступ к этой производной таблице могут быть предоставлены группе PUBLIC (то есть всем пользователям), в то время как права на доступ ко всей таблице EMPLOYEE можно ограничить, разрешив его только тем пользователям, которые имеют право получать информацию о заработной плате.

Используя производную таблицу, можно сделать доступным для прикладной программы часть данных таблицы и проверять вставляемые или изменяемые данные. Имена столбцов производной таблицы могут не совпадать с именами соответствующих столбцов исходных таблиц.

Используя производные таблицы, можно гибко определить, как будут выглядеть данные таблицы для программ и запросов конечных пользователей.

Следующий оператор SQL создает производную таблицу на основе таблицы EMPLOYEE, содержащую список всех сотрудников отдела A00 с личными номерами и номерами телефонов этих сотрудников:

```
CREATE VIEW EMP_VIEW (DA00NAME, DA00NUM, PHONENO)
   AS SELECT LASTNAME, EMPNO, PHONENO FROM EMPLOYEE
   WHERE WORKDEPT = 'A00'
   WITH CHECK OPTION
```

В первой строке этого оператора задается имя производной таблицы и определяются ее столбцы. Имя EMP_VIEW должно быть уникальным в своей схеме в SYSCAT.TABLES. Имя производной таблицы выглядит как имя таблицы, хотя сама она и не содержит данных. В этой производной таблице будет три столбца с именами DA00NAME, DA00NUM и PHONENO, которые соответствуют столбцам LASTNAME, EMPNO и PHONENO из таблицы EMPLOYEE. Каждому из перечисленных столбцов соответствует один столбец в списке выборки в операторе SELECT. Если имена столбцов не заданы, в производной таблице используются те же имена столбцов, что и для столбцов таблицы в операторе SELECT.

Вторая строка - это оператор SELECT, в котором описывается, какие значения должны быть выбраны из базы данных. Он может включать условия ALL, DISTINCT, FROM, WHERE, GROUP BY и HAVING. Имя или имена объектов данных, из которых выбираются столбцы для этой производной таблицы, должны быть заданы после условия FROM.

Условие WITH CHECK OPTION указывает, что для всех операций вставки или изменения для этой производной таблицы должна выполняться проверка на соответствие определению этой производной таблицы и что эти операции должны отвергаться, если они не удовлетворяют этому определению. Такой режим лучше обеспечивает целостность данных, но требует дополнительной обработки. Если такое условие опущено, операции вставки и изменения не будут проверяться на соответствие определению этой производной таблицы.

Следующий оператор SQL создает для таблицы EMPLOYEE ту же производную таблицу, используя условие SELECT AS:

```
CREATE VIEW EMP_VIEW
   SELECT LASTNAME AS DA00NAME,
          EMPNO AS DA00NUM,
          PHONENO
   FROM EMPLOYEE
   WHERE WORKDEPT = 'A00'
   WITH CHECK OPTION
```

Понятия, связанные с данным:

- “Views” в *SQL Reference, Том 1*
- “Привилегии таблиц и производных таблиц” на стр. 254
- “Управление доступом к данным с помощью производных таблиц” на стр. 267
- “Применение триггеров для обновления производных таблиц” на стр. 128

Задачи, связанные с данной темой:

- “Создание типизированной производной таблицы” на стр. 141
- “Удаление строк из таблицы или производной таблицы” на стр. 194
- “Изменение и отбрасывание производной таблицы” на стр. 221
- “Восстановление неработоспособных производных таблиц” на стр. 223

Ссылки, связанные с данной темой:

- “CREATE VIEW statement” в *SQL Reference, Том 2*
- “INSERT statement” в *SQL Reference, Том 2*

Сведения о создании производных таблиц

Типизированная производная таблица основывается на предопределенном структуризированном типе.

Создание типизированной производной таблицы

Процедура:

Для создания типизированной производной таблицы можно использовать оператор CREATE VIEW.

Понятия, связанные с данным:

- “Typed Views” в *Application Development Guide: Programming Server Applications*

Задачи, связанные с данной темой:

- “Creating Typed Views” в *Application Development Guide: Programming Server Applications*
- “Altering Typed Views” в *Application Development Guide: Programming Server Applications*
- “Dropping Typed Views” в *Application Development Guide: Programming Server Applications*

Ссылки, связанные с данной темой:

- “CREATE VIEW statement” в *SQL Reference, Том 2*

Создание материализованной таблицы запроса

Материализованная таблица запроса - это таблица, определение которой основано на результатах обработки запроса. Обычно материализованная таблица запроса содержит предварительные результаты, полученные исходя из данных, содержащихся в таблицах, на которых основано определение этой материализованной таблицы. Если компилятор SQL определит, что для

выполнения запроса эффективнее использовать не базовые таблицы, а материализованную таблицу запроса, то будет использована материализованная таблица, и результаты будут получены быстрее.

Ограничения:

По отношению к материализованной таблице запроса, обслуживаемой пользователем, действуют те же ограничения, что и по отношению к материализованной таблице запроса, обслуживаемой системой, со следующими исключениями:

- Для материализованной таблицы запроса можно выполнять операции INSERT, UPDATE и DELETE. Однако проверка базовых таблиц не выполняется. Ответственность за проверку правильности данных ложится на пользователя.
- Для такой материализованной таблицы запроса допустимы операции LOAD, EXPORT и IMPORT, а также операция репликации данных, однако проверка правильности данных не выполняется.
- Для такой материализованной таблицы запроса не применим оператор REFRESH TABLE.
- Для такой материализованной таблицы не применим оператор SET INTEGRITY ... IMMEDIATE CHECKED.
- Материализованная таблица запроса, обслуживаемая пользователем, должна быть определена с опцией REFRESH DEFERRED.

Материализованные таблицы запроса, определенные с опцией REFRESH DEFERRED, не применяются для оптимизации статического SQL.

Будьте осторожны, задавая для специального регистра CURRENT REFRESH AGE ненулевое значение. Материализованная таблица запроса может не отражать текущие значения из базовой таблицы, поэтому ее применение для оптимизации запроса может привести к тому, что результат выполнения запроса *не* будет соответствовать данным базовой таблицы. Это может быть приемлемо, если известно, что базовые данные не изменились, или если данные таковы, что для них допустима некоторая ошибка в результатах.

Если на основе допустимой *полной выборки* нужно создать новую базовую таблицу, задайте при создании этой таблицы ключевое слово DEFINITION ONLY. Созданная таблица будет рассматриваться не как материализованная таблица запроса, а как базовая таблица. Например, можно создать таблицы исключений, используемые в LOAD и SET INTEGRITY:

```
CREATE TABLE XT AS
   (SELECT T.*, CURRENT TIMESTAMP AS TIMESTAMP,CLOB(",32K)
   AS MSG FROM T) DEFINITION ONLY
```

Ниже перечислены некоторые основные ограничения, которые следует учитывать при работе с материализованными таблицами запроса:

1. Материализованную таблицу запроса нельзя изменять.
2. Если на основе базовой таблицы создана материализованная таблица запроса, то нельзя изменить длину столбца этой базовой таблицы.
3. В материализованную таблицу запроса нельзя импортировать данные.
4. Для материализованной таблицы запроса нельзя создать индекс уникальности.
5. Материализованную таблицу запроса нельзя создать на основе результатов выполнения запроса, в котором используются псевдонимы.

Процедура:

Для того чтобы реплицировать материализованную таблицу запроса на все узлы в среде многораздельных баз данных, создайте эту таблицу с опцией репликации. Такие таблицы называются “реплицированными материализованными таблицами запроса”.

В общем случае материализованная таблица запроса, или реплицированная материализованная таблица запроса, применяется для оптимизации запроса в том случае, если уровень изоляции этой таблицы не ниже уровня изоляции запроса. Например, если запрос выполняется с уровнем изоляции Стабильность на уровне указателя (CS), то для его оптимизации применяются только те реплицированные и обычные материализованные таблицы запроса, уровень изоляции которых равен CS или выше.

Для создания материализованной таблицы запроса применяется оператор CREATE TABLE с условием AS *полная-выборка* и опцией IMMEDIATE или REFRESH DEFERRED.

Для имен столбцов материализованной таблицы запроса можно задать уникальные имена. Список имен столбцов должен содержать столько же имен, сколько столбцов в таблице результатов этой полной выборки. Список имен столбцов должен быть задан, если таблица результатов полной выборки содержит повторяющиеся имена столбцов или столбцы без имен. Столбцы без имен образуются для констант, функций, выражений или операторов присваивания, для которых не заданы имена с помощью условия AS списка выборки. Если список столбцов не задан, столбцы этой таблицы будут иметь те же имена, что и столбцы набора результатов полной выборки.

При работе с большой базой данных или хранилищем данных часто применяются пользовательские приложения, обслуживающие и загружающие материализованные таблицы запроса, обслуживаемые пользователем. При создании материализованной таблицы запроса можно указать, кто ее будет обслуживать: система или пользователь. По умолчанию таблица обслуживается

системой. Эту опцию можно явно задать с помощью условия MAINTAINED BY SYSTEM. Для того чтобы таблица обслуживалась пользователем, укажите условие MAINTAINED BY USER.

При создании материализованной таблицы запроса, обслуживаемой системой, можно указать, следует ли автоматически обновлять эту таблицу при изменении базовой таблицы, или же таблица будет обновляться только с помощью оператора REFRESH TABLE. Для того чтобы материализованная таблица запроса автоматически обновлялась при изменении базовой таблицы (или таблиц), укажите ключевое слово REFRESH IMMEDIATE. Такое немедленное обновление удобно, когда:

- Необходимо, чтобы в запросе применялись текущие данные
- Базовая таблица (или таблицы) изменяется редко
- Обновление сводной таблицы не занимает много времени.

В этом случае материализованная таблица запроса предоставляет предварительные результаты. Если обновление материализованной таблицы запроса должно откладываться на более позднее время, укажите ключевое слово REFRESH DEFERRED. В этом случае материализованная таблица запроса **не** будет отражать изменения, внесенные в базовые таблицы. Такие материализованные таблицы запроса следует использовать в том случае, когда точность данных не существенна. Например, при выполнении запросов DSS может применяться материализованная таблица запроса, содержащая унаследованные данные.

Материализованная таблица запроса, определенная с опцией REFRESH DEFERRED, может использоваться вместо запроса, когда он:

- Удовлетворяет ограничениям на полную выборку для сводной таблицы с немедленным обновлением, за исключением:
  - Не требуется, чтобы список SELECT содержал COUNT(*) или COUNT_BIG(*)
  - Список SELECT может содержать функции столбцов MAX и MIN
  - Разрешено условие HAVING.

В специальном регистре CURRENT REFRESH AGE можно задать интервал времени, в течение которого материализованная таблица запроса, определенная с опцией REFRESH DEFERRED, может применяться для обработки динамических запросов, прежде чем ее потребуется обновить. Для задания значения специального регистра CURRENT REFRESH AGE можно использовать оператор SET CURRENT REFRESH AGE.

Для того чтобы материализованная таблица запроса с отложенным обновлением могла применяться при обработке динамических запросов без ограничения по времени, укажите в специальном регистре CURRENT REFRESH AGE значение ANY или 99999999999999. Все девятки - это максимальное

значение, которое может иметь этот специальный регистр, содержащий значение длительности времени типа данных DECIMAL(20,6). Нулевое значение указывает, что для оптимизации запроса могут применяться только те материализованные таблицы запроса, в определении которых задана опция REFRESH IMMEDIATE. В этом случае материализованные таблицы запроса, определенные с опцией REFRESH DEFERRED, не используются для оптимизации.

Материализованные таблицы запроса, определенные с опцией REFRESH IMMEDIATE, могут применяться как для статических, так и для динамических запросов. Для таких таблиц не нужно задавать специальный регистр CURRENT REFRESH AGE.

Материализованная таблица запроса применяется для маршрутизации запросов в том случае, если в ее определении было задано условие ENABLE QUERY OPTIMIZATION. В случае отложенной материализованной таблицы запроса также необходимо, чтобы специальному регистру CURRENT REFRESH AGE было присвоено значение ANY. Однако для материализованных таблиц запроса, обслуживаемых пользователем, специальный регистр CURRENT REFRESH AGE - это не лучший способ перенаправить запросы. Типы кэшированных данных, доступных для маршрутизации, задаются в специальном регистре CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION.

С течением времени исходные данные изменятся, поэтому материализованная таблица запроса будет содержать неточные данные. Нужно будет использовать оператор REFRESH TABLE.

Понятия, связанные с данным:

- “Isolation levels” в *SQL Reference, Том 1*

Задачи, связанные с данной темой:

- “Изменение свойств материализованной таблицы запроса” на стр. 212
- “Обновление данных в материализованной таблице запроса” на стр. 213
- “Отбрасывание материализованной таблицы запроса и промежуточной таблицы” на стр. 224

Ссылки, связанные с данной темой:

- “CREATE TABLE statement” в *SQL Reference, Том 2*
- “REFRESH TABLE statement” в *SQL Reference, Том 2*
- “SET CURRENT REFRESH AGE statement” в *SQL Reference, Том 2*
- “CURRENT REFRESH AGE special register” в *SQL Reference, Том 1*
- “CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION special register” в *SQL Reference, Том 1*

- “SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION statement” в *SQL Reference, Том 2*

Создание промежуточной таблицы

Промежуточная таблица применяется для инкрементного обновления отложенной материализованной таблицы запроса. В промежуточной таблице накапливаются изменения, которые нужно применить к материализованной таблице запроса для ее синхронизации с содержимым базовых таблиц. Применение промежуточных таблиц позволяет избежать возникновения большого числа конфликтов при попытке установить блокировку, которые обычно возникают в случае обработки запроса на немедленное обновление материализованной таблицы запроса. Кроме того, при выполнении операции REFRESH TABLE не требуется заново создавать всю материализованную таблицу запроса.

Материализованные таблицы запроса позволяют значительно сократить время ответа для сложных запросов, особенно тех из них, которые применяют следующие операции:

- Группировка данных по одному или нескольким столбцам
- Объединение и группировка данных в группе таблиц
- Обращение к часто используемым данным
- Повторное разбиение таблицы или ее части в среде многораздельной базы данных

Ограничения:

Ниже перечислены основные ограничения, которые следует учитывать при работе с промежуточными таблицами:

1. Запрос, применяемый для определения промежуточной таблицы, должен допускать возможность инкрементного обслуживания, то есть он должен следовать тем же правилам, что и материализованная таблица запроса с опцией немедленного обновления.
2. Промежуточную таблицу можно создать только в том случае, если задана опция отложенного обновления. Материализованная таблица запроса, связанная с промежуточной таблицей, также определяется в запросе. Для этой материализованной таблицы запроса должна быть задана опция REFRESH DEFERRED.
3. Если для обновления используются промежуточные таблицы, можно применить только те изменения, которые были внесены вплоть до текущего момента.

Процедура:

Если промежуточная таблица содержит несогласованные или неполные данные, либо находится в состоянии ожидания, то с ее помощью нельзя выполнить инкрементное обновление связанной материализованной таблицы запроса, пока не будет выполнена какая-либо другая операция. При выполнении этой операции содержимое промежуточной таблицы будет синхронизировано с содержимым связанной материализованной таблицы запроса и ее базовых таблиц. Кроме того, промежуточная таблица запроса будет выведена из состояния ожидания. После обновления материализованной таблицы запроса содержимое промежуточной таблицы очищается, а сама промежуточная таблица переходит в обычное состояние. Промежуточную таблицу можно сократить, выполнив оператор SET INTEGRITY с соответствующими опциями. После сокращения промежуточная таблица станет несогласованной. Например, следующий оператор сокращает промежуточную таблицу STAGTAB1:

```
SET INTEGRITY FOR STAGTAB1 PRUNE;
```

После создания промежуточная таблица переводится в состояние ожидания и помечается как несогласованная или неполная относительно содержимого базовых таблиц и связанной материализованной таблицы запроса. Для того чтобы в промежуточной таблице начали накапливаться изменения, вносимые в базовые таблицы, ее необходимо вывести из состояния ожидания и сделать согласованной с содержимым базовых таблиц. Пока промежуточная таблица находится в состоянии ожидания, все попытки внести изменения в базовые таблицы или обновить соответствующую материализованную таблицу запроса будут заканчиваться неудачно.

Промежуточную таблицу можно вывести из состояния ожидания несколькими способами, например::

- SET INTEGRITY FOR <имя промежуточной таблицы> STAGING IMMEDIATE UNCHECKED
- SET INTEGRITY FOR <имя промежуточной таблицы> IMMEDIATE CHECKED

Задачи, связанные с данной темой:

- “Создание материализованной таблицы запроса” на стр. 141
- “Изменение свойств материализованной таблицы запроса” на стр. 212
- “Обновление данных в материализованной таблице запроса” на стр. 213
- “Отбрасывание материализованной таблицы запроса и промежуточной таблицы” на стр. 224

Ссылки, связанные с данной темой:

- “SET INTEGRITY statement” в *SQL Reference, Том 2*

Создание алиаса

Алиас - это метод косвенного задания в операторе SQL таблицы, псевдонима или производной таблицы, позволяющий исключить зависимость этого оператора SQL от полного имени таблицы или производной таблицы. При изменении имени таблицы или производной таблицы потребуется изменить только определение алиаса. Алиас можно создать для другого алиаса. Алиас может использоваться в определении производной таблицы или триггера и в любом операторе SQL, кроме определений проверочных ограничений таблицы, в которых может использоваться имя существующей таблицы или производной таблицы.

Предварительные требования для установки:

Алиас можно определить для таблицы, производной таблицы или алиаса, которые еще не существуют в момент определения. Однако они должны существовать в момент компиляции оператора SQL, использующего этот алиас.

Ограничения:

Алиас может использоваться везде, где может использоваться имя таблицы; алиас может ссылаться на другой алиас, если в цепи таких ссылок не возникает циклических или повторяющихся ссылок.

Алиас не может совпадать с существующим именем таблицы, производной таблицы или алиасом и может ссылаться только на таблицу в той же базе данных. Имя таблицы или производной таблицы, заданное в операторе CREATE TABLE или CREATE VIEW, не может совпадать с именем алиаса в той же схеме.

Для создания алиаса не требуются специальные полномочия, если он создается в схеме, владельцем которой является текущий ID авторизации; в противном случае требуются полномочия DBADM.

Если алиас или объект, на который он ссылается, отброшен, все пакеты, зависящие от этого алиаса, отмечаются как недействительные, а все производные таблицы и триггеры, зависящие от этого алиаса, отмечаются как неработоспособные.

Процедура:

Чтобы создать алиас с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Алиасы**.
2. Щелкните правой кнопкой мыши по папке **Алиасы** и выберите из всплывающего меню пункт **Создать**.
3. Введите необходимую информацию и нажмите кнопку **OK**.

Чтобы создать алиас из командной строки, введите команду:

```
CREATE ALIAS <имя_алиаса> FOR <имя_таблицы>
```

Во время компиляции вместо алиаса в оператор подставляется имя таблицы или производной таблицы. Если по алиасу (или цепочке алиасов) невозможно определить имя таблицы или производной таблицы, возникает ошибка. Например, если WORKERS - это алиас для таблицы EMPLOYEE, во время компиляции:

```
SELECT * FROM WORKERS
```

превратится в

```
SELECT * FROM EMPLOYEE
```

Следующий оператор SQL создает алиас WORKERS для таблицы EMPLOYEE:

```
CREATE ALIAS WORKERS FOR EMPLOYEE
```

Примечание: В DB2 for OS/390 или z/Series используется два разных понятия: алиас и синоним. Эти понятия отличаются от алиасов в DB2 Universal Database:

- Алиасы в DB2 for OS/390 или z/Series:
  - Требуют для создания специальных полномочий или привилегии
  - Не могут ссылаться на другие алиасы.
- Синонимы в DB2 for OS/390 или z/Series:
  - Могут использоваться только их создателем
  - Всегда используются без спецификаторов
  - Отбрасываются при отбрасывании таблицы, на которую они ссылаются
  - Используют отдельное от имен таблиц и производных таблиц пространство имен.

Понятия, связанные с данным:

- “Aliases” в *SQL Reference, Том 1*

Ссылки, связанные с данной темой:

- “CREATE ALIAS statement” в *SQL Reference, Том 2*

Индекс, расширение индекса и спецификация индекса

Индекс - это список позиций строк, отсортированный по содержимому одного или нескольких указанных столбцов. Индексы обычно используются для ускорения доступа к таблице. Но можно использовать их и для поддержания логической структуры данных. Например, *индекс уникальности* не допускает повторения значений в столбцах, гарантируя тем самым, что таблица не содержит совпадающих строк. Индексы могут также создаваться для задания восходящего или нисходящего порядка значений в столбце.

Расширение индекса - это индексный объект, применяемый вместе с индексами, содержащими столбцы структурированных или особых типов.

Спецификация индекса - это компонент метаданных. Она сообщает оптимизатору, что для задаваемого псевдонимом объекта источника данных (таблицы или производной таблицы) существует индекс. Спецификация индекса не содержит списка позиций строк, она представляет собой лишь описание индекса. Оптимизатор использует спецификацию индекса для оптимизации доступа к объекту, заданному псевдонимом. Спецификация индекса генерируется при создании псевдонима в том случае, если в источнике данных для базовой таблицы есть индекс, формат которого поддерживается DB2®.

Примечание: Если нужно, создайте спецификации индексов для псевдонимов таблиц или производных таблиц, определенных над одной таблицей.

Создайте индекс или спецификацию индекса вручную, если:

- Это может улучшить производительность. Например, если нужно, чтобы оптимизатор использовал конкретную таблицу или псевдоним в качестве внутренней таблицы объединения с вложенным циклом, создайте спецификацию индекса для объединяющего столбца, если индекс не существует.
- Индекс для базовой таблицы был добавлен после того, как был создан псевдоним.

Спецификации индексов можно создать и для базовых таблиц, не имеющих индексов (при выполнении оператора CREATE INDEX DB2 не проверяет наличие удаленного индекса). Спецификация индексов не обеспечивает уникальность строк, даже если задано ключевое слово UNIQUE.

Советчик по индексам DB2 - это мастер, помогающий выбрать оптимальный набор индексов. Этот мастер можно вызвать из Центра управления. Соответствующая утилита имеет имя *db2advis*.

Индекс определяется по столбцам базовой таблицы. Он может быть определен создателем таблицы или пользователем, который знает, что требуется прямой доступ к определенным столбцам. Для первичного ключа автоматически создается первичный ключ индекса, если не существует пользовательский индекс.

Для конкретной базовой таблицы может быть определено любое число индексов; они могут положительно влиять на производительность запросов. Однако чем больше существует индексов, тем больше изменений приходится вносить менеджеру баз данных при выполнении операций изменения, удаления и вставки. Создание большого числа индексов для таблицы, для которой выполняется много операций изменения данных, может замедлить обработку запросов. Поэтому используйте индексы, только если очевидны их преимущества для частых обращений.

Максимальное число столбцов в индексе - 16. Для индекса типизированной таблицы максимальное число столбцов - 15. Максимальная длина ключа индекса - 1024 байта. Как отмечено выше, большое число ключей индексов для таблицы может замедлить обработку запросов. Длинные ключи индекса также могут замедлять обработку запросов.

Ключ индекса - это столбец или набор столбцов, на которых определен индекс; эти столбцы определяют полезность индекса. Хотя при создании ключа индекса порядок его столбцов не имеет значения, он учитывается оптимизатором при выборе индекса для оптимизации запроса.

Если индекс определяется для пустой таблицы, индекс создается, но записи индекса будут создаваться при загрузке таблицы или вставке в нее строк. Если таблица содержит данные, менеджер баз данных создает записи индекса при выполнении оператора CREATE INDEX.

При использовании *индекса кластеризации* вставляемые новые строки физически располагаются близко к существующим строкам с похожими значениями ключа. Это улучшает производительность выполнения запросов, поскольку повышают показатель последовательности доступа к страницам данных и эффективность предварительной выборки.

Если нужно, чтобы индекс первичного ключа был индексом кластеризации, не задавайте первичный ключ в операторе CREATE TABLE. После того, как создан первичный ключ, связанный с ним индекс нельзя изменить. Поэтому выполните оператор CREATE TABLE без условия, задающего первичный ключ. Затем выполните оператор CREATE INDEX, задав атрибуты кластеризации. Наконец, используйте оператор ALTER TABLE, чтобы добавить первичный ключ, соответствующий только что созданному индексу. Этот индекс будет использоваться в качестве индекса первичного ключа.

Кластеризация обычно более эффективна, если индекс кластеризации является индексом уникальности.

Столбцы, которые не являются частью ключа индекса уникальности, но данные которых хранятся/поддерживаются в индексе, называются *включенными* столбцами. Включенные столбцы можно задать только для индексов уникальности. При создании индекса с включенными столбцами только столбцы ключа уникальности сортируются и проверяются на уникальность. Использование включенных столбцов улучшает производительность получения данных, если используется этот индекс.

Менеджер баз данных хранит индексы в виде структур деревьев типа B+, нижний уровень которых составляют конечные узлы. В конечных узлах или страницах хранятся сами значения ключей индекса. При создании индекса можно включить функцию фонового слияния конечных страниц индекса. Фоновая дефрагментация индекса предотвращает такую ситуацию, в которой после удаления или обновления большого объема данных на многих конечных страницах индекса остается всего по несколько ключей. Если в описанном случае фоновая дефрагментация индекса не применяется, то память можно освободить только путем реорганизации данных или индекса. Перед тем как включать функцию фоновой дефрагментации страниц индекса, сравните стоимость проверки наличия свободной памяти для слияния страниц и собственно слияния с тем выигрышем, который можно получить от освобождения памяти, а также со стоимостью реорганизации.

Примечания:

1. Страницы, освобожденные в ходе фоновой дефрагментации, могут применяться только для других индексов той же таблицы. При полной реорганизации освобождаемые страницы могут использоваться для других объектов (при работе с хранением, управляемым базой данных) или же становятся свободным дисковым пространством (при работе с хранением, управляемым системой). Кроме того, во время фоновой дефрагментации освобождаются только конечные страницы индекса, тогда как во время полной реорганизации размер индекса уменьшается до минимума за счет сокращения числа конечных и неконечных страниц, а также числа уровней индекса.
2. В индексах, созданных в версии младше версии 8, ключ физически удаляется с конечной страницы при удалении или обновлении строки таблицы. В индексах типа 2 при удалении или обновлении строки ключ помечается как удаленный. Физически он удаляется со страницы только при выполнении очистки через некоторое время после фиксации операции удаления или обновления. Такая очистка может выполняться следующей транзакцией, которая изменяет страницу, содержащую удаленный ключ. Для того чтобы явно активировать операцию очистки, вызовите утилиту REORG INDEXES с опцией CLEANUP ONLY [ALL | PAGES].

Для создания индексов для таблиц в многораздельной базе данных используется тот же оператор CREATE INDEX. Они создаются на многих разделах на основе ключа разделения этой таблицы. Индекс таблицы состоит из локальных индексов этой таблицы на каждом узле в группе узлов. Учтите, что индексы уникальности, определяемые в многораздельной среде, должны быть надмножеством ключа разделения.

Понятия, связанные с данным:

- “Indexes” в *SQL Reference, Том 1*
- “Применение индекса” на стр. 155
- “Применение оператора CREATE INDEX” на стр. 156
- “Создание пользовательского расширенного типа индекса” на стр. 160
- “Привилегии индексов” на стр. 258

Задачи, связанные с данной темой:

- “Включение параллелизма при создании индексов” на стр. 12
- “Создание индекса” на стр. 153
- “Изменение имени таблицы или индекса” на стр. 214
- “Отбрасывание индекса, расширения индекса и спецификации индекса” на стр. 226

Ссылки, связанные с данной темой:

- “CREATE INDEX statement” в *SQL Reference, Том 2*
- “CREATE INDEX EXTENSION statement” в *SQL Reference, Том 2*

Сведения о создании индекса, расширения индекса или спецификации индекса

Пользователь может как работать с индексами, созданными менеджером базы данных, так и создать собственный индекс.

Создание индекса

Индекс - это один или несколько ключей, каждый из которых указывает на строки в таблице. Индекс позволяет эффективнее получать доступ к строкам в таблице, создавая прямой путь к данным через указатели.

Процедура:

Совет по улучшению производительности: Если вы собираетесь выполнить следующий набор задач:

1. Создать таблицу
2. Загрузить таблицы

3. Создать индекс (без опции COLLECT STATISTICS)
4. Выполнить команду RUNSTATS

Если вы собираетесь выполнить следующий набор задач:

1. Создать таблицу
2. Загрузить таблицы
3. Создать индекс (с опцией COLLECT STATISTICS)

то эти задачи лучше выполнять в следующем порядке:

1. Создать таблицу
2. Создать индекс
3. Загрузить таблицу, используя опцию `statistics yes`.

После создания индексов они поддерживаются в правильном состоянии. Поэтому когда прикладные программы используют значения ключа для прямого доступа и обработки строк таблицы, для прямого доступа к строкам может использоваться индекс, содержащий значения этого ключа. Это существенно, поскольку строки физически хранятся в базовой таблице в неупорядоченном виде.

Вы можете создать таблицу многомерной кластеризации (MDC). В этом случае будет создан блочный индекс. В отличие от обычного индекса, который указывает на отдельные строки таблицы, блочный индекс указывает на блоки, или экстенты, данных, поэтому он значительно меньше обычного индекса. Блочные индексы хранятся в том же табличном пространстве, что и обычные индексы.

Если не используется индекс кластеризации, при вставке строки она помещается в наиболее подходящее положение хранения, содержащее достаточно свободного места. Если у таблицы нет индексов, при поиске в ней строк, удовлетворяющих конкретному условию выборки, сканируется вся таблица. Индекс позволяет получать данные, не выполняя долгого последовательного поиска.

Данные индексов могут храниться в том же табличном пространстве, что и данные таблицы, или же в отдельном табличном пространстве для данных индексов. Табличное пространство для хранения данных индексов задается при создании таблицы.

Чтобы создать индекс с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Индексы**.
2. Щелкните правой кнопкой мыши по папке **Индексы** и выберите из всплывающего меню пункт **Создать —> Индекс при помощи мастера**.
3. Для завершения задания выполните шаги в мастере.

Чтобы создать индекс из командной строки, введите команду:

```
CREATE INDEX <имя> ON <имя_таблицы> (<имя_столбца>)
```

Понятия, связанные с данным:

- “Optimizing Load Performance” в *Data Movement Utilities Guide and Reference*
- “Применение индекса” на стр. 155
- “Применение оператора CREATE INDEX” на стр. 156
- “Привилегии индексов” на стр. 258

Задачи, связанные с данной темой:

- “Изменение имени таблицы или индекса” на стр. 214
- “Отбрасывание индекса, расширения индекса и спецификации индекса” на стр. 226

Ссылки, связанные с данной темой:

- “CREATE INDEX statement” в *SQL Reference, Том 2*

Применение индекса

Индекс никогда не используется прикладной программой напрямую. Решение о том, нужно ли использовать индекс, и если да, то какой из существующих индексов лучше выбрать, принимает оптимизатор.

Лучше всего использовать для таблицы индекс, который:

- Использует скоростные диски
- Сильно кластеризован
- Состоит из небольшого числа столбцов
- Применяет столбцы с большой мощностью

Понятия, связанные с данным:

- “Рекомендации по планированию индексов” в *Руководство администратора: Производительность*
- “Рекомендации по повышению производительности с помощью индексов” в *Руководство администратора: Производительность*

- “Доступ к данным с помощью просмотра индекса” в *Руководство администратора: Производительность*
- “Управление обычными таблицами и индексами” в *Руководство администратора: Производительность*
- “Управление таблицами и индексами для таблиц MDC” в *Руководство администратора: Производительность*

Применение оператора CREATE INDEX

Можно создать индекс, допускающий повторение значений (неуникальный индекс), чтобы обеспечить эффективное получение столбцов, не входящих в первичный ключ, и разрешить индексируемым столбцам содержать повторяющиеся значения.

Следующий оператор SQL создает неуникальный индекс с именем LNAME для столбца LASTNAME таблицы EMPLOYEE, отсортированный в возрастающем порядке:

```
CREATE INDEX LNAME ON EMPLOYEE (LASTNAME ASC)
```

Следующий оператор SQL создает уникальный индекс для столбца, содержащего номер телефона:

```
CREATE UNIQUE INDEX PH ON EMPLOYEE (PHONENO DESC)
```

Индекс уникальности обеспечивает отсутствие повторяющихся значений в индексируемых столбцах. В конце выполнения операторов SQL, изменяющих строки или вставляющих новые строки, применяется ограничение. Этот тип индекса нельзя создать, если какие-либо из его столбцов уже содержат повторяющиеся значения.

Ключевое слово ASC располагает записи индекса в восходящем порядке по значениям столбцов, а ключевое слово DESC - в нисходящем порядке. По умолчанию используется восходящий порядок.

Вы можете создать индекс уникальности для двух столбцов, один из которых является включенным столбцом. Первичный ключ задается для столбца, не являющегося включенным столбцом. Оба столбца указываются в каталоге как первичные ключи для одной и той же таблицы. Обычно же для одной таблицы существует только один первичный ключ.

Условие INCLUDE задает дополнительные столбцы, добавляемые в набор столбцов индексных ключей. Никакие столбцы, включенные с этим условием, не используются для обеспечения уникальности. Включенные столбцы могут повысить производительность для некоторых запросов за счет доступа только к индексу. Эти столбцы не должны быть столбцами, которые используются для обеспечения уникальности (иначе вы получите сообщение об ошибке SQLSTATE

42711). Для всех столбцов в ключе уникальности и в индексе применяются ограничения на число столбцов и на сумму атрибутов длин.

Выполняется проверка по выявлению совпадения существующего индекса с определением первичного ключа (с игнорированием столбцов INCLUDE в индексе). Определения индекса считается совпадающими, если в них задан один и тот тот же набор столбцов без учета их порядка или спецификаций направления (по возрастанию или по убыванию). При обнаружении совпадающего определения индекса в описании индекса указывается, что индекс является первичным индексом (как того требует система), и после обеспечения уникальности индекс становится индексом уникальности (если он не был им).

Вот почему для одной таблицы могут существовать несколько первичных ключей, как указано в каталоге.

При работе со структурированным типом может понадобиться создание пользовательских типов индексов. Для этого требуются средства определения функций поддержки индекса, поиска в индексе и применения индекса.

Следующий оператор SQL создает индекс кластеризации с именем INDEX1 для столбца LASTNAME таблицы EMPLOYEE:

```
CREATE INDEX INDEX1 ON EMPLOYEE (LASTNAME) CLUSTER
```

Для эффективного использования внутренней памяти базы данных используются индексы кластеризации с параметром PCTFREE, связанные с оператором ALTER TABLE, что позволяет вставлять новые данные в правильные страницы. При вставке данных в правильные страницы поддерживается порядок кластеризации. Обычно чем больше операций вставки (INSERT) выполняется для таблицы, тем большее значение PCTFREE (для этой таблицы) потребуется для поддержки кластеризации. Поскольку такой индекс определяет порядок размещения данных на физических страницах, для конкретной таблицы можно определить только один индекс кластеризации.

С другой стороны, если значения ключа индекса для новых строк всегда больше существующих значений этого ключа, при заданном атрибуте кластеризации таблицы система будет пытаться помещать новые значения в конец таблицы. Наличие свободного пространства на других страницах может несколько мешать кластеризации. В этом случае вместо того, чтобы использовать индекс кластеризации и задать для таблицы большое значение PCTFREE, лучше перевести таблицу в режим добавления. Для перевода таблицы в режим добавления можно использовать оператор ALTER TABLE APPEND ON.

То же самое относится и к строкам, размер которых был увеличен операцией UPDATE и которые не умещаются на старое место.

Созданный с использованием параметра ALLOW REVERSE SCANS в операторе CREATE INDEX одиночный индекс можно просматривать в прямом или обратном направлении. Иными словами, такие индексы поддерживают просмотр как в направлении, заданном при их создании, так и в противоположном (обратном) направлении. Оператор может выглядеть следующим образом:

```
CREATE INDEX имя-индекса ON имя-таблицы (имя-столбца DESC) ALLOW REVERSE SCANS
```

В этом случае индекс (`имя-индекса`) создается на основе значений столбца (`имя-столбца`), отсортированных по убыванию (`DESC`). Возможность обратного просмотра означает, что несмотря на то, что индекс столбца определен для просмотра в порядке убывания, просмотр может вестись и в порядке возрастания (в обратном порядке). Действительное направление использования индекса выбираете не вы, а оптимизатор при создании и принятии планов доступа.

Условие MINPCTUSED оператора CREATE INDEX задает порог минимального количества используемого пространства на конечных страницах индекса. Если это условие задано, то будет применяться фоновая дефрагментация индекса. Это означает, что если после физического удаления ключа с конечной страницы индекса доля занятой памяти на странице станет меньше указанного порогового значения, то будет проверено, нельзя ли объединить текущую страницу с соседней конечной страницей в одну страницу.

Например, следующий оператор SQL создает индекс, для которого включена функция фоновой дефрагментации:

```
CREATE INDEX LASTN ON EMPLOYEE (LASTNAME) MINPCTUSED 20
```

Если после физического удаления ключа со страницы этого индекса оставшиеся ключи будут занимать не более двадцати процентов страницы индекса, то будет предпринята попытка удалить эту страницу, объединив оставшиеся ключи этой страницы с ключами, расположенными на одной из соседних страниц индекса. Если ключи из этих двух страниц умещаются на одной странице, выполняется слияние страниц и одна страница индекса удаляется.

Во время создания индекса с помощью оператора CREATE INDEX разрешается читать и записывать данные в базовую таблицу и ранее созданные индексы. Для того чтобы запретить доступ к таблице на время создания индекса, заблокируйте таблицу перед созданием индекса с помощью команды LOCK TABLE. Индекс создается путем просмотра базовой таблицы. Все изменения, внесенные в таблицу во время создания индекса, применяются к индексу после его создания. После применения всех изменений таблица стабилизируется, а индекс становится доступным для использования.

При создании индекса уникальности убедитесь, что таблица не содержит одинаковых ключей, и такие ключи не появятся во время создания индекса в

результате выполнения операций вставки. Во время создания индекса используется алгоритм отложенного выявления дубликатов ключей, поэтому при наличии одинаковых ключей это будет обнаружено только в конце процедуры создания индекса, после чего возникнет сбой.

Условие PCTFREE оператора CREATE INDEX задает, какой процент свободного места должен оставляться свободным на каждой страницы индекса при его построении. Если на страницах индекса оставляется больше свободного пространства, реже будет выполняться разбиение страниц. Это уменьшает потребность в реорганизации таблицы, восстанавливающей последовательный порядок страниц, который улучшает эффективность предварительной выборки. Предварительная выборка - важный способ повышения производительности. Как сказано выше, если добавляемые значения ключа всегда больше существующих, можно уменьшить значение условия PCTFREE оператора CREATE INDEX. Это ограничит объем неиспользуемого пространства, зарезервированного на каждой странице.

Во время создания индекса может быть собрана статистика по этому индексу. До выполнения оператора CREATE INDEX доступна только статистика по значениям ключей и физическая статистика. Если во время выполнения оператора CREATE INDEX будет собрана статистика по индексу, вам не потребуется вызывать утилиту RUNSTATS сразу после выполнения этого оператора.

Например, при выполнении следующего оператора SQL будет создан индекс и собрана основная статистика для индекса:

```
CREATE INDEX IDX1 ON TABL1 (COL1) COLLECT STATISTICS
```

Базовая таблица (или таблицы) реплицируемой сводной таблицы должна иметь индекс уникальности и столбцы ключа этого индекса должны использоваться в запросе, определяющем эту реплицируемую сводную таблицу.

В случае внутриразделного параллелизма можно улучшить производительность создания индекса, используя для выполняемых при создании индекса операций сканирования и сортировки данных несколько процессоров. Чтобы разрешить использование нескольких процессоров, задайте для *intra_parallel* значение YES(1) или ANY(-1). Число процессоров, используемых при операции создания индекса, определяется системой и не зависит от параметров конфигурации *dft_degree* или *max_querydegree*, степени параллелизма времени выполнения программы или степени параллелизма при компиляции оператора SQL.

В многораздельных базах данных уникальные индексы должны определяться как надмножества ключа разделения.

Понятия, связанные с данным:

- “Рекомендации по повышению производительности с помощью индексов” в *Руководство администратора: Производительность*
- “Реорганизация индекса” в *Руководство администратора: Производительность*
- “Управление обычными таблицами и индексами” в *Руководство администратора: Производительность*
- “Фоновая дефрагментация индекса” в *Руководство администратора: Производительность*
- “Управление таблицами и индексами для таблиц MDC” в *Руководство администратора: Производительность*

Задачи, связанные с данной темой:

- “Изменение атрибутов таблицы” на стр. 209

Ссылки, связанные с данной темой:

- “Параметр конфигурации Максимальная степень параллелизма запросов - max_querydegree” в *Руководство администратора: Производительность*
- “Параметр конфигурации Разрешение внутриразделького параллелизма - intra_parallel” в *Руководство администратора: Производительность*
- “Параметр конфигурации Степень по умолчанию - dft_degree” в *Руководство администратора: Производительность*
- “CREATE INDEX statement” в *SQL Reference, Том 2*

Создание пользовательского расширенного типа индекса

Для поддержки пользовательских типов индексов DB2® Universal Database позволяет создавать и применять собственные основные компоненты, определяющие способ использования индекса. Компоненты, которые можно заменить:

- Поддержка индекса. Обеспечивает возможность отображения содержимого столбца индекса на ключи индекса. Такое отображение осуществляется пользовательской функцией отображения. В расширенном индексе может участвовать ровно один столбец структурированного типа. В отличие от обычного индекса, расширенный индекс может содержать несколько записей индекса для каждой строки. Наличие нескольких записей индекса каждой строки позволяет сохранить текстовый документ как объект с отдельными записями индекса для каждого ключевого слова документа.
- Использование индекса. Позволяет разработчику прикладных программ связать условия фильтрации (предикаты диапазона) с пользовательской функцией, работа которой была бы в противном случае непонятной для оптимизатора. Это позволяет DB2 избежать отдельных вызовов

пользовательской функции для каждой строки и тем самым избежать переключений контекстов между клиентом и сервером, что значительно улучшает производительность.

Примечание: Чтобы оптимизатор мог использовать пользовательскую функцию, она должна быть детерминированной и не должна допускать побочных эффектов.

Можно также задать необязательную функцию фильтрации данных. Оптимизатор использует этот фильтр для блока выбранных данных перед выполнением пользовательской функции.

Только для столбцов структурированных или особых типов может использоваться расширение индекса для создания пользовательского расширенного типа индекса для таких объектов. Пользовательский расширенный тип индекса не должен:

- Быть определен с индексами кластеризации
- Содержать столбцы INCLUDE.

Понятия, связанные с данным:

- “Сведения о поддержке индекса” на стр. 161
- “Сведения о поиске в индексе” на стр. 162
- “Сведения о применении индекса” на стр. 163
- “Сценарий определения расширения индекса” на стр. 165

Сведения о создании пользовательских расширенных типов индекса

В этом разделе рассмотрены различные аспекты создания пользовательских расширенных типов индекса.

Сведения о поддержке индекса

С помощью оператора CREATE INDEX EXTENSION определяются два компонента, выполняющие эти операции для индекса.

Поддержка индекса - это процесс преобразования содержимого столбца индекса (или исходного ключа) в индексный ключ назначения. Для определения этого процесса преобразования используется табличная функция, ранее определенная в базе данных.

Условие FROM SOURCE KEY задает структурированный тип данных или особый тип для исходного столбца ключа, поддерживаемого этим расширением индекса. Для исходного столбца ключа задается и связывается с ним одно имя параметра и тип данных.

Условие GENERATE KEY USING задает пользовательскую табличную функцию, используемую для генерации ключа индекса. Выходное значение этой функции должно быть задано в условии TARGET KEY. Выходное значение этой функции может также использоваться в качестве входного значения для функции фильтрации, заданного в условии FILTER USING.

Понятия, связанные с данным:

- “Создание пользовательского расширенного типа индекса” на стр. 160

Ссылки, связанные с данной темой:

- “CREATE INDEX EXTENSION statement” в *SQL Reference, Том 2*

Сведения о поиске в индексе

Поиск в индексе отображает аргументы поиска на диапазоны поиска.

Условие WITH TARGET KEY оператора CREATE INDEX EXTENSION задает параметры ключей назначения, генерируемых пользовательской табличной функцией, заданной в условии GENERATE KEY USING. Для столбца ключа назначения задается и связывается с ним одно имя параметра и тип данных. Этот параметр соответствует столбцам таблицы RETURNS пользовательской табличной функции, заданной в условии GENERATE KEY USING.

Условие SEARCH METHODS определяет для этого индекса один или несколько методов поиска. Каждый метод поиска состоит из имени метода, аргументов поиска, функции генерации диапазона и необязательной функции фильтрации индекса. Каждый метод поиска определяет, как пользовательская табличная функция генерирует диапазоны поиска индекса для базового пользовательского индекса. Кроме того, каждый метод поиска определяет, как пользовательская скалярная функция, возвращающая одно значение, может задавать дополнительные спецификаторы для конкретного диапазона поиска.

- Условие WHEN задает метку для метода поиска. Метка - это идентификатор SQL, связанный с именем метода, заданным в правиле применения индекса (которое определяется в условии PREDICATES пользовательской функции). В качестве аргументов функции диапазонов и/или функции фильтрации индекса задаются одно или несколько имен параметров и типов данных. Условие WHEN задает действие, которое может выполнить оптимизатор в случае, когда условие PREDICATES оператора CREATE FUNCTION совпадает со входным запросом.
- Условие RANGE THROUGH задает пользовательскую внешнюю табличную функцию, создающую диапазоны ключей индекса. Это позволяет оптимизатору избежать вызова связанной пользовательской функции в случаях, когда ключи индекса не попадают в диапазоны ключей.
- Необязательное условие FILTER USING можно использовать для задания пользовательской внешней табличной функции или выражения CASE,

используемых для фильтрации записей индекса, возвращаемых функцией создания диапазонов. Если значение, возвращаемое функцией фильтрации индекса или выражением CASE, равно 1, из таблицы считывается строка, соответствующая этой записи индекса. Если возвращено какое-либо иное значение (не равное 1), эта запись индекса отвергается. Эта возможность полезна, если затраты на второй фильтр малы по сравнению с затратами на выполнение исходного метода, а избирательность второго фильтра относительно невелика.

Понятия, связанные с данным:

- “Создание пользовательского расширенного типа индекса” на стр. 160
- “Сведения о поддержке индекса” на стр. 161
- “Сведения о применении индекса” на стр. 163

Ссылки, связанные с данной темой:

- “CREATE INDEX EXTENSION statement” в *SQL Reference, Том 2*

Сведения о применении индекса

Применение индекса происходит при оценке метода поиска.

Оператор CREATE FUNCTION для внешней скалярной функции создает пользовательский предикат, используемый с методами поиска, определенными для этого расширения индекса.

Условие PREDICATES определяет использующие эту функцию предикаты, для которых возможно применение этого расширения индекса (и для которых можно использовать необязательное условие SELECTIVITY, чтобы задать условие поиска для конкретного предиката). Если задано условие PREDICATES, функция должна быть определена с опциями DETERMINISTIC и NO EXTERNAL ACTION.

- Условие WHEN задает конкретное использование определенной в предикате функции с операцией сравнения (=, >, < и другими) и константой или выражением (с использованием условия EXPRESSION AS). Для предиката, использующего эту функцию с той же операцией сравнения и той же константой или выражением, могут использоваться фильтрация и применение индекса. Константы в основном используются для булевских выражений, результат которых - 1 или 0. Для всех остальных случаев лучше использовать условие EXPRESSION AS.
- Условие FILTER USING задает функцию фильтрации, которая может применяться для дополнительной фильтрации таблицы результатов. Это более быстрый вариант определенной функции (используемой в предикате), поскольку уменьшается число строк, для проверки годности которых нужно выполнить пользовательский предикат. Если результаты, создаваемые по

индексу, близки к ожидаемым результатам пользовательского предиката, программа функции фильтрации может быть ненужной.

- Для каждого метода поиска расширения индекса можно дополнительно определить набор правил применения индекса. Можно также определить в расширении индекса метод поиска, чтобы описать назначения поиска, аргументы поиска и то, как они должны использоваться для поиска в индексе.
  - Условие SEARCH BY INDEX EXTENSION задает расширение индекса.
  - Необязательное условие EXACT указывает, что просмотр индекса дает те же результаты, что и применение предиката. Это условие указывает базе данных, что не нужно применять исходную пользовательскую функцию предиката или функцию фильтрации после просмотра индекса. Если просмотр индекса не используется, должны применяться исходная пользовательская функция предиката и функция фильтрации. Если условие EXACT не задано, после просмотра индекса применяется исходный пользовательский предикат. Условие EXACT полезно, когда просмотр индекса возвращает те же результаты, что и предикат. Он предотвращает выполнение запросом пользовательского предиката для результатов, полученных от просмотра индекса. Если предполагается, что индекс не будет возвращать точно те же результаты, что и предикат, не задавайте условие EXACT.
  - Условие WHEN KEY задает назначение поиска. Для ключа задается только одно назначение поиска. Значение, заданное после условия WHEN KEY, указывает имя параметра определяемой функции. Это условие считается истинным, если значение указанного параметра - столбец, входящий в индекс, основанный на заданном расширении индекса.
  - Условие USE определяет аргумент поиска. Аргумент поиска указывает, какой из методов, определенных в расширении индекса, будет использоваться. Заданное здесь имя метода должно совпадать с именем метода, определенным в расширении индекса. В значениях одного или нескольких параметров указываются имена параметров определяемой функции, которые не должны совпадать с именами параметров, заданных для назначения поиска. Число и типы данных значений параметров должны точно соответствовать параметрам, определенным для этого метода в расширении индекса. Соответствие должно быть точным для встроенных и особых типов, а структурированные типы должны быть совместимыми (быть в одной иерархии типов).

Понятия, связанные с данным:

- “Создание пользовательского расширенного типа индекса” на стр. 160
- “Сведения о поддержке индекса” на стр. 161
- “Сведения о поиске в индексе” на стр. 162
- “Сценарий определения расширения индекса” на стр. 165

Ссылки, связанные с данной темой:

- “CREATE FUNCTION (External Scalar) statement” в *SQL Reference, Том 2*

Сценарий определения расширения индекса

Сценарий определения расширения индекса:

1. Определите структурированные типы (иерархию shape). При помощи оператора CREATE TYPE определим иерархию типов, где shape (форма) - это надтип, а nullshape (пустая форма), point (точка), line (линия) и polygon (многоугольник) - подтипы. Эти структурированные типы - модель пространственных объектов. Например, положение магазина - это точка, русло реки - линия, а границы территории - многоугольник. Атрибут mbr - это минимальный ограничивающий прямоугольник. Атрибут gtype указывает тип объекта - точка, линия или многоугольник. Для моделирования географических границ используются атрибуты numpart, numpoint и geometry. Все остальные атрибуты можно игнорировать, поскольку они не используются в этом сценарии.
2. Создайте расширение индекса.
   - Используйте оператор CREATE FUNCTION, чтобы создать функции, используемые для преобразования ключей (gridentry), создания диапазонов (gridrange) и фильтрации индекса (checkduplicate и mbroverlap).
   - Используйте оператор CREATE INDEX EXTENSION, чтобы создать остальные требуемые компоненты индекса.
3. Создайте преобразование ключей, соответствующее компоненту поддержки индекса.

   ```
   CREATE INDEX EXTENSION имя_расш_инд (имя_парам тип_данных, ...)
      FROM SOURCE KEY (имя_парам тип_данных)
      GENERATE KEY USING вызов_табличной_функции
      ...
   ```

   Условие FROM SOURCE KEY задает параметр и тип данных преобразования ключей. Условие GENERATE KEY USING задает функцию отображения исходного ключа на значение, генерируемое этой функцией.
4. Определите функции создания диапазонов и фильтрации индекса, соответствующие компоненту поиска в индексе.

   ```
   CREATE INDEX EXTENSION имя_расш_инд (имя_парам тип_данных, ...)
      ...
      WITH TARGET KEY
         WHEN имя_метода (имя_парам тип_данных, ...)
         RANGE THROUGH вызов_функции_создания_диапазонов
         FILTER USING вызов_функции_фильтрации_индекса
   ```

   Условие WITH TARGET KEY задает определение метода поиска. Условие WHEN задает имя метода. Условие RANGE THROUGH задает функцию, используемую для ограничения рабочей области индекса. Условие FILTER USING задает функцию, используемую для исключения из полученных значений индекса ненужных элементов.

Примечание: В условии FILTER USING вместо функции фильтрации индекса можно задать выражение CASE.

5. Определите предикаты для применения этого расширения индекса.

```
CREATE FUNCTION within (x shape, y shape)
   RETURNS INTEGER
   ...
   PREDICATES
      WHEN = 1
         FILTER USING mbrWithin (x..mbr..xmin, ...)
         SEARCH BY INDEX EXTENSION grid_extension
         WHEN KEY (имя_парам) USE имя_метода(имя_парам)
```

   В условии PREDICATES определяются один или несколько предикатов, каждый из которых начинается с условия WHEN. Объявление предиката начинается с условия WHEN, после которого задаются операция сравнения и константа или условие EXPRESSION AS. Условие FILTER USING задает функцию фильтрации, которая может использоваться для дополнительной фильтрации таблицы результатов. Это более простая версия определенной функции (используемой в предикате); она уменьшает число строк, для проверки годности которых нужно выполнить пользовательский предикат. Условие SEARCH BY INDEX EXTENSION задает способ применения индекса. Для применения индекса определяется набор правил использования метода поиска расширения индекса, который может использоваться для этого индекса. Условие WHEN KEY задает правило применения. Правило применения описывает назначения поиска и аргументы поиска, а также то, как использовать их для поиска в индексе с помощью этого метода.

6. Определите функцию фильтрации.

```
CREATE FUNCTION mbrWithin (...)
```

   Определенная здесь функция будет использоваться в предикате расширения индекса.

Чтобы оптимизатор запросов мог успешно применять созданные для улучшения производительности запросов индексы, в вызове функции можно задать опцию SELECTIVITY. Если известен примерный процент строк, возвращаемый предикатом, то в вызове функции можно указать опцию SELECTIVITY, позволяющую оптимизатору DB2® выбрать более эффективный путь доступа.

В следующем примере пользовательская функция `within` вычисляет центр и радиус (на основании первого и второго параметров соответственно) и строит строку оператора с соответствующей избирательностью:

```
SELECT * FROM customer
   WHERE within(loc, circle(100, 100, 10)) = 1 SELECTIVITY .05
```

В этом примере указанный предикат отфильтровывает 95 процентов строк (`SELECTIVITY .05`) таблицы `customer`.

Понятия, связанные с данным:

- “Создание пользовательского расширенного типа индекса” на стр. 160
- “Сведения о поддержке индекса” на стр. 161
- “Сведения о поиске в индексе” на стр. 162
- “Сведения о применении индекса” на стр. 163

Ссылки, связанные с данной темой:

- “CREATE INDEX EXTENSION statement” в *SQL Reference, Том 2*
- “CREATE FUNCTION (External Scalar) statement” в *SQL Reference, Том 2*

Вызов Мастера настройки производительности из командной строки

Процедура:

После создания базы данных можно вызвать Мастера настройки производительности с помощью команды AUTOCONFIGURE. Это можно сделать и в том случае, если опция AUTOCONFIGURE была задана при создании базы данных.

С помощью опций команды AUTOCONFIGURE можно определить значения некоторых параметров конфигурации и задать область действия этих параметров. В качестве области действия можно указать значение NONE, указывающее, что ни одно из указанных значений применяться не будет; DB ONLY, указывающее, что будут применяться только параметры конфигурации базы данных и пула буферов; или DB AND DBM, указывающее, что будут применяться значения всех параметров.

Глава 3. Изменение базы данных

Эта глава посвящена вопросам, которые надо учитывать перед изменением базы данных, а также изменению или отбрасыванию объектов базы данных.

Перед изменением базы данных

Через некоторое время после разработки структуры базы данных может потребоваться ее изменение. Нужно заново рассмотреть основные аспекты структуры базы данных. Особое внимание нужно уделить следующим вопросам:

- “Изменение свойств логической и физической структуры”
- “Изменение информации о лицензиях”
- “Изменение экземпляров (в UNIX)” на стр. 170
- “Изменение файла конфигурации узла” на стр. 175
- “Изменение файла конфигурации базы данных” на стр. 175

Изменение свойств логической и физической структуры

Перед тем как вносить изменения, влияющие на всю базу данных, следует заново просмотреть логическую и физическую структуру базы данных. Например, при изменении табличного пространства следует проверить выбор используемого типа хранения - SMS или DMS.

Понятия, связанные с данным:

- “Проектирование групп разделов базы данных” в *Руководство администратора: Планирование*
- “Размер объектов базы данных” в *Руководство администратора: Планирование*
- “Проектирование табличного пространства” в *Руководство администратора: Планирование*
- “Рекомендации по выбору табличных пространств для таблиц” в *Руководство администратора: Планирование*
- “Дополнительные особенности структуры базы данных” в *Руководство администратора: Планирование*
- “Рекомендации по созданию таблиц с многомерной кластеризацией” в *Руководство администратора: Планирование*

Изменение информации о лицензиях

В какой-то момент может потребоваться увеличить число лицензий на продукты DB2®. Чтобы проверить использование установленных продуктов и

соответственно увеличить число лицензий, можно использовать Центр лицензий, входящий в Центр управления.

Понятия, связанные с данным:

- “Управление лицензиями” на стр. 31

Изменение экземпляров (в UNIX)

Экземпляры создаются таким образом, чтобы они были как можно более независимы от последующих установок и удалений продуктов.

В большинстве случаев существующие экземпляры будут автоматически получать или терять доступ к функциям устанавливаемых или удаляемых продуктов. Однако при установке или удалении некоторых выполняемых программ или компонентов существующие экземпляры не наследуют автоматически новые параметры конфигурации системы или не получают доступ ко всем дополнительным функциям. В таком случае экземпляр необходимо обновить.

Если продукт DB2® был обновлен путем установки временного исправления программы (PTF) или вставки, то нужно обновить все существующие экземпляры DB2 с помощью команды **db2iupdt**.

Прежде чем пытаться изменить или удалить экземпляр, нужно узнать информацию об экземплярах и серверах разделов баз данных в экземплярах.

Понятия, связанные с данным:

- “Создание экземпляра” на стр. 19

Задачи, связанные с данной темой:

- “Обновление конфигурации экземпляра в UNIX” на стр. 171
- “Удаление экземпляров” на стр. 174

Ссылки, связанные с данной темой:

- “db2iupdt - Update Instances Command” в *Command Reference*

Сведения об изменении экземпляров

Перед изменением экземпляра следует создать список всех существующих экземпляров.

Список экземпляров

Процедура:

Чтобы получить список всех доступных в системе экземпляров с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Экземпляры**.
2. Щелкните правой кнопкой мыши по папке Экземпляры и выберите из всплывающего меню пункт **Добавить**.
3. В окне Добавить экземпляр нажмите кнопку **Обновить**.
4. Щелкните по стрелке вниз, чтобы увидеть список экземпляров баз данных.
5. Нажмите кнопку **Отмена**, чтобы закрыть это окно.

Чтобы получить список всех доступных в системе экземпляров из командной строки, введите команду:

```
db2ilist
```

Чтобы узнать, какой экземпляр используется в текущем сеансе (на поддерживаемых платформах Windows), вызовите команду:

```
set db2instance
```

Обновление конфигурации экземпляра в UNIX

При выполнении команды **db2iupdt** указанный экземпляр обновляется следующим образом:

- Заменяются файлы в подкаталоге `sqllib` начального каталога владельца экземпляра.
- Если был изменен тип узла, создается новый файл конфигурации менеджера баз данных. Для этого объединяются соответствующие значения из существующего файла конфигурации менеджера базы данных и из файла конфигурации менеджера базы данных по умолчанию для нового типа узла. Если создается новый файл конфигурации менеджера базы данных, старый файл конфигурации копируется в подкаталог `backup` подкаталога `sqllib` начального каталога владельца экземпляра.

Процедура:

В AIX команда **db2iupdt** расположена в каталоге `/usr/opt/db2_08_01/instance/`. В HP-UX, Solaris и Linux команда **db2iupdt** расположена в каталоге `/opt/IBM/db2/V8.1/instance/`.

Ниже приведен формат команды:

```
db2iupdt имя_экземпляра
```

`имя_экземпляра` - имя регистрации владельца экземпляра.

Для этой команды можно использовать другие необязательные параметры:

- –h или –?

  Выводит для этой команды меню справки.

- –d

  Задает режим отладки, используемый для обнаружения причин ошибок.

- –a тип_аутентификации

  Задает тип аутентификации для экземпляра. Допустимы значения SERVER, SERVER_ENCRYPT и CLIENT. Если этот параметр не задан, при установке сервера DB2 по умолчанию задается тип аутентификации SERVER. В противном случае задается тип аутентификации CLIENT. Этот тип аутентификации экземпляра применяется для всех баз данных этого экземпляра.

- –e

  Позволяет обновить все существующие экземпляры. Их можно увидеть с помощью команды **db2ilist**.

- –u ID для изолированных функций

  Задает пользователя, под именем которого будут выполняться изолированные пользовательские функции и хранимые процедуры. Это необязательный параметр при установке клиента DB2 или клиента разработки программ DB2. Для других продуктов DB2 это обязательный параметр.

  Примечание: В качестве ID для изолированных функций нельзя указывать значения “root” и “bin”.

- –k

  Это параметр сохраняет текущий тип экземпляра. Если этот параметр не задан, тип текущего экземпляра обновляется до самого высокого доступного типа в следующем порядке:

  – Сервер многораздельных баз данных с локальными и удаленными клиентами (тип экземпляра по умолчанию для DB2 Enterprise - Extended Edition)
  – Сервер баз данных с локальными и удаленными клиентами (тип экземпляра по умолчанию для DB2 Universal Database Enterprise Server Edition)
  – Клиент (тип экземпляра по умолчанию для клиента DB2)

Примеры:

- Если после создания экземпляра был установлен продукт DB2 Universal Database Workgroup Server Edition или DB2 Universal Database Enterprise Server Edition, вызовите следующую команду для обновления экземпляра:

  ```
  db2iupdt -u db2fenc1 db2inst1
  ```

- Если после создания экземпляра установлена система DB2 Connect Enterprise Edition, в качестве FencedID можно использовать имя экземпляра:

  ```
  db2iupdt -u db2inst1 db2inst1
  ```

- Для обновления экземпляров клиентов можно вызвать следующую команду:

```
db2iupdt db2inst1
```

Задачи, связанные с данной темой:

- “Удаление экземпляров” на стр. 174

Ссылки, связанные с данной темой:

- “db2ilist - List Instances Command” в *Command Reference*
- “db2iupdt - Update Instances Command” в *Command Reference*

Обновление конфигурации экземпляра в Windows

При выполнении команды **db2iupdt** указанный экземпляр обновляется так:

- Заменяются файлы в подкаталоге `sqllib` начального каталога владельца экземпляра.
- Если был изменен тип узла, создается новый файл конфигурации менеджера баз данных. Для этого объединяются соответствующие значения из существующего файла конфигурации менеджера базы данных и из файла конфигурации менеджера базы данных по умолчанию для нового типа узла. Если создается новый файл конфигурации менеджера базы данных, старый файл конфигурации копируется в подкаталог `backup` подкаталога `sqllib` начального каталога владельца экземпляра.

Процедура:

Команда **db2iupdt** находится в каталоге `\sqllib\bin`.

Эта команда используется следующим образом:

```
db2iupdt имя_экземпляра
```

`имя_экземпляра` - имя регистрации владельца экземпляра.

Для этой команды можно использовать другие необязательные параметры:

- /h: хост

  Переопределяет имя хоста TCP/IP по умолчанию, если с текущей системой связано несколько имен хоста TCP/IP.

- /p: путь к профилю экземпляра

  Задает новый путь профиля экземпляра для обновляемого экземпляра.

- /r: начальный-порт,конечный-порт

  Задает диапазон портов TCP/IP, применяемый экземпляром многораздельной базы данных при работе с несколькими разделами.

- /u: имя-пользователя,пароль

  Задает имя учетной записи и пароль для службы DB2.

Удаление экземпляров

Процедура:

Чтобы удалить экземпляр с помощью Центра управления:

1. Раскройте дерево объектов и найдите экземпляр, который нужно удалить.
2. Щелкните правой кнопкой мыши по имени этого экземпляра и выберите из всплывающего меню пункт **Удалить**.
3. Включите переключатель **Подтверждение** и нажмите кнопку **OK**.

Чтобы удалить экземпляр из командной строки, введите команду:

```
db2idrop <имя_экземпляра>
```

Подготовка и подробности удаления экземпляра из командной строки:

1. Остановите все прикладные программы, использующие в настоящее время этот экземпляр.
2. Остановите процессор командной строки, выполнив команду **db2 terminate** в каждом командном окне DB2.
3. Остановите экземпляр, выполнив команду **db2stop**.
4. Сделайте резервную копию каталога экземпляра, который указан в переменной реестра DB2INSTPROF.

   В операционных системах UNIX можно сделать резервные копии файлов, расположенных в каталоге `INSTHOME/sqllib` (где `INSTHOME` - начальный каталог владельца экземпляра). Например, можно сохранить файл конфигурации менеджера баз данных `db2systm`, файл `db2nodes.cfg` и программы пользовательских функций или изолированных пользовательских процедур.
5. Только в операционных системах UNIX: Завершите работу в качестве владельца экземпляра.
6. Только в операционных системах UNIX: Зарегистрируйтесь в системе как пользователь с полномочиями root.
7. Введите команду **db2idrop**:

   ```
   db2idrop имя_экземпляра
   ```

   где `имя_экземпляра` - имя экземпляра, который нужно удалить.

   Эта команда удаляет запись об этом экземпляре из списка экземпляров и каталог этого экземпляра.
8. Только в операционных системах UNIX, необязательно: В качестве пользователя с полномочиями root удалите ID пользователя и группу

владельца экземпляра (если они используются только для этого экземпляра). Не удаляйте ID пользователя и группу, если собираетесь заново создать этот экземпляр.

Этот шаг не является обязательным, поскольку владелец экземпляра и группа владельца экземпляра могут использоваться для других целей.

Команда **db2idrop** удаляет запись об экземпляре из списка экземпляров и подкаталог `sqllib` начального каталога владельца экземпляра.

Примечание: В операционной системе UNIX при попытке отбросить экземпляр с помощью команды db2idrop выдается сообщение с информацией о том, что невозможно удалить подкаталог `sqllib`, а в подкаталоге `adm` создается несколько файлов с расширением .nfs. Подкаталог `adm` представляет собой смонтированную файловую систему NFS, и за управление файлами отвечает сервер. Удалите файлы *.nfs из того расположения на файловом сервере, в котором смонтирован каталог. После этого удалите подкаталог `sqllib`.

Ссылки, связанные с данной темой:

- “db2stop - Stop DB2 Command” в *Command Reference*
- “TERMINATE Command” в *Command Reference*
- “STOP DATABASE MANAGER Command” в *Command Reference*
- “db2idrop - Remove Instance Command” в *Command Reference*
- “db2ilist - List Instances Command” в *Command Reference*

Изменение файла конфигурации узла

Если вы планируете изменить группу разделов базы данных (например, добавить, удалить или переместить разделы), то необходимо обновить файл конфигурации узла.

Понятия, связанные с данным:

- “Управление ресурсами сервера баз данных” в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- “ADD DBPARTITIONNUM Command” в *Command Reference*
- “DROP DBPARTITIONNUM VERIFY Command” в *Command Reference*

Изменение файла конфигурации базы данных

Процедура:

Если вы планируете изменить базу данных, нужно проверить значения параметров конфигурации. Некоторые из этих значений могут время от времени изменяться в процессе использования базы данных.

Для изменения конфигурации базы данных воспользуйтесь мастером настройки производительности, предусмотренным в Центре управления, или командой db2 autoconfigure с соответствующими опциями. Этот мастер помогает настроить производительность и выровнять требования к памяти при использовании в экземпляре одной базы данных, советуя, какие параметры конфигурации нужно изменить, и предлагая для них рекомендуемые значения.

Примечание: Если изменены какие-либо параметры, их новые значения вступят в силу:

- Для параметров базы данных: при первом новом соединении с базой данных после завершения соединений всех прикладных программ.
- Для параметров менеджера баз данных: после остановки и запуска экземпляра.

В большинстве случаев значения, рекомендуемые мастером по настройке производительности, обеспечивают лучшую производительность, чем значения по умолчанию, так как эти рекомендации основываются на информации о рабочей нагрузке и конкретном сервере. Однако учтите, что эти рекомендуемые значения служат для улучшения производительности системы базы данных, но не обязательно обеспечивают наилучшую производительность. Их нужно рассматривать как начальную точку для дальнейшей настройки в целях получения оптимальной производительности.

Чтобы изменить конфигурацию базы данных с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Базы данных**.
2. Щелкните правой кнопкой мыши по экземпляру или базе данных, которые нужно изменить, и выберите из всплывающего меню пункт **Настроить производительность при помощи мастера**.
3. На каждой странице измените информацию требуемым образом.
4. Перейдите на страницу **Результаты**, чтобы проверить свою работу и применить какие-либо предложенные параметры конфигурации.
5. Внеся все нужные изменения, нажмите кнопку **Завершить**.

Для запуска мастера настройки производительности из командной строки введите команду AUTOCONFIGURE.

Для изменения отдельных параметров в конфигурации менеджера баз данных из командной строки введите:

```
UPDATE DBM CFG FOR <алиас_базы_данных>
   USING <ключевое_слово_конфигурации>=<значение>
```

В одной команде можно задать одну или несколько комбинаций <ключевое_слово_конфигурации>=<значение>. Большинство изменений в файле конфигурации менеджера баз данных вступают в силу только после того, как они будут загружены в память. Для параметров конфигурации сервера это происходит при выполнении команды START DATABASE MANAGER. Для параметров конфигурации клиента это происходит при перезапуске прикладной программы.

Чтобы просмотреть или напечатать текущие параметры конфигурации менеджера баз данных, используйте команду GET DATABASE MANAGER CONFIGURATION.

Понятия, связанные с данным:

- “Измерение производительности” в *Руководство администратора: Производительность*

Задачи, связанные с данной темой:

- “Изменение конфигурации базы данных в нескольких разделах” на стр. 177
- “Настройка DB2 с помощью параметров конфигурации” в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- “GET DATABASE MANAGER CONFIGURATION Command” в *Command Reference*
- “UPDATE DATABASE MANAGER CONFIGURATION Command” в *Command Reference*

Изменение конфигурации базы данных в нескольких разделах

Процедура:

Если база данных распределена по нескольким разделам, файлы конфигурации базы данных должны быть одинаковыми во всех разделах. Это требуется, потому что компилятор SQL компилирует распределенные операторы SQL, используя информацию из файла конфигурации узлов, и создает план доступа в соответствии с потребностями этого оператора SQL. Если на разделах базы данных файлы конфигурации отличаются, это может привести к тому, что в зависимости от того, на каком разделе базы данных выполняется подготовка оператора, будут получаться разные планы доступа. Для того чтобы изменить файл конфигурации во всех разделах базы данных, воспользуйтесь командой **db2_all**.

Понятия, связанные с данным:

- “Вызов команд в среде многораздельной базы данных” на стр. 371

Задачи, связанные с данной темой:

- “Изменение файла конфигурации базы данных” на стр. 175

Изменение базы данных

Для изменения баз данных надо выполнить почти столько же задач, как и для создания баз данных. В этих задачах изменяются или удаляются объекты созданной ранее базы данных. Это следующие задачи:

- “Отбрасывание базы данных”
- “Изменение группы разделов базы данных” на стр. 179
- “Изменение табличного пространства” на стр. 180
- “Отбрасывание схемы” на стр. 190
- “Изменение структуры и содержимого таблицы” на стр. 191
- “Изменение пользовательского структурированного типа” на стр. 214
- “Удаление и обновление строк в типизированной таблице” на стр. 214
- “Изменение имени таблицы или индекса” на стр. 214
- “Отбрасывание таблицы” на стр. 216
- “Отбрасывание пользовательской временной таблицы” на стр. 218
- “Отбрасывание триггера” на стр. 218
- “Отбрасывание пользовательской функции (UDF), отображения функции или метода” на стр. 219
- “Отбрасывание пользовательского типа (UDT) или отображения типа” на стр. 220
- “Изменение и отбрасывание производной таблицы” на стр. 221
- “Восстановление неработоспособных производных таблиц” на стр. 223
- “Отбрасывание материализованной таблицы запроса и промежуточной таблицы” на стр. 224
- “Восстановление неработоспособных сводных таблиц” на стр. 225
- “Отбрасывание индекса, расширения индекса и спецификации индекса” на стр. 226
- “Зависимости операторов при изменении объектов” на стр. 227

Отбрасывание базы данных

Процедура:

Хотя некоторые объекты в базе данных могут быть изменены, сама база данных не может быть изменена: ее нужно отбросить и создать заново. Отбрасывание базы данных может иметь далеко идущие последствия, поскольку при этом удаляются все ее объекты, контейнеры и связанные с ними файлы. Отброшенная база данных удаляется из каталогов баз данных.

Чтобы отбросить базу данных с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Базы данных**.
2. Щелкните правой кнопкой мыши по базе данных, которую нужно отбросить, и выберите из всплывающего меню пункт **Отбросить**.
3. Включите переключатель **Подтверждение** и нажмите кнопку **OK**.

Чтобы отбросить базу данных из командной строки, введите команду:

```
DROP DATABASE <имя>
```

Следующая команда удаляет базу данных SAMPLE:

```
DROP DATABASE SAMPLE
```

Примечание: Если вы собираетесь продолжить эксперименты с базой данных SAMPLE, не отбрасывайте ее. Если вы отбросили базу данных SAMPLE, а затем обнаружили, что она снова вам нужна, ее можно создать заново.

Ссылки, связанные с данной темой:

- “GET SNAPSHOT Command” в *Command Reference*
- “DROP DATABASE Command” в *Command Reference*
- “LIST ACTIVE DATABASES Command” в *Command Reference*

Изменение группы разделов базы данных

Процедура:

После добавления или отбрасывания разделов необходимо перераспределить данные по новому набору разделов базы данных из группы разделов. Для этого вызовите команду REDISTRIBUTE DATABASE PARTITION GROUP.

Понятия, связанные с данным:

- “Перераспределение данных” в *Руководство администратора: Производительность*
- “Управление ресурсами сервера баз данных” в *Руководство администратора: Производительность*

Задачи, связанные с данной темой:

- “Перераспределение данных между разделами” в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- “REDISTRIBUTE DATABASE PARTITION GROUP Command” в *Command Reference*

Изменение табличного пространства

Процедура:

При создании базы данных создаются по крайней мере три табличных пространства: одно табличное пространство каталогов (SYSCATSPACE), одно пользовательское табличное пространство (с именем по умолчанию USERSPACE1) и одно системное временное табличное пространство (с именем по умолчанию TEMPSPACE1). Необходимо иметь по крайней мере одно табличное пространство каждого из этих типов. Можно также добавить дополнительные пользовательские и временные табличные пространства.

Примечание: Табличное пространство каталога SYSCATSPACE нельзя отбросить или создать заново. Всегда должно быть по крайней мере одно системное временное табличное пространство с размером страницы, равным 4 Кб. Вы можете создать другие системные временные табличные пространства. Также нельзя изменить размер страниц или размер экстента табличного пространства после его создания.

Задачи, связанные с данной темой:

- “Добавление контейнера в табличное пространство DMS” на стр. 180
- “Изменение контейнеров в табличном пространстве DMS” на стр. 182
- “Добавление контейнера в табличное пространство SMS раздела” на стр. 185
- “Переименовать табличное пространство” на стр. 186
- “Отбрасывание пользовательского табличного пространства” на стр. 187
- “Отбрасывание системного временного табличного пространства” на стр. 188
- “Отбрасывание пользовательского временного табличного пространства” на стр. 190

Ссылки, связанные с данной темой:

- “ALTER TABLESPACE statement” в *SQL Reference, Том 2*

Сведения об изменении табличных пространств

В этом разделе описаны действия, связанные с изменением табличных пространств.

Добавление контейнера в табличное пространство DMS

Процедура:

Можно увеличить размер табличного пространства DMS (то есть табличного пространства, при создании которого задано условие MANAGED BY DATABASE), добавив к нему один или несколько контейнеров.

В результате добавления контейнеров в табличное пространство или расширения существующих контейнеров может быть выполнено перераспределение данных. Процесс перераспределения заключается в перемещении экстентов табличного пространства из одного расположения в другое. В ходе этого процесса делается попытка сохранить равномерное распределение данных в табличном пространстве. Перераспределение данных не обязательно затрагивает все контейнеры. Это зависит от многих факторов, в частности, от конфигурации существующих контейнеров, размера новых контейнеров и того, насколько заполнено табличное пространство.

При добавлении контейнеров в существующее табличное пространство они не обязательно размещаются, начиная с блока 0. Начало размещения контейнеров в карте определяется менеджером баз данных в зависимости от размера добавляемых контейнеров. Если контейнер не очень велик, он располагается таким образом, чтобы он заканчивался в последнем блоке карты. Большие контейнеры размещаются, начиная с блока 0.

Перераспределение данных не выполняется, если при добавлении контейнеров создается новый набор блоков. Новый набор блоков создается в том случае, если в операторе ALTER TABLESPACE задано условие BEGIN NEW STRIPE SET. Для того чтобы добавить контейнеры в существующие наборы блоков, укажите в операторе ALTER TABLESPACE условие ADD TO STRIPE SET.

В процессе перераспределения доступ к табличному пространству не ограничивается. Если нужно добавить несколько контейнеров, их следует добавлять одновременно.

Чтобы добавить контейнер к табличному пространству DMS с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Табличные пространства**.
2. Щелкните правой кнопкой по табличному пространству, в которое нужно добавить контейнер, и выберите из всплывающего меню пункт **Изменить**.
3. Нажмите кнопку **Добавить**, введите необходимую информацию и нажмите кнопку **OK**.

Чтобы добавить контейнер к табличному пространству DMS из командной строки, введите команду:

```
ALTER TABLESPACE <имя>
   ADD (DEVICE '<путь>' <размер>, FILE '<имя-файла>' <размер>)
```

В следующем примере показано, как в системе UNIX добавить в табличное пространство два новых контейнера типа устройство (по 10000 страниц каждый):

```
ALTER TABLESPACE RESOURCE
  ADD (DEVICE '/dev/rhd9'  10000,
       DEVICE  '/dev/rhd10' 10000)
```

Учтите, что оператор ALTER TABLESPACE позволяет изменить другие свойства табличного пространства, которые могут влиять на производительность.

Понятия, связанные с данным:

- “Влияние табличного пространства на оптимизацию запроса” в *Руководство администратора: Производительность*
- “Добавление контейнеров в табличное пространство DMS и их расширение” в *Руководство администратора: Планирование*

Задачи, связанные с данной темой:

- “Добавление контейнера в табличное пространство SMS раздела” на стр. 185

Ссылки, связанные с данной темой:

- “ALTER TABLESPACE statement” в *SQL Reference, Том 2*

Изменение контейнеров в табличном пространстве DMS

Ограничения:

Каждое устройство с линейным доступом может применяться в качестве одного контейнера. Размер устройства с линейным доступом фиксирован. Его нельзя изменить после создания устройства. Перед изменением размера или расширением контейнера, созданного на основе устройства с линейным доступом, необходимо убедиться, что новый размер контейнера не превосходит размер устройства с линейным доступом.

Процедура:

Вы можете изменить размер контейнеров, расположенных в табличном пространстве DMS (то есть табличном пространстве, при создании которого было задано условие MANAGED BY DATABASE).

Для того чтобы увеличить размер одного или нескольких контейнеров в табличном пространстве DMS с помощью Центра управления, выполните следующие действия:

1. Раскройте дерево объектов и найдите папку **Табличные пространства**.
2. Щелкните правой кнопкой по табличному пространству, в которое нужно добавить контейнер, и выберите из всплывающего меню пункт **Изменить**.
3. Выберите **Изменить размер**, задайте необходимую информацию и нажмите кнопку **Ok**.

Кроме того, вы можете отбросить контейнеры, расположенные в табличном пространстве DMS, уменьшить их размер или добавить новые контейнеры, не перераспределяя данные по всем контейнерам.

Отбросить существующие контейнеры табличного пространства или уменьшить их размер можно только в том случае, если число отбрасываемых экстентов или экстентов, удаляемых в результате уменьшения размера, не превосходит числа свободных экстентов, расположенных выше верхней границы. Верхняя граница задает максимальный номер выделенной страницы в табличном пространстве. Верхняя граница не всегда совпадает с числом занятых страниц в табличном пространстве, так как некоторые экстенты ниже этой границы могли быть освобождены для повторного использования.

Число свободных экстентов, расположенных выше верхней границы, важно знать потому, что все остальные экстенты должны располагаться в одном логическом разделе внутри табличного пространства. В итоговом табличном пространстве должно быть достаточно места для хранения всех данных. Если свободного места недостаточно, будет отправлено сообщение об ошибке (SQL20170N, SQLSTATE 57059).

Для отбрасывания контейнеров применяется опция DROP оператора ALTER TABLESPACE. Например:

```
ALTER TABLESPACE TS1 DROP (FILE 'file1', DEVICE '/dev/rdisk1')
```

Для уменьшения размера существующих контейнеров можно воспользоваться опцией RESIZE или REDUCE. Опция RESIZE позволяет увеличить или уменьшить размер контейнеров, указанных в операторе. С ее помощью нельзя увеличить размер одних контейнеров и уменьшить размер других. Опцию RESIZE следует использовать в том случае, если известен новый нижний предел размера контейнера. Опцию REDUCE следует использовать в том случае, если вы не знаете текущий размер контейнера.

Для того чтобы уменьшить размер одного или нескольких контейнеров в табличном пространстве DMS с помощью командной строки, введите:

```
ALTER TABLESPACE <имя>
   REDUCE (FILE '<имя-файла>' <размер>)
```

В следующем примере показано, каким образом в системе Windows можно уменьшить размер файлового контейнера (который уже содержит 1 000 страниц):

```
ALTER TABLESPACE PAYROLL
  REDUCE (FILE 'd:\hldr\finance'  200)
```

После выполнения этой операции размер файла сократится с 1 000 до 800 страниц.

Для того чтобы увеличить размер одного или нескольких контейнеров в табличном пространстве DMS с помощью командной строки, введите:

```
ALTER TABLESPACE <имя>
   RESIZE (DEVICE '<путь>' <размер>)
```

В следующем примере показано, как в системе UNIX увеличить в табличном пространстве два контейнера типа устройство (каждый из которых уже содержит 1000 страниц):

```
ALTER TABLESPACE HISTORY
  RESIZE (DEVICE '/dev/rhd7'  2000,
          DEVICE  '/dev/rhd8' 2000)
```

После этой операции размер этих двух устройств будет увеличен с 1000 до 2000 страниц. Содержимое табличного пространства можно перераспределить между контейнерами. В процессе этого перераспределения доступ к табличному пространству не ограничивается.

Чтобы расширить один или несколько контейнеров в табличном пространстве DMS из командной строки, введите команду:

```
ALTER TABLESPACE <имя>
   EXTEND (FILE '<имя-файла>' <размер>)
```

В следующем примере показано, каким образом в системе Windows можно увеличить размер файловых контейнеров (которые уже содержат по 1 000 страниц) в табличном пространстве:

```
ALTER TABLESPACE PERSNEL
  EXTEND (FILE 'e:\wrkhist1'  200
          FILE  'f:\wrkhist2'  200)
```

После выполнения этой операции размер указанных файлов увеличится с 1 000 до 1 200 страниц. Содержимое табличного пространства можно перераспределить между контейнерами. В процессе этого перераспределения доступ к табличному пространству не ограничивается.

Контейнеры DMS (как контейнеры файлов, так и контейнеры непосредственных устройств), добавленные во время или после создания табличного пространства или расширенные после создания табличного пространства, обрабатываются

параллельно через программы предварительного чтения. Чтобы достичь большего параллелизма операций по созданию контейнеров или изменению их размеров, можно увеличить число выполняемых в системе программ предварительного чтения. Единственный процесс, который не выполняется в параллельном режиме - это регистрация этих действий и (в случае создания контейнеров) маркировка контейнеров.

Примечание: Для максимального увеличения параллелизма операторов CREATE TABLESPACE или ALTER TABLESPACE (при добавлении новых контейнеров к существующему табличному пространству) убедитесь, что число программ предварительного чтения не меньше числа добавляемых контейнеров. Число программ предварительного чтения задается в параметре конфигурации базы данных *num_ioservers*. Для того чтобы изменение значения этого параметра вступило в силу, необходимо завершить работу базы данных. Другими словами, все прикладные программы и пользователи должны отключиться от базы данных.

Учтите, что оператор ALTER TABLESPACE позволяет изменить другие свойства табличного пространства, которые могут влиять на производительность.

Ссылки, связанные с данной темой:

- “ALTER TABLESPACE statement” в *SQL Reference, Том 2*

Добавление контейнера в табличное пространство SMS раздела

Ограничения:

Контейнеры можно добавить только в то табличное пространство SMS, расположенное в разделе, в котором еще нет контейнеров.

Процедура:

Чтобы добавить контейнер к табличному пространству SMS с помощью командной строки, введите команду:

```
ALTER TABLESPACE <имя>
   ADD ('<путь>')
   ON NODE (<номер_раздела>)
```

Раздел с указанным номером, либо все разделы (узлы) с номерами из указанного диапазона, должны существовать в группе разделов базы данных, к которой относится табличное пространство. Номер_раздела можно указать явно или внутри диапазона только в одном условии *ON NODE* оператора.

В следующем примере показано, как добавить новый контейнер на третий раздел группы узлов, используемой табличным пространством “plans” в операционной системе на основе UNIX:

```
ALTER TABLESPACE plans
   ADD ('/dev/rhdisk0')
   ON NODE (3)
```

Задачи, связанные с данной темой:

- “Добавление контейнера в табличное пространство DMS” на стр. 180
- “Изменение контейнеров в табличном пространстве DMS” на стр. 182

Ссылки, связанные с данной темой:

- “ALTER TABLESPACE statement” в *SQL Reference, Том 2*

Переименовать табличное пространство

Ограничения:

Нельзя переименовать табличное пространство SYSCATSPACE.

Нельзя переименовать табличное пространство, находящееся в состоянии “отложенного повтора” или “выполняемого повтора”.

При восстановлении табличного пространства, которое было переименовано после создания резервной копии, нужно использовать в команде RESTORE DATABASE новое имя табличного пространства. Если задано старое имя табличного пространства, оно не будет найдено. Аналогично, при выполнении повтора табличного пространства с помощью команды ROLLFORWARD DATABASE нужно использовать новое имя. Если используется старое имя табличного пространства, оно не будет найдено.

Процедура:

Существующему табличному пространству можно дать новое имя, не затрагивая отдельные объекты в этом табличном пространстве. При переименовании табличного пространства изменяются все записи каталога, относящиеся к этому табличному пространству.

Ссылки, связанные с данной темой:

- “RENAME TABLESPACE statement” в *SQL Reference, Том 2*

Изменение состояния табличного пространства

Процедура:

Условие SWITCH ONLINE оператора ALTER TABLESPACE можно использовать для вывода табличного пространства из состояния OFFLINE, если контейнеры, связанные с этим табличным пространством, вновь стали доступными. Табличное пространство выводится из состояния OFFLINE, когда остальная база данных остается в рабочем состоянии и продолжает использоваться.

Вместо того, чтобы использовать это условие, можно отсоединить от базы данных все прикладные программы и затем вновь установить соединения прикладных программ с базой данных. Это выводит табличное пространство из состояния OFFLINE.

Чтобы вывести табличное пространство из состояния OFFLINE при помощи командной строки, введите:

```
db2 ALTER TABLESPACE <имя>
   SWITCH ONLINE
```

Ссылки, связанные с данной темой:

- "ALTER TABLESPACE statement" в *SQL Reference, Том 2*

Отбрасывание пользовательского табличного пространства

Процедура:

При отбрасывании пользовательского табличного пространства удаляются все данные в этом табличном пространстве, освобождаются контейнеры, удаляются записи каталогов и все объекты, определенные в этом табличном пространстве, отбрасываются или отмечаются как недействительные.

Можно повторно использовать контейнеры пустого табличного пространства, отбросив это табличное пространство, но прежде чем пытаться повторно использовать эти контейнеры, необходимо выполнить принятие (COMMIT) команды DROP TABLESPACE.

Можно отбросить пользовательское табличное пространство, содержащее в себе все данные таблицы, включая индексы и большие объекты. Можно также отбросить пользовательское табличное пространство, таблицы которого могут располагаться в нескольких табличных пространствах. То есть данные таблицы могут храниться в одном табличном пространстве, индексы - в другом, а большие объекты - в третьем. Все эти три табличных пространства должны отбрасываться одновременно с помощью одного оператора. Все табличные пространства, содержащие таблицу, должны задаваться в одном операторе, иначе запрос отбрасывания не будет выполнен.

Чтобы отбросить пользовательское табличное пространство с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Табличные пространства**.
2. Щелкните правой кнопкой мыши по табличному пространству, которое нужно отбросить, и выберите из всплывающего меню пункт **Отбросить**.
3. Включите переключатель **Подтверждение** и нажмите кнопку **OK**.

Чтобы отбросить пользовательское табличное пространство из командной строки, введите команду:

```
DROP TABLESPACE <имя>
```

Следующий оператор SQL отбрасывает табличное пространство ACCOUNTING:

```
DROP TABLESPACE ACCOUNTING
```

Задачи, связанные с данной темой:

- “Отбрасывание системного временного табличного пространства” на стр. 188
- “Отбрасывание пользовательского временного табличного пространства” на стр. 190

Ссылки, связанные с данной темой:

- “COMMIT statement” в *SQL Reference, Том 2*
- “DROP statement” в *SQL Reference, Том 2*

Отбрасывание системного временного табличного пространства

Ограничения:

Если в системном временном табличном пространстве размер страницы равен 4 Кб, то перед его удалением необходимо создать другое системное временное табличное пространство. В новом табличном пространстве размер страницы должен составлять 4 Кб, так как в базе данных должно быть по крайней мере одно системное временное табличное пространство с размером страницы 4 Кб. Например, если есть только одно системное временное табличное пространство с размером страницы 4 Кб, оно относится к типу SMS, и в него нужно добавить контейнер, то вначале необходимо создать новое системное временное табличное пространство с размером страницы 4 Кб, содержащее необходимое число контейнеров, а затем отбросить старое временное табличное пространство. (Для добавления контейнера в табличное пространство DMS не нужно отбрасывать и заново создавать табличное пространство.)

Процедура:

Размер страницы табличного пространства по умолчанию равен 4 Кб.

Чтобы отбросить системное табличное пространство с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Табличные пространства**.
2. Если есть только одно системное временное табличное пространство, щелкните правой кнопкой мыши на папке **Табличные пространства** и выберите в выпадающем меню пункт **Создать —> Табличное пространство с помощью мастера**. В противном случае перейдите к четвертому шагу.
3. Выполните шаги в мастере, чтобы создать необходимое новое системное временное табличное пространство.
4. Еще раз щелкните по папке **Табличные пространства**, чтобы увидеть список существующих табличных пространств в правой части окна (в панели содержимого).
5. Щелкните правой кнопкой мыши по системному временному табличному пространству, которое нужно отбросить, и выберите из всплывающего меню пункт **Отбросить**.
6. Включите переключатель **Подтверждение** и нажмите кнопку **OK**.

Для создания системного временного табличного пространства служит следующий оператор:

```
CREATE SYSTEM TEMPORARY TABLESPACE <имя>
   MANAGED BY SYSTEM USING ('<каталоги>')
```

Затем отбросьте системное табличное пространство из командной строки, введя команду:

```
DROP TABLESPACE <имя>
```

Следующий оператор SQL создает новое системное временное табличное пространство с именем TEMPSPACE2:

```
CREATE SYSTEM TEMPORARY TABLESPACE TEMPSPACE2
   MANAGED BY SYSTEM USING ('d:\systemp2')
```

Создав TEMPSPACE2, можно отбросить исходное системное временное табличное пространство TEMPSPACE1 с помощью команды:

```
DROP TABLESPACE TEMPSPACE1
```

Можно повторно использовать контейнеры пустого табличного пространства, отбросив это табличное пространство, но прежде чем пытаться повторно использовать эти контейнеры, необходимо выполнить принятие (COMMIT) команды DROP TABLESPACE.

Задачи, связанные с данной темой:

- “Отбрасывание пользовательского табличного пространства” на стр. 187
- “Отбрасывание пользовательского временного табличного пространства” на стр. 190

Ссылки, связанные с данной темой:

- “CREATE TABLESPACE statement” в *SQL Reference, Том 2*
- “DROP statement” в *SQL Reference, Том 2*

Отбрасывание пользовательского временного табличного пространства

Процедура:

Пользовательское временное табличное пространство можно отбросить лишь в том случае, если в нем не объявлены никакие временные таблицы. При отбрасывании табличного пространства не делается попытка отбросить все объявленные в нем временные таблицы.

Примечание: Объявленная временная таблица неявно отбрасывается, когда объявившая ее прикладная программа отсоединяется от базы данных.

Задачи, связанные с данной темой:

- “Отбрасывание пользовательского табличного пространства” на стр. 187
- “Отбрасывание системного временного табличного пространства” на стр. 188

Ссылки, связанные с данной темой:

- “DROP statement” в *SQL Reference, Том 2*

Отбрасывание схемы

Процедура:

Перед отбрасыванием схемы необходимо отбросить все объекты в этой схеме или переместить их в другую схему. При выполнении оператора DROP имя заданной в нем схемы должно быть в каталоге, в противном случае будет возвращен код ошибки.

Чтобы отбросить схему с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Схемы**.
2. Щелкните правой кнопкой мыши по схеме, которую нужно отбросить, и выберите из всплывающего меню пункт **Отбросить**.
3. Включите переключатель **Подтверждение** и нажмите кнопку **OK**.

Чтобы отбросить схему из командной строки, введите команду:

```
DROP SCHEMA <имя>
```

В следующем примере отбрасывается схема ″joeschma″:

```
DROP SCHEMA joeschma RESTRICT
```

Ключевое слово RESTRICT задает правило, в соответствии с которым в удаляемой из базы данных схеме не должны быть определены никакие объекты.

Ссылки, связанные с данной темой:

- “DROP statement” в *SQL Reference, Том 2*

Изменение структуры и содержимого таблицы

Задачи по изменению структуры и содержимого таблицы:

- “Сжатие существующих таблиц”
- “Добавление столбцов в существующую таблицу” на стр. 192
- “Изменение определения столбца” на стр. 193
- “Удаление строк из таблицы или производной таблицы” на стр. 194
- “Изменение определения столбца идентификации” на стр. 195
- “Изменение ограничения” на стр. 196
- “Определение генерируемого столбца в существующей таблице” на стр. 203
- “Объявление таблицы нестабильного объема” на стр. 207
- “Изменение ключей разделения” на стр. 208
- “Изменение атрибутов таблицы” на стр. 209
- “Изменение свойств материализованной таблицы запроса” на стр. 212
- “Обновление данных в материализованной таблице запроса” на стр. 213

Учтите, что нельзя изменять триггеры для таблиц; необходимо отбросить не соответствующие задачам триггеры (смотрите раздел “Отбрасывание триггера” на стр. 218) и создать вместо них новые (смотрите раздел “Создание триггера” на стр. 125).

Сжатие существующих таблиц

Существующую таблицу можно изменить таким образом, чтобы в ней применялся сжатый формат строк. Общий размер столбцов в байтах в сжатом формате строк может превышать общий размер столбцов в байтах в исходном формате строк (т.е. формате, не допускающем сжатие), если общий размер в байтах не превышает допустимую длину строки таблицы в табличном пространстве. Например, в табличном пространстве с размером страницы 4 Кб допустимая длина строки составляет 4005 байт. Если допустимая длина строки будет превышена, то будет возвращено сообщение об ошибке SQL0670N. Формула для подсчета размера в байтах определена в операторе CREATE TABLE.

Аналогично, строки существующей таблицы можно преобразовать из сжатого формата в обычный формат. При этом действует то же правило относительно общего размера столбцов в байтах, и при необходимости отправляется сообщение об ошибке SQL0670N.

Сжатый формат строк рекомендуется установить для тех таблиц, в которых большинство значений совпадает с системными значениями по умолчанию или представляют собой пустое значение. Например, если таблица содержит столбец INTEGER, 90% значений которого равно 0 (значение по умолчанию для типа данных INTEGER) или NULL, то сжатие этой таблицы и этого столбца позволит сэкономить значительный объем дисковой памяти.

Для того чтобы установить сжатый формат строк на уровне таблицы и, возможно, на уровне столбца, укажите при изменении таблицы условие VALUE COMPRESSION. Условие ACTIVATE VALUE COMPRESSION означает, что нужно включить сжатие данных таблицы, а условие DEACTIVATE VALUE COMPRESSION означает, что нужно выключить сжатие данных таблицы.

Если задано условие DEACTIVATE VALUE COMPRESSION, то неявно будут выключены все опции COMPRESS SYSTEM DEFAULT, заданные для столбцов таблицы.

Новый формат будет применяться всеми строками, вставленными, загруженными и обновленными после изменения формата строк таблицы. Для того чтобы все строки были преобразованы в новый формат, необходимо реорганизовать таблицу или обновить существующие строки перед изменением формата строк.

Понятия, связанные с данным:

- “Сжатие новых таблиц” на стр. 100

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*
- “CREATE TABLE statement” в *SQL Reference, Том 2*

Добавление столбцов в существующую таблицу

Процедура:

Определение столбца включает в себя имя столбца, тип данных и все необходимые ограничения.

При добавлении столбцов в таблицу эти столбцы логически помещаются справа от самого правого уже существующего определения столбца. При добавлении в существующую таблицу нового столбца изменяется только описание таблицы в

системном каталоге, это не оказывает немедленного влияния на обращение к данным таблицы. Существующие записи физически изменяются только при выполнении оператора UPDATE. При получении из таблицы существующей строки для нового столбца в зависимости от его определения возвращается пустое значение или значение по умолчанию. Столбцы, добавляемые после создания таблицы, нельзя определять как NOT NULL: они должны быть определены как NOT NULL WITH DEFAULT или как столбцы, которые могут содержать пустые значения.

Чтобы добавить столбцы в существующую таблицу с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой по таблице, в которую нужно добавить столбцы, и выберите из всплывающего меню пункт **Изменить**.
3. Перейдите на страницу **Столбцы**, введите информацию для нового столбца и нажмите кнопку **OK**.

Чтобы добавить столбцы в существующую таблицу с помощью командной строки, введите команду:

```
ALTER TABLE <имя_таблицы>
   ADD <имя_столбца> <тип_данных> <атрибут_пустого_значения>
```

Для добавления столбцов можно использовать оператор SQL. В следующем операторе используется оператор ALTER TABLE для добавления трех столбцов к таблице EMPLOYEE:

```
ALTER TABLE EMPLOYEE
   ADD MIDINIT CHAR(1)   NOT NULL WITH DEFAULT
   ADD HIREDATE DATE
   ADD WORKDEPT CHAR(3)
```

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*

Изменение определения столбца

Процедура:

Можно изменить характеристики столбца, увеличив длину существующего столбца VARCHAR. Число символов, до которого можно увеличить размер столбца, зависит от используемого размера страницы.

Чтобы изменить столбцы в существующей таблице с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. В списке таблиц в правой панели щелкните правой кнопкой по таблице, в которой нужно изменить столбцы, и выберите из всплывающего меню пункт **Изменить**.
3. Перейдите на страницу **Столбцы**, выберите столбец и нажмите кнопку **Изменить**.
4. В поле **Длина** введите новую длину этого столбца (в байтах) и нажмите кнопку **OK**.

Чтобы изменить столбцы в существующей таблице из командной строки, введите команду:

```
ALTER TABLE ALTER COLUMN
   <имя_столбца> <тип_изменения>
```

Например, чтобы увеличить длину столбца до 4000 символов, используйте оператор, подобный следующему:

```
ALTER TABLE ALTER COLUMN
   COLNAM1 SET DATA TYPE VARCHAR(4000)
```

Нельзя изменить столбцы типизированной таблицы. Однако можно задать область видимости для существующего столбца ссылочного типа, для которого область видимости еще не определена. Например:

```
ALTER TABLE ALTER COLUMN
   COLNAMT1 ADD SCOPE TYPTAB1
```

Задачи, связанные с данной темой:

- “Изменение определения столбца идентификации” на стр. 195

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*

Удаление строк из таблицы или производной таблицы

Процедура:

Можно изменять содержимое таблицы или производной таблицы, удаляя из нее строки. Удаление строки из производной таблицы удаляет строки из таблицы, на которой основана эта производная таблица. Оператор DELETE используется, чтобы:

- Удалить одну или несколько строк, найденных по условию поиска. Это называется *DELETE с поиском*.
- Удалить ровно одну строку, определенную текущим положением указателя. Это называется *позиционным DELETE*.

Оператор DELETE может быть встроен в прикладную программу или выполняться как динамический оператор SQL.

Если модифицируемая таблица связана с другими таблицами реляционными связями, удаление строк имеет некоторые особенности. Если указанная таблица или базовая таблица указанной производной таблицы является родительской таблицей, при правиле удаления RESTRICT у выбранных для удаления строк не должно быть никаких зависимых. Более того, при правиле удаления RESTRICT удаление не должно вызывать каскадное удаление зависимых строк через отношение зависимости.

Если операция удаления не нарушает правила удаления RESTRICT, выбранные строки удаляются.

Например, чтобы удалить отдел (DEPTNO) “D11” из таблицы отделов (DEPARTMENT), воспользуйтесь командой:

```
DELETE FROM department WHERE deptno='D11'
```

Если при выполнении многострочного удаления происходит ошибка, в таблице не производится никаких изменений. Если происходит ошибка, препятствующая удалению всех строк, удовлетворяющих условию поиска и всем операциям, определяемым существующими реляционными ограничениями, в таблицах не производится никаких изменений.

При успешном выполнении оператора DELETE устанавливается одна или несколько монопольных блокировок, если такие блокировки еще не существуют. Блокировки снимаются после оператора COMMIT или ROLLBACK. Блокировки могут не дать другим прикладным программам выполнять операций над таблицей.

Понятия, связанные с данным:

- “Блокировки и управление одновременностью” в *Руководство администратора: Производительность*
- “Блокировки и производительность” в *Руководство администратора: Производительность*
- “Факторы, влияющие на блокировку” в *Руководство администратора: Производительность*
- “Рекомендации по настройке блокировок” в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- “DELETE statement” в *SQL Reference, Том 2*

Изменение определения столбца идентификации

Процедура:

Если вы повторно создаете таблицу после операции импорта или загрузки и в этой таблице есть столбец IDENTITY, то он будет переустановлен, чтобы после повторного создания содержимого таблицы генерация значений IDENTITY началась с 1. При вставке строк в эту повторно создаваемую таблицу значения в столбце IDENTITY не должны снова начаться с 1. В столбце IDENTITY не должно быть повторений значений. Чтобы этого не случилось, надо:

1. Повторно создать таблицу.
2. Загрузить в эту таблицу данные с условием MODIFIED BY IDENTITYOVERRIDE. Данные загрузятся в таблицу, но для строк не будут сгенерированы значения идентификации.
3. Выполнить запрос для получения последнего значения счетчика для столбца IDENTITY:

   ```
   SELECT MAX(<столбец IDENTITY>)
   ```

   Этот запрос возвратит значение, эквивалентное тому, которое могло бы быть в столбце IDENTITY этой таблицы.
4. Задайте в операторе ALTER TABLE условие RESTART:

   ```
   ALTER TABLE <имя таблицы> ALTER COLUMN <столбец IDENTITY>
      RESTART WITH <последнее значение счетчика>
   ```
5. Вставьте в таблицу новую строку. Значение столбца IDENTITY будет сгенерировано на основе значения, указанного в условии RESTART WITH.

Ссылки, связанные с данной темой:

- “MAX aggregate function” в *SQL Reference, Том 1*
- “ALTER TABLE statement” в *SQL Reference, Том 2*
- “LOAD Command” в *Command Reference*

Изменение ограничения

Чтобы изменить ограничения, надо отбросить их и создать вместо них новые. Дополнительную информацию смотрите в разделах:

- “Добавление ограничения”
- “Отбрасывание уникального ограничения” на стр. 200

Добавление ограничения

Для добавления ограничений используется оператор ALTER TABLE. Дополнительную информацию об этом операторе, включая его синтаксис, смотрите в руководстве *SQL Reference*.

Добавление ограничения уникальности:

Процедура:

В существующую таблицу можно добавить ограничения уникальности. Имя ограничения не должно совпадать с именами других ограничений, заданных в

этом операторе ALTER TABLE, и должно быть уникальным в таблице (среди имен всех определенных ограничений реляционной целостности). Перед завершением оператора существующие данные проверяются на соответствие новому условию.

Следующий оператор SQL добавляет в таблицу EMPLOYEE ограничение уникальности, представляющее новый способ уникальной идентификации сотрудников в этой таблице:

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT NEWID UNIQUE(EMPNO,HIREDATE)
```

Задачи, связанные с данной темой:

- “Определение ограничения уникальности” на стр. 103
- “Отбрасывание ограничения уникальности” на стр. 200

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*

Добавление первичных ключей:

Процедура:

Чтобы добавить ограничения к большой таблице, лучше перевести таблицу в состояние отложенной проверки, добавить ограничения и затем проверить таблицу, получив общий список нарушающих правила строк. Чтобы явно задать состояние отложенной проверки, используйте оператор SET INTEGRITY (если это родительская таблица, все зависимые и дочерние таблицы будут неявно переведены в состояние отложенной проверки).

Чтобы добавить первичные ключи с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой по таблице, которую нужно изменить, и выберите из всплывающего меню пункт **Изменить**.
3. На странице **Первичный ключ** выберите один или несколько столбцов в качестве первичных ключей и нажмите кнопку со стрелкой, чтобы переместить их.
4. Необязательно: Введите имя ограничения этого первичного ключа.
5. Нажмите кнопку **OK**.

Чтобы добавить первичные ключи из командной строки, введите команду:

```
ALTER TABLE <имя>
   ADD CONSTRAINT <имя_столбца>
   PRIMARY KEY <имя_столбца>
```

Задачи, связанные с данной темой:

- “Добавление внешних ключей” на стр. 198

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*
- “SET INTEGRITY statement” в *SQL Reference, Том 2*

Добавление внешних ключей:

Процедура:

После добавления в таблицу внешнего ключа могут быть отмечены как недействительные пакеты и кэшируемые динамические операторы SQL, содержащие:

- Операторы, вставляющие или изменяющие данные в таблице, содержащей этот внешний ключ
- Операторы, изменяющие или удаляющие данные в родительской таблице.

Чтобы добавить внешние ключи с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой по таблице, которую нужно изменить, и выберите из всплывающего меню пункт **Изменить**.
3. На странице **Внешние ключи** нажмите кнопку **Добавить**.
4. В окне **Добавить внешние ключи** задайте информацию о родительской таблице.
5. Выберите один или несколько столбцов в качестве внешних ключей и нажмите кнопку со стрелкой, чтобы переместить их.
6. Задайте действие, выполняемое с зависимой таблицей при удалении или изменении строки родительской таблицы. Для внешнего ключа можно также задать имя ограничения.
7. Нажмите кнопку **OK**.

Чтобы добавить внешние ключи из командной строки, введите команду:

```
ALTER TABLE <имя>
   ADD CONSTRAINT <имя_столбца>
    FOREIGN KEY <имя_столбца>
    ON DELETE <тип_действия>
    ON UPDATE <тип_действия>
```

В следующих примерах показано использование оператора ALTER TABLE для добавления в таблицу первичных ключей и внешних ключей:

```
ALTER TABLE PROJECT
  ADD CONSTRAINT PROJECT_KEY
      PRIMARY KEY (PROJNO)
```

```
ALTER TABLE EMP_ACT
  ADD CONSTRAINT ACTIVITY_KEY
      PRIMARY KEY (EMPNO, PROJNO, ACTNO)
  ADD CONSTRAINT ACT_EMP_REF
      FOREIGN KEY (EMPNO)
      REFERENCES EMPLOYEE
      ON DELETE  RESTRICT
  ADD CONSTRAINT ACT_PROJ_REF
      FOREIGN KEY (PROJNO)
      REFERENCES PROJECT
      ON DELETE CASCADE
```

Понятия, связанные с данным:

- “Зависимости операторов при изменении объектов” на стр. 227

Задачи, связанные с данной темой:

- “Добавление первичных ключей” на стр. 197

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*

Добавление проверочного ограничения для таблицы:

Процедура:

Проверочные ограничения можно добавить в существующую таблицу с помощью оператора ALTER TABLE. Имя ограничения не должно совпадать с именами других ограничений, заданных в этом операторе ALTER TABLE, и должно быть уникальным в таблице (среди имен всех определенных ограничений реляционной целостности). Перед завершением оператора существующие данные проверяются на соответствие новому условию.

Чтобы добавить ограничения к большой таблице, лучше перевести таблицу в состояние отложенной проверки, добавить ограничения и затем проверить таблицу, получив общий список нарушающих правила строк. Чтобы явно задать состояние отложенной проверки, используйте оператор SET INTEGRITY (если это родительская таблица, все зависимые и дочерние таблицы будут неявно переведены в состояние отложенной проверки).

После добавления в таблицу проверочного ограничения таблицы могут быть отмечены как недействительные пакеты и кэшируемые динамические операторы SQL, выполняющие вставку или изменение данных этой таблицы.

Чтобы добавить проверочное ограничение таблицы с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой по таблице, которую нужно изменить, и выберите из всплывающего меню пункт **Изменить**.
3. На странице **Проверочное ограничения** нажмите кнопку **Добавить**.
4. Введите необходимую информацию в окне **Добавить проверочное ограничение** и нажмите кнопку **OK**.
5. На странице **Проверочное ограничения** нажмите кнопку **OK**.

Чтобы добавить проверочное ограничение таблицы из командной строки, введите команду:

```
ALTER TABLE <имя>
   ADD CONSTRAINT <имя> (<ограничение>)
```

Следующий оператор SQL добавляет в таблицу EMPLOYEE ограничение, требующее, чтобы сумма зарплата и комиссионных была больше $25000:

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT REVENUE CHECK (SALARY + COMM > 25000)
```

Понятия, связанные с данным:

- “Зависимости операторов при изменении объектов” на стр. 227

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*
- “SET INTEGRITY statement” в *SQL Reference, Том 2*

Отбрасывание уникального ограничения

Для отбрасывания ограничений используется оператор ALTER TABLE. Дополнительную информацию об этом операторе, включая его синтаксис, смотрите в руководстве *SQL Reference*.

Отбрасывание ограничения уникальности:

Процедура:

С помощью оператора ALTER TABLE можно явно отбросить ограничение уникальности. Имена всех ограничений уникальности таблицы можно найти в производной таблице системного каталога SYSCAT.INDEXES.

Следующий оператор SQL отбрасывает ограничение уникальности NEWID из таблицы EMPLOYEE:

```
ALTER TABLE EMPLOYEE
   DROP UNIQUE NEWID
```

При удалении этого ограничения уникальности делаются недействительными все пакеты или кэшируемые динамические операторы SQL, которые используют это ограничение.

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*

Отбрасывание первичных ключей:

Процедура:

Чтобы отбросить первичный ключ с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой по таблице, которую нужно изменить, и выберите из всплывающего меню пункт **Изменить**.
3. В правой части страницы **Первичный ключ** выберите первичный ключ, который нужно удалить, и нажмите кнопку со стрелкой, чтобы переместить его в поле **Доступные столбцы** в левой части страницы.
4. Нажмите кнопку **OK**.

Чтобы отбросить первичный ключ из командной строки, введите команду:

```
ALTER TABLE <имя>
   DROP PRIMARY KEY
```

После отбрасывания ограничения внешнего ключа могут быть отмечены как недействительные пакеты или кэшируемые динамические операторы SQL, содержащие:

- Операторы, вставляющие или изменяющие данные в таблице, содержащей этот внешний ключ
- Операторы, изменяющие или удаляющие данные в родительской таблице.

Понятия, связанные с данным:

- “Зависимости операторов при изменении объектов” на стр. 227

Задачи, связанные с данной темой:

- “Отбрасывание внешнего ключа” на стр. 201

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*

Отбрасывание внешнего ключа:

Процедура:

Чтобы отбросить внешний ключ с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой по таблице, которую нужно изменить, и выберите из всплывающего меню пункт **Изменить**.
3. На странице **Внешние ключи** нажмите кнопку **Добавить**.
4. В правой части страницы выберите внешний ключ, который нужно удалить, и нажмите кнопку со стрелкой, чтобы переместить его в поле **Доступные столбцы** в левой части страницы.
5. На странице **Внешние ключи** нажмите кнопку **OK**.

Чтобы отбросить внешний ключ из командной строки, введите команду:

```
ALTER TABLE <имя>
   DROP FOREIGN KEY <имя_внешнего_ключа>
```

В следующих примерах для отбрасывания из таблицы первичных ключей и внешних ключей используются условия DROP PRIMARY KEY и DROP FOREIGN KEY оператора ALTER TABLE:

```
ALTER TABLE EMP_ACT
  DROP PRIMARY KEY
  DROP FOREIGN KEY ACT_EMP_REF
  DROP FOREIGN KEY ACT_PROJ_REF
ALTER TABLE PROJECT
  DROP PRIMARY KEY
```

Понятия, связанные с данным:

- “Зависимости операторов при изменении объектов” на стр. 227

Задачи, связанные с данной темой:

- “Отбрасывание первичных ключей” на стр. 201

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*

Отбрасывание проверочного ограничения для таблицы:

Процедура:

Проверочное ограничение таблицы можно явно отбросить или изменить с помощью оператора ALTER TABLE или же неявно отбросить при выполнении оператора DROP TABLE.

После отбрасывания проверочного ограничения таблицы становятся недействительными все пакеты и кэшируемые динамические операторы SQL, выполняющие вставку или изменение данных этой таблицы. Имена всех проверочных ограничений таблицы можно найти в производной таблице каталога SYSCAT.CHECKS. Перед тем как отбрасывать проверочное ограничение таблицы, имя которого сгенерировано системой, найдите это имя в производной таблице каталога SYSCAT.CHECKS.

Чтобы отбросить проверочное ограничение таблицы с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой по таблице, которую нужно изменить, и выберите из всплывающего меню пункт **Изменить**.
3. На странице **Проверочное ограничения** выберите проверочное ограничение, которое нужно отбросить, нажмите кнопку **Удалить** и затем нажмите кнопку **OK**.

Чтобы отбросить проверочное ограничение таблицы из командной строки, введите команду:

```
ALTER TABLE <имя_таблицы>
   DROP CHECK <имя_проверочного_ограничения>
```

Следующий оператор SQL отбрасывает табличное проверочное ограничение REVENUE из таблицы EMPLOYEE:

```
ALTER TABLE EMPLOYEE
   DROP CHECK REVENUE
```

Понятия, связанные с данным:

- “Зависимости операторов при изменении объектов” на стр. 227

Задачи, связанные с данной темой:

- “Добавление проверочного ограничения для таблицы” на стр. 199

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*

Определение генерируемого столбца в существующей таблице

Генерируемый столбец определяется в базовой таблице, в которой сохраняемое значение не задается операциями вставки или изменения, а вычисляется по некоторой формуле. Генерируемый столбец можно создать при создании таблицы или при изменении существующей таблицы.

Предварительные требования для установки:

Генерируемые столбцы можно определить только с типами данных, для которых определена операция сравнения. Типы данных, которые нельзя использовать для генерируемых столбцов: структурированные типы, LOB, CLOB, DBCLOB, LONG VARCHAR, LONG VARGRAPHIC и пользовательские типы, определенные на основе этих типов.

Генерируемые столбцы нельзя использовать в ограничениях, уникальных индексах, реляционных ограничениях, первичных ключах и глобальных временных таблицах. Таблицы, созданные с опцией LIKE, и материализованные производные таблицы не наследуют свойства генерируемых столбцов.

Ограничения:

Для вставки или изменения значений генерируемых столбцов должно использоваться ключевое слово DEFAULT. Когда при вставке используется ключевое слово DEFAULT, не нужно перечислять эти столбцы в списке столбцов. Вместо этого для них можно задать DEFAULT (значение по умолчанию) в списке значений. Когда ключевое слово DEFAULT используется при изменении строк, оно позволяет заново вычислять значения генерируемых столбцов, помещенные командой SET INTEGRITY без проверки.

Порядок обработки триггеров требует, чтобы генерируемые столбцы не использовались в триггерах BEFORE в их заголовках (перед обновлением) или телах. В соответствии с порядком обработки генерируемые столбцы обрабатываются после триггеров BEFORE.

Утилита db2look не видит проверочные ограничения, генерируемые генерируемым столбцом.

При использовании репликации таблица назначения не должна использовать в своем отображении генерируемые столбцы. При репликации есть две возможности:

- В таблице назначения надо вместо генерируемого столбца определять обычный (то есть не генерируемый) столбец
- Генерируемый столбец должен быть опущен из отображения таблицы назначения

Ограничения при работе с генерируемыми столбцами:

- Генерируемые столбцы не должны зависеть друг от друга.
- Используемые для создания генерируемых столбцов выражения не должны содержать подзапросы. Сюда относятся выражения с функциями, выполняющими операцию READS SQL DATA.
- Для генерируемых столбцов не разрешены проверочные ограничения.

Процедура:

Чтобы определить генерируемый столбец, выполните следующие шаги:

1. Переведите таблицу в состояние отложенной проверки.

   ```
   SET INTEGRITY FOR t1 OFF
   ```

2. Измените таблицу, добавив один или несколько генерируемых столбцов.

   ```
   ALTER TABLE t1 ADD COLUMN c3 DOUBLE GENERATED ALWAYS AS (c1 + c2),
                  ADD COLUMN c4 GENERATED ALWAYS AS
                     (CASE WHEN c1 > c3 THEN 1 ELSE NULL END))
   ```

3. Присвойте правильные значения генерируемым столбцам. Это можно сделать следующими способами:

   - Заново вычислите и присвойте значения генерируемым столбцам с помощью следующего оператора:

     ```
     SET INTEGRITY FOR t1 IMMEDIATE CHECKED FORCE GENERATED
     ```

     Если оператор SET INTEGRITY не будет выполнен из-за недостатка свободного пространства в журнале, то увеличьте объем доступной памяти для активного журнала и заново вызовите оператор SET INTEGRITY.

     Примечание: В этот момент можно использовать таблицы исключений.

   - Если объем доступной памяти для активного журнала увеличить нельзя, то присвойте генерируемым столбцам значения по умолчанию с помощью операторов обновления.

     a. Установите монопольную блокировку этой таблицы. Это запретит все обращения к этой таблице, кроме транзакций чтения без принятия. Обратите внимание, что таблица будет разблокирована после выполнения первой промежуточной операции принятия. В связи с этим прочие транзакции смогут прочитать строки, содержащие сгенерированные столбцы, которым еще не присвоены их значения по умолчанию.

        ```
        LOCK TABLE t1
        ```

     b. Не выполняйте проверку сгенерированных столбцов

        ```
        SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
        ```

     c. При необходимости проверьте, что в таблице отсутствуют другие нарушения целостности, и выведите таблицу из состояния ожидания проверки

        ```
        SET INTEGRITY FOR t1 IMMEDIATE CHECKED
        ```

     d. Обновите генерируемые столбцы, используя чередующиеся предикаты и операции принятия, чтобы предотвратить переполнение журналов.

        ```
        UPDATE t1 SET (c3, c4) = (DEFAULT, DEFAULT) WHERE <предикат>
        ```

     e. Разблокируйте таблицу, завершив транзакцию с помощью оператора принятия.

        ```
        COMMIT
        ```

- Другой способ, которым можно воспользоваться, если нельзя увеличить объем памяти активного журнала, основан на применении указателя:
  a. Определите указатель FOR UPDATE для таблицы. Если блокировки не должны сниматься после выполнения промежуточной операции принятия, укажите опцию WITH HOLD.

     ```
     DECLARE C1 CURSOR WITH HOLD FOR S1
     ```

     Где S1 определена следующим образом:

     ```
     SELECT '0' FROM t1 FOR UPDATE OF C3, C4
     ```

  b. Откройте курсор.

     ```
     OPEN C1
     ```

  c. Не выполняйте проверку сгенерированных столбцов

     ```
     SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
     ```

  d. При необходимости проверьте, что в таблице отсутствуют другие нарушения целостности, и выведите таблицу из состояния ожидания проверки

     ```
     SET INTEGRITY FOR t1 IMMEDIATE CHECKED
     ```

  e. В цикле переберите все строки таблицы, выполняя указанный ниже оператор для каждой строки. Этот оператор присваивает генерируемым столбцам значения по умолчанию. Первая выборка должна быть сделана сразу после того, как таблица будет выведена из состояния ожидания проверки. В этом случае таблица будет заблокирована на протяжении всего указателя.

     ```
     UPDATE t1 SET (C3, C4) = (DEFAULT, DEFAULT) WHERE CURRENT OF C1
     ```

     Выполните промежуточную операцию принятия, чтобы избежать переполнения журналов.

  f. Закройте курсор и выполните операцию принятия, чтобы разблокировать таблицу.

     ```
     CLOSE C1
     COMMIT
     ```

- Известно, что таблица была создана с опцией NOT LOGGED INITIALLY. В этом случае для таблицы отключена запись информации в журналы, что вызывает обычные последствия и риск при работе со значениями генерируемых столбцов.
  a. Активируйте опцию NOT LOGGED INITIALLY.

     ```
     ALTER TABLE t1 ACTIVATE NOT LOGGED INITIALLY
     ```

  b. Сгенерируйте значения.

     ```
     SET INTEGRITY FOR t1 IMMEDIATE CHECKED FORCE GENERATION
     ```

  c. Опять отключите опцию NOT LOGGED INITIALLY, выполнив принятие транзакции.

     ```
     COMMIT
     ```

Значения для генерируемых столбцов можно также просто проверить, применив выражение, как при проверке проверочного ограничения равенства:

```
SET INTEGRITY FOR t1 IMMEDIATE CHECKED
```

Если в генерируемый столбец были помещены значения, например, с помощью операции LOAD, и известно, что эти значения совпадают с результатами выражения генерации, таблицу можно вывести из состояния отложенной проверки, не выполняя проверку или присваивание значений:

```
SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
```

Задачи, связанные с данной темой:

- “Определение генерируемых столбцов в новых таблицах” на стр. 111

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*
- “COMMIT statement” в *SQL Reference, Том 2*
- “LOCK TABLE statement” в *SQL Reference, Том 2*
- “SET INTEGRITY statement” в *SQL Reference, Том 2*
- “UPDATE statement” в *SQL Reference, Том 2*
- “db2look - DB2 Statistics and DDL Extraction Tool Command” в *Command Reference*
- “db2gncol - Update Generated Column Values Command” в *Command Reference*

Объявление таблицы нестабильного объема

Процедура:

Таблица *нестабильного объема* - это таблица, объем содержимого которой может во время работы меняться от нулевого до очень большого. Нестабильность объема или чрезвычайное непостоянство содержимого такого типа таблиц делает ненадежными показатели статистики, собранные функцией RUNSTATS. Статистика собирается в определенный момент времени и отражает состояние таблицы только на этот момент времени. При создании плана доступа, использующего таблицу нестабильного объема, может получиться неверный план или план с плохой производительностью. Например, если показатели статистики собраны в момент, когда таблица нестабильного объема была пуста, оптимизатор предпочтет для доступа к этой таблице использовать сканирование таблицы, а не сканирование индексов.

Чтобы предотвратить подобные ситуации, можно с помощью оператора ALTER TABLE объявить таблицу таблицей нестабильного объема. Если таблица объявлена таблицей нестабильного объема, оптимизатор будет пытаться использовать для нее сканирование индексов вместо сканирования таблицы.

Планы доступа, использующие объявленные таблицы нестабильного объема, не будут зависеть от существующей статистики для этих таблиц.

Чтобы объявить таблицу таблицей нестабильного объема с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой по таблице, которую нужно изменить, и выберите из всплывающего меню пункт **Изменить**.
3. На странице **Таблица** включите переключатель **Объем таблицы может сильно меняться во время работы** и нажмите кнопку **OK**.

Чтобы объявить таблицу "таблицей нестабильного объема" из командной строки, введите команду:

```
ALTER TABLE <имя_таблицы>
   VOLATILE CARDINALITY
```

Ссылки, связанные с данной темой:

- "ALTER TABLE statement" в *SQL Reference, Том 2*

Изменение ключей разделения

Процедура:

Ключ разделения можно изменить только для таблиц в одноразделных группах узлов. Сначала отбросьте существующий ключ разделения, а затем создайте новый.

Следующий оператор SQL отбрасывает ключ разделения MIX_INT из таблицы MIXREC:

```
ALTER TABLE MIXREC
   DROP PARTITIONING KEY
```

Нельзя изменить ключ разделения таблицы в многораздельной группе узлов. При попытке сделать это возникает ошибка.

Чтобы изменить ключ разделения для многораздельной группы узлов, можно:

- Экспортировать все данные в одноразделную группу узлов и затем следовать представленным выше инструкциям.
- Или экспортировать все данные, отбросить таблицу, заново создать таблицу, переопределив ключ разделения, и затем импортировать все данные.

Ни один из этих методов не удобен для крупных баз данных, поэтому перед созданием крупных баз данных важно определить подходящий ключ разделения.

Понятия, связанные с данным:

- “Ключи разделения” в *Руководство администратора: Планирование*

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*

Изменение атрибутов таблицы

Процедура:

Вам может потребоваться изменить атрибуты таблицы, такие как опцию захвата данных, процент свободного пространства на каждой странице (PCTFREE), размер блокировок или режим добавления.

Объем свободного пространства, оставляемого на каждой странице таблицы, задается параметром PCTFREE; это важный фактор эффективности использования индексов кластеризации. Правильное значение зависит от природы существующих и ожидаемых будущих данных. Значение параметра PCTFREE влияет на выполнение операций LOAD и REORG, но игнорируется операциями вставки, изменения и импорта.

Увеличение значения PCTFREE позволит сохранить кластеризацию в течение более длительного времени, однако для его применения потребуется больше дисковой памяти.

С помощью параметра LOCKSIZE можно задать размер блокировок, используемый при обращениях к этой таблице. По умолчанию при создании таблицы задается использование блокировок уровня строки. Использование блокировок уровня таблицы может улучшить производительность запросов, так как ограничивает число блокировок, которые приходится устанавливать и освобождать.

Задав опцию APPEND ON, можно улучшить общую производительность для этой таблицы. Она ускоряет выполнение операций вставки, устраняя необходимость поддерживать информацию о свободном пространстве.

Таблицу с индексом кластеризации нельзя перевести в режим добавления. Аналогично индекс кластеризации нельзя создать для таблицы в режиме добавления.

Понятия, связанные с данным:

- “Блокировки и управление одновременностью” в *Руководство администратора: Производительность*
- “Атрибуты блокировки” в *Руководство администратора: Производительность*

- “Блокировки и производительность” в *Руководство администратора: Производительность*
- “Факторы, влияющие на блокировку” в *Руководство администратора: Производительность*
- “Рекомендации по настройке блокировок” в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*

Изменение столбца идентификации

Процедура:

Для изменения атрибутов существующего столбца идентификации используйте оператор ALTER TABLE.

Есть разные способы изменить столбец идентификации, чтобы придать ему некоторые свойства последовательностей.

Некоторые задачи уникальны для оператора ALTER TABLE и столбца идентификации:

- RESTART сбрасывает последовательность, связанную со столбцом идентификации, к значению, неявно или явно заданному в качестве начального при создании столбца идентификации.
- RESTART WITH <числовая-константа> сбрасывает последовательность, связанную со столбцом идентификации, к значению числовой константы. Числовая константа может представлять собой любое положительное или отрицательное значение без ненулевых цифр после десятичной точки, которое можно присвоить столбцу идентификации.

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*

Изменение последовательности

Процедура:

Для изменения атрибутов существующей последовательности используется оператор ALTER SEQUENCE.

Возможны следующие изменения атрибутов последовательности:

- Изменение приращения между будущими значениями
- Установка новых минимальных или максимальных значений

- Изменение числа членов последовательности, сохраняемых в кэше
- Установка или отмена циклического повторения последовательности
- Настройка или отмена генерации членов последовательности в порядке запросов
- Перезапуск последовательности

Есть две задачи, не встречающиеся в ходе создания последовательности. Это:

- RESTART. Сбрасывает последовательность к значению, неявно или явно заданному в качестве начального значения при создании последовательности.
- RESTART WITH <числовая-константа>. Сбрасывает последовательность к значению числовой константы точного типа. Числовая константа может представлять собой любое положительное или отрицательное значение без ненулевых цифр после десятичной точки.

После перезапуска последовательности или задания циклического повторения могут генерироваться повторные члены последовательности. Оператор ALTER SEQUENCE влияет только на будущие члены последовательности.

Тип данных последовательности изменять нельзя. Вместо этого нужно отбросить существующую последовательность и затем создать новую последовательность, выбрав новый тип данных.

Все кэшируемые значения последовательности, не используемые DB2, при изменении последовательности теряются.

Задачи, связанные с данной темой:

- “Отбрасывание последовательности” на стр. 211

Ссылки, связанные с данной темой:

- “ALTER SEQUENCE statement” в *SQL Reference, Том 2*

Отбрасывание последовательности

Процедура:

Чтобы удалить последовательность, используйте оператор DROP.

Конкретную последовательность можно отбросить так:

```
DROP SEQUENCE имя_последовательности
```

где `имя_последовательности` - имя отбрасываемой последовательности, включая имя явной или неявной схемы для точного указания существующей последовательности.

Последовательности, создаваемые системой для столбцов идентификации (IDENTITY), нельзя отбрасывать при помощи оператора DROP SEQUENCE.

После того, как последовательность отброшена, все привилегии в отношении этой последовательности также отбрасываются.

Задачи, связанные с данной темой:

- “Изменение последовательности” на стр. 210

Ссылки, связанные с данной темой:

- “DROP statement” в *SQL Reference, Том 2*

Изменение свойств материализованной таблицы запроса

Процедура:

С учетом некоторых ограничений, материализованную таблицу запроса можно преобразовать в обычную таблицу, и наоборот. Тип других таблиц изменить нельзя. Преобразовать можно только обычные таблицы и материализованные таблицы запроса. Например, реплицированную материализованную таблицу запроса нельзя преобразовать в обычную таблицу, и наоборот.

После преобразования обычной таблицы в материализованную таблицу запроса она переводится в состояние отложенной проверки. При выполнении такого преобразования полная выборка в определении материализованной таблицы запроса должна совпадать с определением исходной таблицы, то есть:

- Должно совпадать число столбцов.
- Должны совпадать имена и позиции столбцов.
- Должны совпадать типы данных.

Если на основе исходной таблицы определена материализованная таблица запроса, то исходную таблицу нельзя преобразовать в материализованную таблицу запроса. Если для исходной таблицы созданы триггеры, проверочные ограничения, реляционные ограничения или индекс уникальности, то ее нельзя преобразовать в материализованную таблицу запроса. Оператор ALTER TABLE, преобразующий таблицу в материализованную таблицу запроса, нельзя использовать для изменения других свойств таблицы.

При преобразовании обычной таблицы в материализованную таблицу запроса полная выборка из определения материализованной таблицы запроса не должна ссылаться на исходную таблицу ни напрямую, ни косвенно, то есть посредством производных таблиц, псевдонимов и материализованных таблиц запроса.

Для преобразования материализованной таблицы запроса в обычную таблицу воспользуйтесь следующим оператором:

```
ALTER TABLE сводная_таблица
   SET SUMMARY AS DEFINITION ONLY
```

Для преобразования обычной таблицы в материализованную таблицу запроса воспользуйтесь следующим оператором:

```
ALTER TABLE обычная_таблица
   SET SUMMARY AS <полная_выборка>
```

Ограничения, которые накладываются на полную выборку при преобразовании обычной таблицы в материализованную таблицу запроса, похожи на ограничения, применяемые при создании сводной таблицы с помощью оператора CREATE SUMMARY TABLE.

Задачи, связанные с данной темой:

- “Создание материализованной таблицы запроса” на стр. 141
- “Обновление данных в материализованной таблице запроса” на стр. 213
- “Отбрасывание материализованной таблицы запроса и промежуточной таблицы” на стр. 224

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*
- “CREATE TABLE statement” в *SQL Reference, Том 2*

Обновление данных в материализованной таблице запроса

Процедура:

С помощью оператора REFRESH TABLE можно обновить данные в одной или нескольких материализованных таблицах запроса. Этот оператор может быть встроен в прикладную программу или выполняться динамически. Для выполнения этого оператора необходимо обладать полномочиями SYSADM или DBADM или привилегией CONTROL для обновляемой таблицы.

Ниже приведен пример оператора, обновляющего данные в материализованной таблице запроса:

```
REFRESH TABLE SUMTAB1
```

Задачи, связанные с данной темой:

- “Создание материализованной таблицы запроса” на стр. 141
- “Изменение свойств материализованной таблицы запроса” на стр. 212

Ссылки, связанные с данной темой:

- “REFRESH TABLE statement” в *SQL Reference, Том 2*

Изменение пользовательского структурированного типа

Процедура:

После создания структурированного типа может потребоваться добавление или удаление атрибутов этого структурированного типа. Для этого используется оператор ALTER TYPE (структурированный).

Понятия, связанные с данным:

- “User-Defined Structured Types” в *Application Development Guide: Programming Server Applications*
- “Structured Type Hierarchies” в *Application Development Guide: Programming Server Applications*

Задачи, связанные с данной темой:

- “Defining Structured Types” в *Application Development Guide: Programming Server Applications*

Ссылки, связанные с данной темой:

- “ALTER TYPE (Structured) statement” в *SQL Reference, Том 2*

Удаление и обновление строк в типизированной таблице

Строки типизированных таблиц можно удалить, используя операторы DELETE с поиском или с указателями. Строки типизированных таблиц можно изменить, используя операторы UPDATE с поиском или с указателями.

Понятия, связанные с данным:

- “Typed Tables” в *Application Development Guide: Programming Server Applications*

Ссылки, связанные с данной темой:

- “DELETE statement” в *SQL Reference, Том 2*
- “UPDATE statement” в *SQL Reference, Том 2*

Изменение имени таблицы или индекса

Вы можете изменить имя существующей таблицы в схеме, сохранив полномочия и индексы, созданные для исходной таблицы.

Предварительные требования:

Таблица или индекс, который следует переименовать, может представлять собой алиас таблицы или индекса.

Ограничения:

Изменяемое имя таблицы или индекса не должно быть именем таблицы или индекса каталога, сводной таблицы или индекса, типизированной таблицы, объявленной глобальной временной таблицы, псевдонимом или именем объекта, отличного от таблицы, производной таблицы и алиаса.

На существующую таблицу или индекс не должны ссылаться:

- Производные таблицы
- Триггеры
- Реляционные ограничения
- Сводные таблицы
- Области видимости существующих ссылочных столбцов

Эта таблица не должна также иметь проверочных ограничений или генерируемых столбцов, кроме столбца идентификации. Все пакеты и кэшируемые динамические операторы SQL, зависящие от исходной таблицы, становятся недействительными. Наконец, все алиасы, ссылающиеся на исходную таблицу, не изменяются.

Следует проверить соответствующие таблицы каталога и убедиться, что переименуемая таблица или индекс не нарушает каких-либо из этих ограничений.

Процедура:

Для того чтобы переименовать таблицу или индекс с помощью Центра управления, выполните следующие действия:

1. Разверните дерево объектов и найдите папку **Таблицы** или **Производные таблицы**.
2. Щелкните правой кнопкой мыши на имени таблицы или производной таблицы, которую нужно переименовать, и выберите во всплывающем меню пункт **Переименовать**.
3. Введите новое имя таблицы или производной таблицы и нажмите кнопку **Ok**.

Чтобы переименовать существующую таблицу из командной строки, введите команду:

```
RENAME TABLE <имя_схемы>.<имя_таблицы> TO <новое_имя>
```

Следующий оператор SQL переименовывает таблицу EMPLOYEE в схеме COMPANY в таблицу EMPL:

```
RENAME TABLE COMPANY.EMPLOYEE TO EMPL
```

Для того чтобы переименовать индекс с помощью командной строки, введите:

```
RENAME INDEX <имя-схемы>.<имя-индекса> TO <новое-имя>
```

Следующий оператор SQL изменяет имя индекса EMPIND в схеме COMPANY на MSTRIND:

```
RENAME INDEX COMPANY.EMPIND TO MSTRIND
```

Для пакетов, ссылающихся на переименованную таблицу или индекс, необходимо заново выполнить связывание. Связывание выполняется автоматически даже в том случае, если есть другой индекс с таким же именем. До появления более эффективного индекса в пакете будет применяться прежний индекс с новым именем.

Ссылки, связанные с данной темой:

- “RENAME statement” в *SQL Reference, Том 2*

Отбрасывание таблицы

Процедура:

Таблицу можно отбросить с помощью оператора SQL DROP TABLE.

При отбрасывании таблицы строка с информацией об этой таблице удаляется из каталога SYSCAT.TABLES, это также влияет на все другие объекты, зависящие от этой таблицы. Например:

- Отбрасываются все имена столбцов.
- Отбрасываются индексы, созданные для каких-либо столбцов таблицы.
- Все производные таблицы, созданные на основе этой таблицы, отмечаются как неработоспособные.
- Неявно отменяются все привилегии для отброшенной таблицы и зависимых производных таблиц.
- Отбрасываются все реляционные ограничения, в которых эта таблица является родительской или зависимой.
- Все пакеты и кэшируемые динамические операторы SQL, зависящие от отброшенной таблицы, отмечаются как недействительные и остаются в таком состоянии, пока не будут заново созданы зависимые объекты. Сюда входят пакеты, зависимые от каких-либо надтаблиц над отбрасываемой подтаблицей в ее иерархии.
- Все ссылочные столбцы, для которых отброшенная таблица определена как область видимости, становятся ссылочными столбцами “без области видимости”.
- Это не влияет на определение алиаса, так как алиас может быть неопределенным.
- Все триггеры, зависящие от отброшенной таблицы, отмечаются как неработоспособные.

- Все файлы, связанные при помощи каких-либо столбцов DATALINK, становятся несвязанными. Операция отмены связей выполняется не сразу, то есть эти файлы могут не быть немедленно доступны для других операций.

Чтобы отбросить таблицу с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой мыши по таблице, которую нужно отбросить, и выберите из всплывающего меню пункт **Отбросить**.
3. Включите переключатель **Подтверждение** и нажмите кнопку **OK**.

Чтобы отбросить таблицу из командной строки, введите команду:

```
DROP TABLE <имя_таблицы>
```

Следующий оператор отбрасывает таблицу DEPARTMENT:

```
DROP TABLE DEPARTMENT
```

Если таблица имеет подтаблицы, нельзя отбросить только ее одну. Однако все таблицы в иерархии таблиц можно отбросить с помощью одного оператора DROP TABLE HIERARCHY, как показано в следующем примере:

```
DROP TABLE HIERARCHY person
```

В операторе DROP TABLE HIERARCHY должно задаваться имя корневой таблицы отбрасываемой иерархии.

Существуют различия между отбрасыванием иерархии таблиц и отбрасыванием конкретной таблицы:

- DROP TABLE HIERARCHY не активирует триггеры удаления, которые активируются отдельными операторами DROP TABLE. Например, при отбрасывании отдельной подтаблицы активируются триггеры удаления для ее надтаблиц.
- DROP TABLE HIERARCHY не записывает в журнал записи для отдельных строк отброшенных таблиц. Вместо этого в журнал вносится информация об отбрасывании иерархии как об одном событии.

Понятия, связанные с данным:

- “Зависимости операторов при изменении объектов” на стр. 227

Задачи, связанные с данной темой:

- “Отбрасывание пользовательской временной таблицы” на стр. 218
- “Восстановление неработоспособных производных таблиц” на стр. 223

Ссылки, связанные с данной темой:

- “DROP statement” в *SQL Reference, Том 2*

Отбрасывание пользовательской временной таблицы

Для создания пользовательской временной таблицы применяется оператор DECLARE GLOBAL TEMPORARY TABLE.

Предварительные требования:

При отбрасывании такой таблицы в качестве спецификатора имени таблицы нужно задать имя схемы SESSION и эта операция должна выполняться в прикладной программе, в которой таблица была создана.

Ограничения:

Пакеты не могут зависеть от этого типа таблиц, поэтому при отбрасывании такой таблицы не возникает недействительных пакетов.

Процедура:

При отбрасывании пользовательской временной таблицы, созданной до активной единицы работы или точки сохранения, эта таблица функционально отбрасывается и прикладная программа не может к ней обращаться. Однако в ее табличном пространстве для нее еще остается зарезервированным некоторое количество пространства и это не позволяет отбросить пользовательское временное табличное пространство до принятия единицы работы или выполнения точки сохранения.

Задачи, связанные с данной темой:

- “Создание временной пользовательской таблицы” на стр. 112

Ссылки, связанные с данной темой:

- “DROP statement” в *SQL Reference, Том 2*
- “SET SCHEMA statement” в *SQL Reference, Том 2*

Отбрасывание триггера

Процедура:

Объект триггера можно отбросить с помощью оператора DROP, но при этом зависимые пакеты могут быть отмечены как недействительные, а именно:

- Если отброшен триггер update без явного списка столбцов, становятся недействительными пакеты, в которых выполняется операция обновления (update) для данной таблицы.
- Если отброшен триггер update с явным списком столбцов, недействительными становятся только пакеты, в которых выполняется операция обновления

(update) хотя бы одного столбца данной таблицы, перечисленного в списке имен-столбцов оператора CREATE TRIGGER.

- Если отброшен триггер insert, становятся недействительными пакеты, в которых выполняется операция вставки (insert) для данной таблицы.
- Если отброшен триггер delete, становятся недействительными пакеты, в которых выполняется операция удаления (delete) для данной таблицы.

Пакет остается недействительными, пока прикладная программа явно не выполнит его связывание или повторное связывание или пока этот пакет не будет запущен и менеджер баз данных не выполнит для него повторное связывание автоматически.

Задачи, связанные с данной темой:

- “Создание триггера” на стр. 125

Ссылки, связанные с данной темой:

- “DROP statement” в *SQL Reference, Том 2*

Отбрасывание пользовательской функции (UDF), отображения функции или метода

Пользовательскую функцию, шаблон функции или отображение функции можно отбросить с помощью оператора DROP.

Предварительные требования для установки:

От функции или шаблона функции могут зависеть другие объекты. Чтобы можно было отбросить функцию, необходимо сначала удалить все такие зависимости, включая отображения функций (за исключением пакетов, которые отмечаются как неработоспособные).

Ограничения:

Пользовательскую функцию нельзя отбросить, если от нее зависит производная таблица, триггер, проверочное ограничение таблицы или другая пользовательская функция. Нельзя отбросить функции, неявно созданные оператором CREATE DISTINCT TYPE. Нельзя отбросить функции из схемы SYSIBM или SYSFUN.

Процедура:

Отображение функции можно отключить с помощью опции отображения DISABLE.

Для пакетов, помеченных как неработоспособные, не выполняется неявное связывание. Для таких пакетов нужно выполнить связывание с помощью

команды BIND или REBIND, либо их нужно подготовить с помощью команды PREP. При отбрасывании пользовательской функции делаются недействительными все использующие ее пакеты или кэшируемые динамические операторы SQL.

При отбрасывании отображения функции пакет отмечается как недействительный. Повторное связывание выполняется автоматически и оптимизатор пытается использовать локальную функцию. Если локальная функция - это шаблон, неявное повторное связывание не будет выполнено.

Ссылки, связанные с данной темой:

- "DROP statement" в *SQL Reference, Том 2*
- "BIND Command" в *Command Reference*
- "PRECOMPILE Command" в *Command Reference*
- "REBIND Command" в *Command Reference*

Отбрасывание пользовательского типа (UDT) или отображения типа

С помощью оператора DROP можно отбросить пользовательский тип или отображение типов.

Ограничения:

Нельзя отбросить пользовательский тип, если он используется:

- В определении столбца существующей таблицы или производной таблицы (особые типы)
- В качестве типа существующей типизированной таблицы или типизированной производной таблицы (структурированный тип)
- В качестве надтипа другого структурированного типа

Нельзя отбросить отображение типов по умолчанию; его можно только переопределить, создав другое отображение типов.

Менеджер баз данных пытается отбросить все функции, зависящие от этого особого типа. Если такую пользовательскую функцию нельзя отбросить, нельзя отбросить и данный пользовательский тип. Пользовательскую функцию нельзя отбросить, если от нее зависит производная таблица, триггер, проверочное ограничение таблицы или другая пользовательская функция. При отбрасывании пользовательского типа становятся недействительными все использующие его пакеты или кэшируемые динамические операторы SQL.

Учтите, что можно отбросить только преобразования, определенные вами или другими разработчиками прикладных программ; встроенные преобразования и связанные с ними определения групп нельзя отбросить.

Процедура:

Для отбрасывания пользовательского типа применяется оператор DROP.

Если для пользовательского типа создано преобразование и вы собираетесь отбросить этот тип, возможно, следует отбросить это преобразование. Для этого используется оператор DROP TRANSFORM.

Понятия, связанные с данным:

- “Пользовательский тип (UDT)” на стр. 134

Задачи, связанные с данной темой:

- “Создание пользовательского особого типа” на стр. 135
- “Создание отображения типа” на стр. 137

Ссылки, связанные с данной темой:

- “DROP statement” в *SQL Reference, Том 2*

Изменение и отбрасывание производной таблицы

С помощью оператора ALTER VIEW можно изменить определение существующей производной таблицы, задав область видимости для столбца ссылочного типа. Оператор DROP позволяет удалить производную таблицу.

Предварительные требования:

При изменении производной таблицы область видимости можно задавать только для существующего столбца ссылочного типа, для которого область видимости еще не определена. Кроме того, этот столбец не должен наследоваться от производной надтаблицы.

Ограничения:

Для внесения изменений в исходное содержимое производной таблицы вам потребуются триггеры. Для внесения других изменений в производную таблицу необходимо отбросить, а затем заново создать эту таблицу.

Процедура:

Столбец, имя которого задано в операторе ALTER VIEW, должен иметь тип данных REF (тип, содержащий имя типизированной таблицы или имя типизированной производной таблицы). С помощью триггеров INSTEAD OF можно изменить содержимое производной таблицы.

Эта операция не влияет на другие объекты базы данных, такие как таблицы и индексы, но зависимые пакеты и кэшируемые динамические операторы отмечаются как недействительные.

Для изменения определения производной таблицы с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Производные таблицы**.
2. Щелкните правой кнопкой по производной таблице, которую нужно изменить, и выберите из всплывающего меню пункт **Изменить**.
3. В окне **Изменить производную таблицу** введите или измените комментарий и нажмите кнопку **OK**.

Чтобы изменить производную таблицу из командной строки, введите команду:

```
ALTER VIEW <имя_производной_таблицы> ALTER <имя_столбца>
   ADD SCOPE <имя_типизированной_таблицы_или_производной_таблицы>
```

Чтобы отбросить производную таблицу с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Производные таблицы**.
2. Щелкните правой кнопкой мыши по производной таблице, которую нужно отбросить, и выберите из всплывающего меню пункт **Отбросить**.
3. Включите переключатель **Подтверждение** и нажмите кнопку **OK**.

Чтобы отбросить производную таблицу из командной строки, введите команду:

```
DROP VIEW <имя_производной_таблицы>
```

В следующем примере показано, как отбросить производную таблицу EMP_VIEW:

```
DROP VIEW EMP_VIEW
```

Все производные таблицы, зависящие от отбрасываемой производной таблицы, станут неработоспособными.

Как и в случае иерархии таблиц, можно отбросить всю иерархию производных таблиц с помощью одного оператора, задав в нем имя корневой производной таблицы этой иерархии, как показано в следующем примере:

```
DROP VIEW HIERARCHY VPerson
```

Понятия, связанные с данным:

- “Зависимости операторов при изменении объектов” на стр. 227

Задачи, связанные с данной темой:

- “Создание триггера” на стр. 125
- “Создание производной таблицы” на стр. 137
- “Восстановление неработоспособных производных таблиц” на стр. 223

Ссылки, связанные с данной темой:

- “ALTER VIEW statement” в *SQL Reference, Том 2*
- “DROP statement” в *SQL Reference, Том 2*

Восстановление неработоспособных производных таблиц

Процедура:

Производные таблицы могут стать *непригодными к использованию*:

- В результате аннулирования привилегии для базовой таблицы.
- Если отброшена таблица, алиас или функция.
- Если стала неработоспособной базовая производная таблица. (Базовая производная таблица - это типизированная производная таблица, на основе которой создана другая типизированная производная таблица.)
- Если отброшены производные таблицы, от которых они зависят.

Следующие шаги помогают восстановить неработоспособную производную таблицу:

1. Узнайте оператор SQL, который использовался для создания этой производной таблицы. Эту информацию можно получить из столбца TEXT производной таблицы каталога SYSCAT.VIEW.
2. Заново создайте эту производную таблицу с помощью оператора CREATE VIEW с тем же именем и тем же определением производной таблицы.
3. Используйте оператор GRANT, чтобы вновь предоставить все привилегии, которые были ранее предоставлены для этой производной таблицы. (Учтите, что все привилегии, предоставленные для неработоспособной производной таблицы, отменяются.)

Если неработоспособную производную таблицу не нужно восстанавливать, можно явно отбросить ее с помощью оператора DROP VIEW или же создать новую производную таблицу с тем же именем, но с другим определением.

Для неработоспособной производной таблицы сохраняются только записи в производных таблицах каталога SYSCAT.TABLES и SYSCAT.VIEWS; все записи в производных таблицах каталога SYSCAT.VIEWDEP, SYSCAT.TABAUTH, SYSCAT.COLUMNS и SYSCAT.COLAUTH удаляются.

Задачи, связанные с данной темой:

- “Изменение и отбрасывание производной таблицы” на стр. 221

Ссылки, связанные с данной темой:

- “CREATE VIEW statement” в *SQL Reference, Том 2*
- “DROP statement” в *SQL Reference, Том 2*
- “GRANT (Table, View, or Nickname Privileges) statement” в *SQL Reference, Том 2*
- “SYSCAT.VIEWS catalog view” в *SQL Reference, Том 1*

Отбрасывание материализованной таблицы запроса и промежуточной таблицы

Процедура:

Материализованную таблицу запроса и промежуточную таблицу нельзя изменить, но можно отбросить.

При этом будут отброшены все индексы, первичные ключи, внешние ключи и проверочные ограничения для этой таблицы. Все производные таблицы и триггеры, ссылающие на эту таблицу, становятся неработоспособными. Все пакеты, зависящие от каких-либо отброшенных или отмеченных как неработоспособные объектов, становятся недействительными.

Для того чтобы отбросить материализованную таблицу запроса с помощью Центра управления, выполните следующие действия:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой мыши на имени материализованной таблицы запроса или промежуточной таблицы и выберите во всплывающем меню пункт **Отбросить**.
3. Включите переключатель **Подтверждение** и нажмите кнопку **OK**.

Для того чтобы отбросить материализованную таблицу запроса или промежуточную таблицу с помощью командной строки, введите:

```
DROP TABLE <имя_таблицы>
```

Следующий оператор SQL отбрасывает материализованную таблицу запроса XT:

```
DROP TABLE XT
```

Материализованную таблицу запроса можно отбросить явно с помощью оператора DROP TABLE, либо неявно при отбрасывании одной из базовых таблиц.

Промежуточную таблицу можно отбросить явно с помощью оператора DROP TABLE, либо неявно при отбрасывании связанной материализованной таблицы запроса.

Понятия, связанные с данным:

- “Зависимости операторов при изменении объектов” на стр. 227

Задачи, связанные с данной темой:

- “Создание материализованной таблицы запроса” на стр. 141
- “Создание промежуточной таблицы” на стр. 146

Ссылки, связанные с данной темой:

- “DROP statement” в *SQL Reference, Том 2*

Восстановление неработоспособных сводных таблиц

Процедура:

Сводные таблицы могут стать *неработоспособными* в результате отзыва привилегии SELECT для базовой таблицы.

Следующие шаги помогают восстановить неработоспособную сводную таблицу:

- Узнайте оператор SQL, который использовался для создания этой сводной таблицы. Эту информацию можно получить из столбца TEXT производной таблицы каталога SYSCAT.VIEW.
- Заново создайте эту сводную таблицу с помощью оператора CREATE SUMMARY TABLE с тем же именем и тем же определением сводной таблицы.
- Используйте оператор GRANT, чтобы вновь предоставить все привилегии, которые были ранее предоставлены для этой сводной таблицы. (Учтите, что все привилегии, предоставленные для неработоспособной сводной таблицы, отменяются.)

Если неработоспособную сводную таблицу не нужно восстанавливать, ее можно явно отбросить с помощью оператора DROP TABLE или можно создать новую сводную таблицу с тем же именем, но с другим определением.

Для неработоспособной сводной таблицы сохраняются только записи в производных таблицах каталога SYSCAT.TABLES и SYSCAT.VIEWS; все записи в производных таблицах каталога SYSCAT.VIEWDEP, SYSCAT.TABAUTH, SYSCAT.COLUMNS и SYSCAT.COLAUTH удаляются.

Ссылки, связанные с данной темой:

- “CREATE TABLE statement” в *SQL Reference, Том 2*
- “DROP statement” в *SQL Reference, Том 2*
- “GRANT (Table, View, or Nickname Privileges) statement” в *SQL Reference, Том 2*
- “SYSCAT.VIEWS catalog view” в *SQL Reference, Том 1*

Отбрасывание индекса, расширения индекса и спецификации индекса

Ограничения:

Ни одно из условий в определении индекса, расширении индекса или спецификации индекса нельзя изменить; необходимо отбросить индекс или расширение индекса и создать их заново. (Отбрасывание индекса или спецификации индекса не вызывает отбрасывания каких-либо других объектов, но некоторые пакеты могут стать недействительными.)

Задаваемое имя расширения индекса должно указывать расширение индекса, описанное в каталоге.Условие RESTRICT задает правило, согласно которому не должны существовать индексы, зависящие от определения этого расширения индекса. Если от этого расширения индекса зависит базовый индекс, операция отбрасывания не будет выполнена.

Нельзя явно отбросить индекс первичного ключа или ключа уникальности (за исключением спецификации индекса). Для его отбрасывания нужно использовать один из следующих методов:

- Если первичный индекс или ограничение уникальности были автоматически созданы для первичного ключа или ключа уникальности, при отбрасывании этого первичного ключа или ключа уникальности будет отброшен и этот индекс. Для этого используется оператор ALTER TABLE.
- Если первичный индекс или ограничение уникальности были определены пользователем, необходимо сначала отбросить первичный ключ или ключ уникальности с помощью оператора ALTER TABLE. После отбрасывания первичного ключа или уникального ключа этот индекс уже не будет считаться первичным индексом или индексом уникальности и его можно будет отбросить явно.

Процедура:

Чтобы отбросить индекс, расширение индекса или спецификацию индекса с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Индексы**.
2. Щелкните правой кнопкой мыши по индексу, который нужно отбросить, и выберите из всплывающего меню пункт **Отбросить**.
3. Включите переключатель **Подтверждение** и нажмите кнопку **OK**.

Чтобы отбросить индекс, расширение индекса или спецификацию индекса из командной строки, введите команду:

```
DROP INDEX <имя_индекса>
```

Следующий оператор SQL отбрасывает индекс с именем PH:

```
DROP INDEX PH
```

Следующий оператор SQL отбрасывает расширение индекса с именем IX_MAP:

```
DROP INDEX EXTENSION ix_map RESTRICT
```

Все пакеты и кэшируемые динамические операторы SQL, зависящие от отброшенных индексов, отмечаются как недействительные. Изменения, вызванные добавлением или отбрасыванием индексов, не влияют на прикладные программы.

Понятия, связанные с данным:

- “Зависимости операторов при изменении объектов” на стр. 227

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в *SQL Reference, Том 2*
- “DROP statement” в *SQL Reference, Том 2*

Зависимости операторов при изменении объектов

Зависимости операторов включают в себя зависимости пакетов и кэшируемых динамических операторов SQL. *Пакет* - это объект базы данных, содержащий информацию, необходимую менеджеру баз данных для обеспечения наиболее эффективного доступа к данным для конкретной прикладной программы. *Связывание* - это процесс, создающий пакет, необходимый менеджеру баз данных для обращения к базе данных при выполнении прикладной программы.

Пакеты и кэшируемые динамические операторы SQL могут зависеть от многих типов объектов.

Ссылки на эти объекты могут быть явными (пример: таблица или пользовательская функция, заданные в операторе SQL SELECT). Они могут также быть неявными (пример: зависимая таблица, которую нужно проверить, чтобы убедиться, что при удалении строки из родительской таблицы не нарушается реляционное ограничение). Пакеты также зависят от привилегий, предоставленных создателю пакета.

Если пакет или кэшируемый динамический оператор SQL зависит от некоторого объекта, и этот объект был отброшен, то этот пакет или кэшируемый динамический оператор SQL становится “недействительным”. Если пакет зависит от пользовательской функции, и эта функция была отброшена, то пакет становится “неработоспособным”.

Кэшируемый динамический оператор SQL, находящийся в состоянии ″недействительный″, автоматически заново оптимизируется при следующем

использовании. Если объект, необходимый этому оператору, был отброшен, выполнение этого динамического оператора может быть неудачным и будет выдано сообщение об ошибке.

Для пакета, находящегося в состоянии ″недействительный″, повторное связывание выполняется неявно при его следующем использовании. Для такого пакета можно также явно выполнить повторное связывание. Если пакет отмечен как недействительный из-за отбрасывания триггера, повторно связанный пакет более не будет запускать этот триггер.

Для пакета, находящегося в состоянии ″неработоспособный″, необходимо явно выполнить повторное связывание, прежде чем его можно будет использовать.

Объекты базы данных объединения имеют аналогичные зависимости. Например, при отбрасывании сервера становятся недействительными все пакеты или кэшируемые динамические операторы SQL, в которых используются псевдонимы, связанные с этим сервером.

В некоторых случаях повторное связывание пакета невозможно. Например, если таблица была отброшена и не создана заново, то нельзя повторно выполнить связывание пакета. В этом случае нужно или заново создать отброшенный объект или изменить прикладную программу, чтобы она не использовала этот объект.

В многих других случаях (например, если было отброшено одно из ограничений) повторное связывание пакета возможно.

Следующие производные таблицы каталога помогут вам определить состояние пакета и его зависимости:

- SYSCAT.PACKAGEAUTH
- SYSCAT.PACKAGEDEP
- SYSCAT.PACKAGES

Понятия, связанные с данным:

- “Package Creation Using the BIND Command” в *Application Development Guide: Programming Client Applications*
- “Application, Bind File, and Package Relationships” в *Application Development Guide: Programming Client Applications*
- “Package Rebinding” в *Application Development Guide: Programming Client Applications*

Ссылки, связанные с данной темой:

- “DROP statement” в *SQL Reference, Том 2*
- “SYSCAT.PACKAGEAUTH catalog view” в *SQL Reference, Том 1*

- “SYSCAT.PACKAGEDEP catalog view” в *SQL Reference, Том 1*
- “SYSCAT.PACKAGES catalog view” в *SQL Reference, Том 1*
- “BIND Command” в *Command Reference*
- “REBIND Command” в *Command Reference*

Часть 2. Защита баз данных

Глава 4. Управление доступом к базам данных

Безопасность баз данных - важнейшая обязанность администратора баз данных и системного администратора. Обеспечение безопасности базы данных включает несколько аспектов:

- Предотвращение случайных потерь данных и нарушений целостности данных из-за отказов оборудования и системы.
- Предотвращение несанкционированного доступа к ценным данным. Вы должны обеспечить закрытость конфиденциальной информации от тех, кому она не требуется для работы.
- Предотвращение ущерба от действий неуполномоченных лиц - уничтожения данных и искажения данных.
- Мониторинг обращений пользователей к данным, который обсуждается в разделе Глава 5, “Аудит действий DB2” на стр. 283.

Далее описаны следующие темы:

Планирование мер безопасности: Определите сначала цели для плана управления доступом к базам данных и решите, кто, к каким объектам и при каких обстоятельствах будет иметь доступ. В вашем плане также должно быть указано, какие для достижения этих целей будут использованы функции баз данных, функции других программ, административные процедуры.

Выбор ID пользователей и групп для установки

Соображения безопасности должны учитываться администратором DB2® с момента установки продукта.

Для установки DB2 потребуются имя пользователя, имя группы и пароль. Во время установки все требуемые параметры администратора имеют значения по умолчанию. Когда же установка DB2 с параметрами по умолчанию завершена, администратору настоятельно рекомендуется создать новые имена пользователей, имена групп и пароли, прежде чем создавать экземпляры, в которых будут размещены базы данных. Использование новых имен пользователей, имен группы и паролей уменьшит риск того, что другой пользователь, зная значения по умолчанию, использует их для несанкционированного доступа к экземплярам и базам данных.

Пароли очень важны для аутентификации пользователей. Если на уровне операционной системы не задается никаких аутентификационных требований, а база данных использует для аутентификации пользователей средства операционной системы, соединение будет разрешено всем пользователям. Например, в операционной системе UNIX® незаданные пароли рассматриваются как NULL. Любой пользователь с неуказанным паролем будет рассматриваться как пользователь с паролем NULL. С точки зрения операционной системы эти пароли совпадает, пользователь считается прошедшим проверку и допускается к соединению с базой данных. Если вы хотите, чтобы аутентификацию пользователей для базы данных выполняла операционная система, пароли следует задавать на уровне этой операционной системы.

Еще одна рекомендация, связанная с мерами безопасности после установки DB2 - изменить привилегии, предоставленные пользователям по умолчанию. В ходе установки привилегии SYSADM предоставляются по умолчанию следующим пользователям в каждой из операционных систем:

Windows® 9x	Всем пользователям Windows 98 или Windows ME.
Windows NT®	В Windows NT, Windows 2000, Windows XP и Windows .NET - пользователю DB2, входящему в группу Администраторы.
UNIX	Всем пользователям DB2 из первичной группы ID пользователя владельца экземпляра.

Привилегии SYSADM - самый сильный набор привилегий, доступный в DB2. (Привилегии рассматриваются ниже в этой главе). Поэтому вы, возможно, не захотите, чтобы все эти пользователи имели привилегии SYSADM по умолчанию. DB2 позволяет администратору предоставлять привилегии группам и отдельным пользователям и отзывать их.

Изменяя параметр конфигурации *sysadm_group* менеджера баз данных, администратор может управлять тем, какая группа будет определена как группа управления системой с привилегиями системного администратора.

Требованиями безопасности диктуются следующие меры при установке DB2 и при последующем создании экземпляров и баз данных.

Группа, определенная как группа управления системой (путем изменения параметра *sysadm_group*) должна существовать в системе. Желательно, чтобы имя этой группы ясно свидетельствовало о принадлежности группы владельцам экземпляра. ID пользователей и группы, принадлежащие этой группе, имеют полномочия системного администратора для своих экземпляров.

Рекомендуется создать ID пользователя - владельца экземпляра, который будет легко ассоциироваться с конкретным экземпляром. Этот ID пользователя должен входить в группу SYSADM, описанную выше. Рекомендуется также использовать этот ID пользователя - владельца экземпляра только как член группы владельца экземпляра и не использовать его в других группах. Это поможет предотвратить неконтролируемый рост числа ID пользователей и групп, способных изменять среду экземпляра.

Создаваемый ID пользователя необходимо связать с паролем, чтобы проводить аутентификацию перед каждым входом в данные и базы данных в пределах экземпляра. При создании пароля рекомендуется следовать указаниям по заданию паролей, принятым в вашей организации.

Понятия, связанные с данным:

- “Правила присвоения имен в среде NLS” на стр. 327
- “Правила присвоения имен в среде Unicode” на стр. 329
- “Особенности защиты при работе с пользователями Windows NT” на стр. 235
- “Особенности защиты при работе с пользователями UNIX” на стр. 236
- “Аутентификация” в *Руководство администратора: Планирование*
- “Авторизация” в *Руководство администратора: Планирование*
- “Общие правила именования объектов и пользователей” на стр. 237
- “Правила именования” на стр. 321
- “Правила именования пользователей, групп и ID пользователей” на стр. 324

Сведения о защите для различных операционных систем

В различных операционных системах применяются различные подходы к вопросам безопасности. В этом разделе рассмотрены некоторые аспекты безопасности, связанные с операционными системами.

Особенности защиты при работе с пользователями Windows NT

Полномочия системного администратора (SYSADM) предоставляются всем учетным записям пользователей DB2®, которые входят в группу Администраторы на том компьютере, на котором определена учетная запись.

По умолчанию в среде домена Windows® права доступа SYSADM к экземпляру предоставляются только тем пользователям домена, которые входят в группу Администраторы на контроллере домена. Поскольку DB2 всегда производит авторизацию на том компьютере, где была определена учетная запись, добавление пользователя домена в локальную группу администраторов на сервере не дает этой группе прав SYSADM пользователя домена.

Примечание: В среде домена, например, в Windows NT, DB2 идентифицирует только первые 64 группы, которые соответствуют требованиям и ограничениям, и в которые входит ID пользователя. Может быть создано более 64 групп.

Чтобы избежать добавления пользователя домена в группу администраторов в PDC, нужно создать глобальную группу и добавить туда пользователей (и домена, и локальных), которым вы хотите предоставить права SYSADM. Для этого нужно ввести следующие команды:

```
DB2STOP
DB2 UPDATE DBM CFG USING SYSADM_GROUP global_group
DB2START
```

Понятия, связанные с данным:

- “Особенности защиты при работе с пользователями UNIX” на стр. 236

Особенности защиты при работе с пользователями UNIX

В DB2® Universal Database пользователю root не разрешено напрямую работать в качестве администратора базы данных. Для получения полномочий администратора базы данных необходимо вызвать команды **su - <владелец экземпляра>**.

На платформах на основе UNIX нужно создать группу для изолированных пользовательских функций и хранимых процедур, и все ID пользователей, использующие изолированные пользовательские функции или хранимые процедуры, должны быть членами этой группы. Как и у группы SYSADM, у группы изолированных пользовательских функций и хранимых процедур должно быть содержательное имя. ID пользователей, вошедшие в группу изолированных пользовательских функций и хранимых процедур, получат по умолчанию все связанные с ней права и полномочия.

Из соображений безопасности не рекомендуется в качестве ID изолированных функций использовать имя экземпляра. Однако если использовать изолированные пользовательские функции и хранимые процедуры не предполагается, можно не создавать новый ID пользователя и задать в качестве ID изолированных функций имя экземпляра.

В общем случае рекомендуется создать ID пользователя, ясно связанный с группой. Имя пользователя для изолированных пользовательских функций и хранимых процедур задается в качестве параметра сценария создания экземпляра (**db2icrt ... -u <ID для изолированных функций>**). Этот параметр не требуется при установке клиентов DB2 и комплекта разработчика программ DB2.

Понятия, связанные с данным:

- “Особенности защиты при работе с пользователями Windows NT” на стр. 235

Общие правила именования объектов и пользователей

Есть правила для именования всех объектов и пользователей. Часть правил относится только к некоторым платформам. Например, определенные правила регулируют использование в имени букв верхнего и нижнего регистров.

- На платформах UNIX® имена должны быть заданы строчными буквами.
- На платформах Windows® имена могут содержать как прописные, так и строчные буквы.

В UNIX команда **db2icrt** создает основной каталог библиотек SQL (`sqllib`) в домашнем каталоге владельца экземпляра.

В Windows каталог экземпляра расположен в подкаталоге `/sqllib` того каталога, в котором установлен продукт DB2®.

Понятия, связанные с данным:

- “Правила именования” на стр. 321

Способы аутентификации, поддерживаемые сервером

Чтобы получить доступ к экземпляру или базе данных, пользователь сначала должен пройти *аутентификацию*. *Тип аутентификации* экземпляра определяет, каким образом и где происходит проверка пользователя. Тип аутентификации сохраняется в файле конфигурации менеджера баз данных на сервере. Первоначально он задается при создании экземпляра. Тип аутентификации распространяется на весь экземпляр, определяя доступ к серверу баз данных и всем базам данных, которыми он управляет.

Если предполагается доступ к источникам данных из базы данных объединения, нужно предусмотреть проведение аутентификации для источников данных и определение типов для аутентификации объединения.

Поддерживаются следующие типы аутентификации:

SERVER

Задает, что аутентификация происходит на сервере при помощи средств защиты локальной операционной системы. Если ID пользователя и пароль задаются при попытке соединения или подключения, прежде чем разрешить этому пользователю доступ к экземпляру, они сравниваются с допустимыми сочетаниями ID пользователя и пароля на сервере. Этот механизм защиты принимается по умолчанию.

Примечание: Программа сервера определяет, является соединение локальным или удаленным. При локальных соединениях для успешной аутентификации типа SERVER не требуется ID пользователя и пароля.

SERVER_ENCRYPT

Задает, что сервер принимает зашифрованные схемы аутентификации типа SERVER. Если не задана аутентификация клиента, клиент проходит аутентификацию по методу, выбранному на сервере.

Если способ аутентификации клиента равен SERVER, то клиент проходит аутентификацию, передавая серверу ID и пароль пользователя. Если способ аутентификации клиента равен SERVER_ENCRYPT, то клиент проходит аутентификацию, передавая ID пользователя и пароль в зашифрованном виде.

Если у клиента задано SERVER_ENCRYPT, а на сервере - SERVER, будет возвращена ошибка из-за несоответствия уровней аутентификации.

CLIENT

Задает, что аутентификация происходит на разделе базы данных при вызове программы с использованием средств защиты операционной системы. Прежде чем разрешить ID пользователя доступ к экземпляру, этот ID пользователя и пароль, заданные при попытке соединения или подключения, сравниваются с допустимыми сочетаниями ID пользователя и пароля на узле клиента. Никакой дальнейшей аутентификации на сервере баз данных не производится. Иногда такая процедура называется единым входом в систему.

Если пользователь зарегистрировался локально или как клиент, он известен только данной рабочей станции клиента.

Если на удаленном экземпляре тип аутентификации - CLIENT, окончательный тип аутентификации зависит от еще двух параметров: *trust_allclnts* и *trust_clntauth*.

Уровень мер безопасности CLIENT только для ДОВЕРЕННЫХ клиентов:

Доверенные клиенты - это клиенты, имеющие надежные локальные системы мер безопасности. Доверенными считаются все клиенты, кроме клиентов с операционной системой Windows® 9x.

Если выбран тип аутентификации CLIENT, могут быть включены дополнительные опции, защищающие от клиентов, у которых операционная среда не имеет встроенных мер безопасности.

Чтобы защититься от ненадежных клиентов, администратор может включить режим аутентификации доверенных клиентов, задав для параметра *trust_allclnts* значение NO. Тогда все надежные платформы смогут проводить аутентификацию пользователя для сервера. Ненадежные клиенты проходят аутентификацию на сервере и должны предоставлять ID пользователя и пароль. Параметр конфигурации *trust_allclnts* позволяет задать, доверяете ли вы клиентам. По умолчанию значение этого параметра - YES.

Примечание: Можно задать доверие всем клиентам (*trust_allclnts* имеет значение YES), но при этом отметить некоторых клиентов, как не имеющих собственной надежной системы мер защиты при аутентификации.

Кроме того, вы, возможно, захотите аутентификацию даже доверенных клиентов выполнять на сервере. Чтобы указать, где будут проверяться доверенные клиенты, используйте параметр конфигурации *trust_clntauth*. По умолчанию значение этого параметра - CLIENT.

Примечание: Только для доверенных клиентов, если при попытке выполнить операторы соединения CONNECT или подключения ATTACH не заданы явно ID пользователя и пароль, проверка пользователя производится на клиенте. Параметр *trust_clntauth* влияет только на место проверки информации в операторах с условиями USER и USING.

Для того чтобы запретить свободный доступ всем клиентам, кроме клиентов DRDA® из систем DB2® для OS/390® и z/OS, DB2 для VM и VSE и DB2 для iSeries, присвойте параметру *trust_allclnts* значение DRDAONLY. Только этим клиентам будет разрешено выполнять аутентификацию пользователей. Все остальные клиенты должны проходить аутентификацию на сервере, предоставляя ID пользователя и пароль.

Параметр *trust_clntauth* определяет, где должны проходить аутентификацию перечисленные выше клиенты: если *trust_clntauth* имеет значение client, аутентификация производится на клиенте. Если

trust_clntauth имеет значение server, аутентификация производится на клиенте, если пароль не предоставлен, и на сервере, если пароль предоставлен.

Таблица 5. Режимы аутентификации при использовании сочетаний параметров TRUST_ALLCLNTS и TRUST_CLNTAUTH.

TRUST_ ALLCLNTS	TRUST_ CLNTAUTH	Аутент. ненадежных не–DRDA клиентов без пароля	Аутент. ненадежных не–DRDA клиентов с паролем	Аутент. доверенных не–DRDA клиентов без пароля	Аутент. доверенных не–DRDA клиентов с паролем	Аутент. клиентов DRDA без пароля	Аутент. клиентов DRDA с паролем
Да	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT
Да	SERVER	CLIENT	SERVER	CLIENT	SERVER	CLIENT	SERVER
Нет	CLIENT	SERVER	SERVER	CLIENT	CLIENT	CLIENT	CLIENT
Нет	SERVER	SERVER	SERVER	CLIENT	SERVER	CLIENT	SERVER
DRDAONLY	CLIENT	SERVER	SERVER	SERVER	SERVER	CLIENT	CLIENT
DRDAONLY	SERVER	SERVER	SERVER	SERVER	SERVER	CLIENT	SERVER

KERBEROS

Используется, когда и клиент, и сервер DB2 работают в операционных системах, где поддерживается протокол защиты Kerberos. Протокол защиты Kerberos выполняет аутентификацию как дополнительная служба аутентификации, используя обычное шифрование для создания общего секретного ключа. Этот ключ становится паролем пользователя и используется для проверки личности пользователя во всех случаях, когда требуются локальные или сетевые службы. Этот ключ устраняет необходимость передавать имя пользователя и пароль по сети как открытый текст. Протокол защиты Kerberos позволяет своими средствами регистрироваться на удаленном сервере DB2.

Процедура аутентификации Kerberos в операционной системе Windows выглядит следующим образом:

1. Центр рассылки ключей (KDC), расположенный на контроллере домена, выполняет аутентификацию пользователя, вошедшего в систему компьютера-клиента с помощью учетной записи домена. После этого Центр рассылки ключей выдает клиенту начальный паспорт (TGT).
2. На первом этапе установления соединения сервер отправляет клиенту имя целевого субъекта, совпадающее с именем учетной записи службы сервера DB2. Используя имя целевого субъекта и начальный паспорт, клиент запрашивает служебный паспорт у службы выдачи паспортов (TGS), расположенной на контроллере домена. Если начальный паспорт клиента и имя целевого субъекта действительны, то TGS выдает клиенту служебный паспорт.

3. Клиент отправляет свой служебный паспорт серверу по каналу связи (например, по соединению TCP/IP).
4. Сервер проверяет служебный паспорт клиента. Если он действителен, то процедура аутентификации завершается.

На компьютере клиента можно зарегистрировать базы данных в каталоге, явно указав тип аутентификации Kerberos и имя целевого субъекта сервера. Это даст возможность пропустить первый этап настройки соединения.

Если заданы ID пользователя и пароль, клиент запросит начальный паспорт для учетный записи пользователя и передаст его для аутентификации.

KRB_SERVER_ENCRYPT

Задает, что сервер принимает аутентификацию KERBEROS или зашифрованные схемы аутентификации типа SERVER. Если аутентификация клиента - KERBEROS, клиент проходит аутентификацию с помощью служб защиты Kerberos. Если способ аутентификации клиента отличен от KERBEROS, либо служба аутентификации Kerberos недоступна, то тип аутентификации в системе будет эквивалентен SERVER_ENCRYPT.

Примечание: Типы аутентификации Kerberos поддерживаются только на клиентах и серверах, работающих в среде Windows 2000, Windows XP или Windows .NET. Кроме того, компьютеры клиента и сервера должны входить в один домен или надежные домены. Такой тип аутентификации следует применять в том случае, если сервер поддерживает Kerberos, и некоторые, но не все компьютеры клиентов поддерживают аутентификацию Kerberos.

Примечания:

1. Выбираемый вами тип аутентификации имеет значение лишь в тех случаях, когда к базе данных обращаются удаленные клиенты или используются возможности базы данных объединения. Большинство пользователей, обращающихся к базе данных через локальных клиентов, проходят аутентификацию на том компьютере, на которой находится база данных. Исключение может составлять случай, когда применяется Kerberos.
2. Избегайте опасности случайно заблокировать экземпляр при изменении информации аутентификации, поскольку доступ к файлу конфигурации защищен информацией, содержащейся в самом файле конфигурации. Доступом к экземпляру управляют следующие параметры файла конфигурации менеджера баз данных:
   - AUTHENTICATION *
   - SYSADM_GROUP *

- TRUST_ALLCLNTS
- TRUST_CLNTAUTH
- SYSCTRL_GROUP
- SYSMAINT_GROUP

* отмечает два самых важных параметра, чаще всего порождающих проблемы.

Есть несколько мер предосторожности. Если вы случайно заблокировали систему DB2, на всех платформах доступна опция защиты от сбоев, позволяющая отменить обычные проверки защиты DB2 и изменить файл конфигурации менеджера баз данных при помощи обладающего высокими привилегиями пользователя локальной системы защиты. Этот пользователь *всегда* имеет полномочия изменять файл конфигурации менеджера баз данных и, следовательно, может исправить ошибки.Однако этот обход защиты позволяет изменять файл конфигурации менеджера баз данных только локально. Нельзя использовать этот прием удаленно и для других команд DB2. Идентификатор этого пользователя:

- на платформах UNIX®: владелец экземпляра
- на платформах NT: один из членов локальной группы “администраторов”
- на других платформах нет локальной системы защиты, поэтому все пользователи все равно должны проходить проверку локальной защиты

Понятия, связанные с данным:

- “Особенности аутентификации удаленных клиентов” на стр. 242
- “Особенности аутентификации в многораздельной базе данных” на стр. 243
- “DB2 for Windows NT и функции защиты Windows NT - Введение” на стр. 393

Ссылки, связанные с данной темой:

- “Параметр конфигурации Тип аутентификации - authentication” в *Руководство администратора: Производительность*
- “Параметр конфигурации Доверять всем клиентам - trust_allclnts” в *Руководство администратора: Производительность*
- “Параметр конфигурации Аутентификация доверенных клиентов - trust_clntauth” в *Руководство администратора: Производительность*

Особенности аутентификации удаленных клиентов

При регистрации базы данных в каталоге для удаленного доступа тип аутентификации может быть задан в записи каталога баз данных.

Если для доступа к базе данных применяется оператор DB2® Connect, и значение не задано, то применяется аутентификация SERVER.

Тип аутентификации указывать не обязательно. Если он не задан, то на клиенте по умолчанию будет применяться значение SERVER_ENCRYPT. Однако если сервер не поддерживает тип аутентификации SERVER_ENCRYPT, клиент повторит операцию, используя значение, поддерживаемое сервером. Если сервер поддерживает несколько типов аутентификации, клиент вернет сообщение об ошибке. Это сделано для того, чтобы не был выбран неверный тип аутентификации. В этом случае клиент должен зарегистрировать базу данных в каталоге, указав поддерживаемый тип аутентификации. Если тип аутентификации задан, и совпадает со значением, указанным на сервере, то аутентификация выполняется немедленно. Если значения не совпадают, то DB2 попытается выполнить восстановление. Это может привести к выполнению дополнительных действий для устранения расхождений или к возникновению ошибки, если восстановление выполнить не удастся. В случае несовпадения правильным считается значение на сервере.

Понятия, связанные с данным:

- “Способы аутентификации, поддерживаемые сервером” на стр. 237

Особенности аутентификации в многораздельной базе данных

В распределенной базе данных во всех разделах должен быть определен один и тот же набор пользователей и групп. Если определения не будут совпадать, пользователь может после авторизации получить разные права в разных разделах. Рекомендуется обеспечить согласованность всех разделов.

Понятия, связанные с данным:

- “Способы аутентификации, поддерживаемые сервером” на стр. 237

Привилегии, полномочия и авторизация

Привилегии позволяют пользователям создавать ресурсы баз данных и обращаться к ним. *Уровни полномочий* предоставляют способ объединения привилегий и высокоуровневых операций обслуживания и утилит менеджера баз данных. Все это вместе позволяет управлять доступом к менеджеру баз данных и его объектам баз данных. Пользователи могут обращаться только к тем объектам, для которых у них есть соответствующая *авторизация*, то есть нужные привилегии или полномочия.

На рис. 3 на стр. 244 показана связь между полномочиями и их областью действия (база данных, менеджер баз данных).

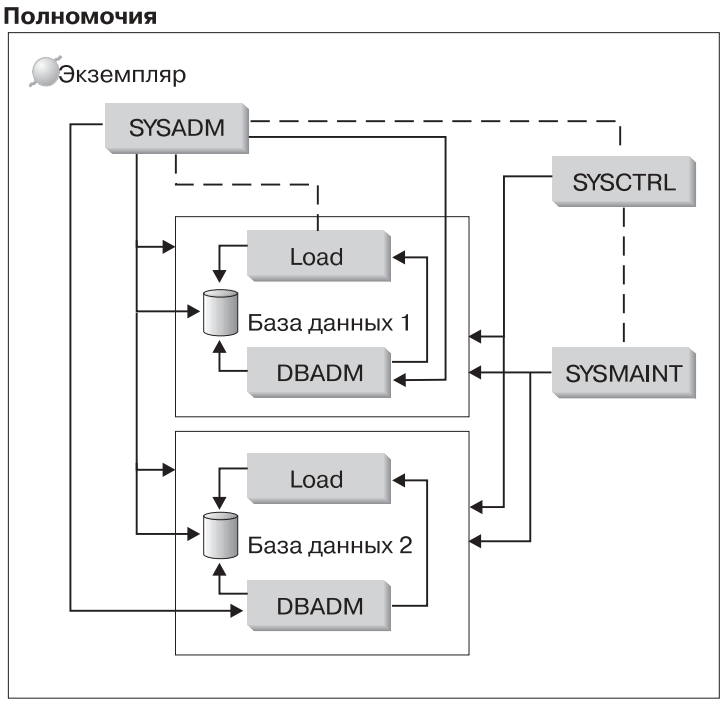


Рисунок 3. Иерархия полномочий

У пользователя или группы могут быть один или несколько из числа следующих уровней авторизации:

- Полномочия управления (SYSADM и DBADM) дают полные наборы привилегии для набора объектов.
- Системные полномочия (SYSCTRL и SYSMAINT) дают полные наборы привилегии для управления системой, но не позволяют обращаться к данным.
- Полномочия LOAD (загрузка) дают привилегии использования утилиты LOAD и утилиты AutoLoader, чтобы загружать данные в таблицы.
- Привилегия владельца (в некоторых случаях называемая также привилегией CONTROL) дает полные привилегии для конкретного объекта.
- Отдельные привилегии можно предоставлять, чтобы разрешить пользователю выполнять конкретные операции на конкретных объектах.
- Неявные привилегии могут предоставляться пользователю с привилегией выполнения пакета. Пока пользователи могут выполнять прикладную программу, им не обязательно требуются явные привилегии для объектов данных, используемых в пакете.

Пользователи с полномочиями управления (SYSADM и DBADM) и привилегиями владельца (CONTROL) могут предоставлять привилегии другим пользователям и отзывать их с помощью операторов GRANT и REVOKE

соответственно. Можно также предоставить другому пользователю привилегию для таблицы, производной таблицы или схемы, если это привилегия с опцией WITH GRANT OPTION. Однако опция WITH GRANT OPTION не позволяет отозвать уже предоставленную привилегию. Чтобы отозвать привилегию, нужно иметь полномочия SYSADM, DBADM или привилегию CONTROL.

Пользователя или группу можно наделить любым сочетанием отдельных привилегий и полномочий.При связывании привилегии с ресурсом этот ресурс должен существовать. Это значит, что нельзя предоставить пользователю привилегию SELECT для таблицы, если эта таблица еще не создана.

Примечание: Следует проявлять осторожность при предоставлении полномочий и привилегий имени авторизации, если пользователя с этим именем авторизации еще нет. Если позже будет создан пользователь с этим именем авторизации, он автоматически получит все полномочия и привилегии, связанные с этим именем авторизации.

Понятия, связанные с данным:

- “Полномочия системного администратора (SYSADM)” на стр. 246
- “Полномочия управления системой (SYSCTRL)” на стр. 247
- “Полномочия обслуживания системы (SYSMAINT)” на стр. 248
- “Полномочия управления базой данных (DBADM)” на стр. 248
- “Полномочия LOAD” на стр. 249
- “Привилегии базы данных” на стр. 250
- “Привилегии схем” на стр. 252
- “Привилегии табличных пространств” на стр. 254
- “Привилегии таблиц и производных таблиц” на стр. 254
- “Привилегии пакетов” на стр. 257
- “Привилегии индексов” на стр. 258
- “Привилегии последовательностей” на стр. 258
- “Управление доступом к объектам базы данных” на стр. 259
- “Неявно запрашиваемые привилегии для пакета” на стр. 265
- “Привилегии процедур, функций и методов” на стр. 259

Сведения о привилегиях, полномочиях и авторизации

В этом разделе описаны различные полномочия и связанные с ними права доступа.

Полномочия системного администратора (SYSADM)

Полномочия SYSADM - самый высокий уровень полномочий управления. Пользователи с полномочиями SYSADM могут запускать утилиты, выдавать команды базы данных и менеджера баз данных и обращаться к данным в любой таблице в составе любой базы данных в экземпляре менеджера баз данных. Эти полномочия позволяют управлять всеми объектами баз данных в экземпляре, включая базы данных, таблицы, производные таблицы, индексы, пакеты, схемы, серверы, алиасы, типы данных, функции, процедуры, триггеры, табличные пространства, группы разделов базы данных, пулы буферов и мониторы событий.

Полномочия SYSADM предоставляются группе, заданной в параметре конфигурации *sysadm_group*. Членство в этой группе управляется вне менеджера баз данных через утилиту защиты, используемую на вашей платформе.

Только пользователь с полномочиями SYSADM может выполнять следующие функции:

- Перенастроить базу данных
- Изменить файл конфигурации менеджера баз данных (включая задание групп с полномочиями SYSCTRL и SYSMAINT)
- Предоставить полномочия DBADM.

Примечание: Когда пользователи с полномочиями SYSADM создают базы данных, они автоматически получают явные полномочия DBADM для этих баз данных. Если создатель базы данных будет удален из группы SYSADM и вы также хотите предотвратить его доступ к этой базе данных с использованием полномочий DBADM, нужно явно отозвать эти полномочия DBADM.

Понятия, связанные с данным:

- “Полномочия управления системой (SYSCTRL)” на стр. 247
- “Полномочия обслуживания системы (SYSMAINT)” на стр. 248
- “Полномочия управления базой данных (DBADM)” на стр. 248
- “Полномочия LOAD” на стр. 249

Ссылки, связанные с данной темой:

- “Параметр конфигурации Имя группы с правами администратора системы - sysadm_group” в *Руководство администратора: Производительность*
- “Параметр конфигурации Имя группы с правами управления системой - sysctrl_group” в *Руководство администратора: Производительность*
- “Параметр конфигурации Имя группы с правами обслуживания системы - sysmaint_group” в *Руководство администратора: Производительность*

Полномочия управления системой (SYSCTRL)

Полномочия SYSCTRL - самый высокий уровень полномочий управления системой. Эти полномочия позволяют выполнять обслуживание и операции с утилитами для экземпляра менеджера баз данных и его баз данных. Эти операции могут затрагивать системные ресурсы, но не дают прямого доступа к данным в базах. Полномочия управления системой предназначены для пользователей, управляющих экземпляром менеджера баз данных, который содержит конфиденциальные данные.

Полномочия SYSCTRL предоставляются группе, заданной в параметре конфигурации *sysctrl_group*. Если эта группа задана, членство в этой группе управляется вне менеджера баз данных через утилиту защиты, используемую на вашей платформе.

Только пользователь с полномочиями SYSCTRL и выше может:

- Изменять каталог баз данных, узла и DCS (distributed connection services - служба распределенных соединений)
- Принудительно отключать пользователей от системы
- Создавать и отбрасывать базы данных
- Отбрасывать, создавать и изменять табличные пространства
- Выполнять восстановление в новую базу данных.

Кроме того, пользователь с полномочиями SYSCTRL может выполнять функции пользователей с полномочиями обслуживания системы (SYSMAINT).

Пользователи с полномочиями SYSCTRL имеют также неявную привилегию соединяться с базой данных.

Примечание: Когда пользователи с полномочиями SYSCTRL создают базы данных, они автоматически получают явные полномочия DBADM для этих баз данных. Если создатель базы данных будет удален из группы SYSCTRL и вы также хотите предотвратить его доступ к этой базе данных с использованием полномочий DBADM, нужно явно отозвать эти полномочия DBADM.

Понятия, связанные с данным:

- “Полномочия системного администратора (SYSADM)” на стр. 246
- “Полномочия обслуживания системы (SYSMAINT)” на стр. 248

Ссылки, связанные с данной темой:

- “Параметр конфигурации Имя группы с правами управления системой - sysctrl_group” в *Руководство администратора: Производительность*

Полномочия обслуживания системы (SYSMAINT)

Полномочия SYSMAINT - второй уровень полномочий управления системой. Эти полномочия позволяют выполнять обслуживание и операции с утилитами для экземпляра менеджера баз данных и его баз данных. Эти операции могут затрагивать системные ресурсы, но не дают прямого доступа к данным в базах. Полномочия обслуживания системы предназначены для пользователей, обслуживающих базы данных в экземпляре менеджера баз данных, который содержит конфиденциальные данные.

Полномочия SYSMAINT предоставляются группе, указанной в параметре конфигурации *sysmaint_group*. Если эта группа задана, членство в этой группе управляется вне менеджера баз данных через утилиту защиты, используемую на вашей платформе.

Только пользователь с системными полномочиями SYSMAINT и выше может:

- Изменять файлы конфигурации баз данных
- Создавать резервные копии баз данных и табличных пространств
- Выполнять восстановление в существующую базу данных
- Выполнять восстановление с повтором транзакций
- Запускать и останавливать экземпляры
- Восстанавливать табличное пространство
- Запускать трассировку
- Делать с помощью монитора баз данных снимки экземпляра менеджера баз данных или его баз данных.

Пользователь с полномочиями SYSMAINT, DBADM и выше может:

- Выполнять запросы состояния табличного пространства
- Изменять файлы хронологии журналов
- Стабилизировать табличное пространство
- Реорганизовать таблицу
- Собрать статистику каталога с помощью утилиты **RUNSTATS**.

Пользователи с полномочиями SYSMAINT имеют также неявную привилегию соединяться с базой данных.

Полномочия управления базой данных (DBADM)

Полномочия DBADM - второй уровень полномочий управления. Они действуют только для конкретной базы данных и позволяют пользователю запускать определенные утилиты, выдавать команды базы данных и обращаться к данным в любой таблице в составе этой базы данных. Вместе с полномочиями DBADM предоставляются привилегии BINDADD, CONNECT, CREATETAB,

CREATE_NOT_FENCED и IMPLICIT_SCHEMA. Только пользователь с полномочиями SYSADM может предоставлять и отозвать полномочия DBADM. Пользователь с полномочиями DBADM может предоставлять другим привилегии для базы данных и может отозвать любую привилегию у любого пользователя, независимо от того, кто ее предоставил.

Только пользователь с полномочиями DBADM и выше может:

- Читать файлы журналов
- Создавать, активировать и отбрасывать мониторы событий.

Пользователь с полномочиями DBADM, SYSMAINT и выше может:

- Выполнять запросы состояния табличного пространства
- Изменять файлы хронологии журналов
- Стабилизировать табличное пространство
- Реорганизовать таблицу
- Собрать статистику каталога с помощью утилиты **RUNSTATS**.

Примечание: DBADM может выполнять перечисленные функции только на той базе данных, для которой у него есть полномочия DBADM.

Понятия, связанные с данным:

- “Полномочия системного администратора (SYSADM)” на стр. 246
- “Полномочия управления системой (SYSCTRL)” на стр. 247
- “Полномочия обслуживания системы (SYSMAINT)” на стр. 248
- “Полномочия LOAD” на стр. 249

Полномочия LOAD

Если пользователю предоставлены полномочия LOAD на уровне базы данных и привилегия INSERT на уровне таблицы, то он может загружать данные в таблицу с помощью команды **LOAD**.

Если пользователю предоставлены полномочия LOAD на уровне базы данных и привилегия INSERT на уровне таблицы, то он может вызвать команду **LOAD RESTART** или **LOAD TERMINATE**, если предыдущей операцией загрузки была загрузка для вставки данных.

Если предыдущей операцией загрузки была загрузка с заменой, то для применения команд **LOAD RESTART** и **LOAD TERMINATE** пользователю дополнительно должна быть предоставлена привилегия DELETE.

Если в операции загрузки применяются таблицы исключений, пользователю должна быть предоставлена привилегия INSERT для таблиц исключений.

Пользователю с такими полномочиями разрешено применять команды **QUIESCE TABLESPACES FOR TABLE**, **RUNSTATS** и **LIST TABLESPACES**.

Понятия, связанные с данным:

- “Privileges, Authorities, and Authorizations Required to Use Load” в *Data Movement Utilities Guide and Reference*
- “Привилегии таблиц и производных таблиц” на стр. 254

Ссылки, связанные с данной темой:

- “RUNSTATS Command” в *Command Reference*
- “QUIESCE TABLESPACES FOR TABLE Command” в *Command Reference*
- “LIST TABLESPACES Command” в *Command Reference*
- “LOAD Command” в *Command Reference*

Привилегии базы данных

На рис. 4 приведен список привилегий базы данных.

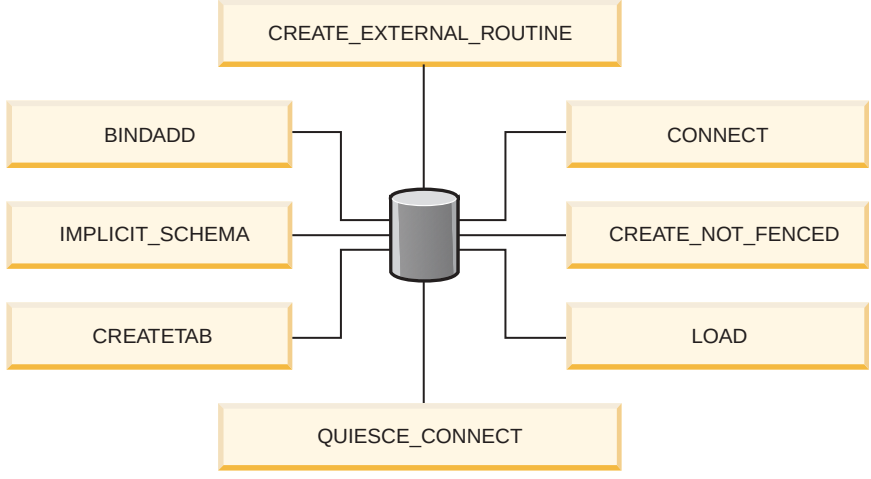


Рисунок 4. Привилегии базы данных

Привилегии базы данных касаются действий на базе данных в целом:

- CONNECT дает пользователю доступ к базе данных
- BINDADD позволяет пользователю создавать новые пакеты в базе данных
- CREATETAB позволяет пользователю создавать новые таблицы в базе данных
- CREATE_NOT_FENCED позволяет пользователю создавать пользовательские функции и “неизолированные” процедуры. Пользовательские функции и “неизолированные” процедуры нужно крайне тщательно тестировать,

поскольку менеджер баз данных не защищает свою память и управляющие блоки от этих пользовательских функций и процедур. (Поэтому плохо написанные и плохо протестированные пользовательские функции и процедуры, которым разрешили работать в “неизолированном режиме”, могут создать серьезные проблемы для вашей системы.)

- Привилегия IMPLICIT_SCHEMA позволяет любому пользователю неявно создать схему, создав оператором CREATE объект с именем еще не существующей схемы. Владельцем неявно созданной схемы становится SYSIBM, а привилегию создавать объекты в этой схеме получает группа PUBLIC (то есть все пользователи).
- Привилегия LOAD позволяет пользователю загрузить данные в таблицу.
- Привилегия QUIESCE_CONNECT позволяет пользователю обращаться к базе данных, когда она стабилизирована.
- Привилегия CREATE_EXTERNAL_ROUTINE позволяет пользователю создавать процедуры для приложений и других пользователей базы данных.

Только пользователи с полномочиями SYSADM или DBADM могут предоставлять эти привилегии другим пользователям и отзывать их.

Примечание: При создании базы данных следующие привилегии автоматически предоставляются группе PUBLIC (то есть всем):

- CREATETAB
- BINDADD
- CONNECT
- IMPLICIT_SCHEMA
- привилегия USE для табличного пространства USERSPACE1
- привилегия SELECT для производных таблиц системного каталога.

Чтобы удалить любую привилегию, пользователь DBADM или SYSADM должен явно отозвать ее у группы PUBLIC.

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260
- “Отзыв привилегий” на стр. 261

Особенности полномочий неявного задания схемы (IMPLICIT_SCHEMA)

Когда создается новая база данных или перенастраивается база данных предыдущего выпуска, полномочия IMPLICIT_SCHEMA для этой базы данных предоставляются группе PUBLIC. С этими полномочиями любой пользователь может создать схему, создав объект и задав имя еще не существующей схемы.

Владельцем неявно созданной схемы становится SYSIBM, а привилегию создавать объекты в этой схеме получает группа PUBLIC (то есть все пользователи).

Если нужно управлять правами создавать объекты схем для этой базы данных, следует отозвать у группы PUBLIC полномочия IMPLICIT_SCHEMA для базы данных. После этого останется только три (3) возможности создать объект схемы:

- Любой пользователь может создать схему, задав собственное имя авторизации в операторе CREATE SCHEMA.
- Любой пользователь с полномочиями DBADM может явно создать любую схему, если она еще не существует, и может при желании сделать владельцем схемы другого пользователя.
- Любой пользователь с полномочиями DBADM имеет полномочия IMPLICIT_SCHEMA для базы данных (независимо от группы PUBLIC) и, следовательно, может неявно создать схему с любым именем во время создания других объектов баз данных. Владельцем неявно созданной схемы становится SYSIBM, а привилегию создавать объекты в этой схеме получает группа PUBLIC (то есть все пользователи).

Пользователь всегда может явно создавать собственные схемы, используя свое собственное имя авторизации.

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260
- “Отзыв привилегий” на стр. 261

Привилегии схем

Привилегии схем относятся к категории привилегий объектов. Привилегии объектов перечислены на рис. 5 на стр. 253.

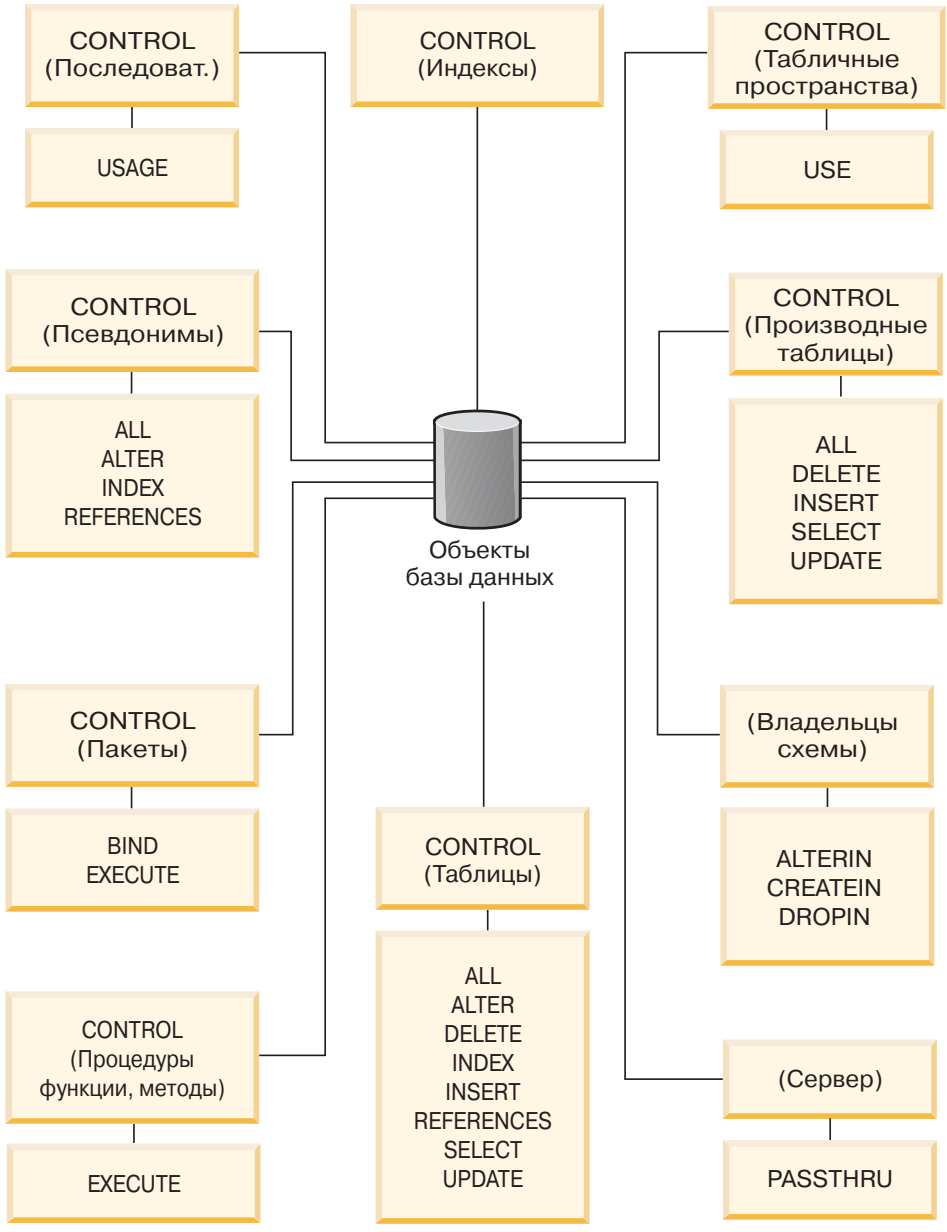


Рисунок 5. Привилегии объектов

Привилегии схем касаются действий для схем в базе данных. Пользователь может быть наделен следующими привилегиями:

- CREATEIN позволяет пользователю создавать объекты в пределах схемы.
- ALTERIN позволяет пользователю изменять объекты в пределах схемы.

- DROPIN позволяет пользователю отбрасывать объекты из схемы.

Владелец схемы имеет все эти привилегии и может передавать их другим. Объекты, доступные в пределах объекта схемы - это таблицы, производные таблицы, индексы, пакеты, типы данных, функции, триггеры, процедуры и алиасы.

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260
- “Отзыв привилегий” на стр. 261

Привилегии табличных пространств

Привилегии табличных пространств затрагивают действия для табличных пространств в базе данных. Пользователю может быть предоставлена привилегия USE для табличного пространства, после чего он сможет создавать таблицы в пределах табличного пространства.

Владелец табличного пространства - в типичном случае создатель с полномочиями SYSADM или SYSCTRL - имеет привилегию USE и может передавать эту привилегию другим. По умолчанию в момент создания базы данных привилегия USE для табличного пространства USERSPACE1 предоставляется группе PUBLIC, но эту привилегию можно отозвать.

Нельзя использовать привилегию USE для SYSCATSPACE и любых временных системных табличных пространств.

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260
- “Отзыв привилегий” на стр. 261

Привилегии таблиц и производных таблиц

Привилегии таблиц и производных таблиц касаются действий для таблиц или производных таблиц в базе данных. Для применения следующих привилегий пользователю должна быть предоставлена привилегия CONNECT на уровне базы данных:

- Привилегия CONTROL предоставляет пользователю все привилегии для таблицы или производной таблицы, включая возможность отбросить таблицу, а также давать и отзывать отдельные привилегии таблицы. Чтобы предоставить привилегию CONTROL, нужно иметь полномочия SYSADM или DBADM. Создатель таблицы автоматически получает привилегию CONTROL для этой таблицы. Создатель производной таблицы автоматически получает привилегию CONTROL, только если у него есть привилегия CONTROL для всех таблиц и производных таблиц, на которые есть ссылки в определении этой производной таблицы, или же полномочия SYSADM или DBADM.

- Привилегия ALTER позволяет пользователю добавлять к таблице столбцы, добавлять и изменять комментарии таблицы и ее столбцов, добавить первичный ключ или ограничение уникальности и создать или отбросить проверочное ограничение таблицы. Пользователь также может создавать в таблице триггеры, хотя для этого потребуются дополнительные полномочия для всех объектов, упоминаемых в триггере (в частности, полномочия SELECT для таблицы, если в триггере есть ссылки на какие-либо столбцы таблицы). Пользователь с привилегией ALTER для всех дочерних таблиц может отбросить первичный ключ; пользователь с привилегией ALTER для таблицы и привилегией REFERENCES для родительской таблицы или привилегией REFERENCES для соответствующих столбцов может создать или отбросить внешний ключ. Пользователь с привилегией ALTER может также добавлять комментарии к таблице с помощью COMMENT ON.
- Привилегия DELETE позволяет пользователю удалять строки из таблицы или производной таблицы.
- Привилегия INDEX позволяет пользователю создать индекс для таблицы. Создатель индекса автоматически получает привилегию CONTROL для этого индекса.
- Привилегия INSERT позволяет пользователю вставлять строки в таблицу или производную таблицу, а также запускать утилиту IMPORT.
- Привилегия REFERENCES позволяет пользователю создать и отбросить внешний ключ, задав родительский статус таблицы по отношению к другим таблицам. Пользователь может обладать этой привилегией и для конкретных столбцов.
- Привилегия SELECT позволяет пользователю получить строку из таблицы или производной таблицы, создать для таблицы производную таблицу и запустить утилиту EXPORT.
- Привилегия UPDATE позволяет пользователю изменить запись в таблице, производной таблице или в одном или нескольких конкретных столбцах таблицы или производной таблицы. Пользователь может обладать этой привилегией и для конкретных столбцов.

Привилегию предоставлять эти привилегии другим можно дать с помощью опции WITH GRANT OPTION оператора GRANT.

Примечание: Когда пользователю или группе предоставляется привилегия CONTROL для таблицы, все остальные привилегии для этой таблицы автоматически предоставляются с опцией WITH GRANT OPTION. Если затем привилегия пользователя CONTROL для таблицы отзывается, у пользователя сохраняются остальные привилегии, которые были предоставлены автоматически. Чтобы отозвать все привилегии, предоставленные с привилегией

CONTROL, нужно либо явно отозвать каждую отдельную привилегию, либо задать ключевое слово ALL в операторе REVOKE, например:

```
REVOKE ALL
  ON EMPLOYEE FROM USER HERON
```

Для типизированных таблиц у привилегий таблиц и производных таблиц есть свои особенности.

Примечание: Привилегии могут предоставляться независимо на каждом уровне иерархии таблиц. В результате пользователь, которому предоставили привилегию для надтаблицы в иерархии типизированных таблиц, может также неявно действовать на все подтаблицы. Однако непосредственно работать с подтаблицей пользователь сможет, только если у него есть необходимая привилегия для этой подтаблицы.

Отношение надтаблица/подтаблица между таблицами иерархии таблиц означает, что такие операции, как SELECT, UPDATE и DELETE затронут строки таблицы назначения операции и все ее подтаблицы (если такие есть). Такое поведение можно назвать *подстановочным*. Например, предположим, что вы создали таблицу сотрудников Employee типа Employee_t с подтаблицей начальников Manager типа Manager_t. Начальник - это особый вид сотрудника, что задается отношением тип/подтип между структурированными типами Employee_t и Manager_t и в соответствующем отношении таблица/подтаблица между таблицами Employee и Manager. Вследствие этого отношения запрос SQL:

```
SELECT * FROM Employee
```

вернет идентификатор объекта и атрибуты Employee_t и для обычных сотрудников, и для начальников. Аналогичным образом, операция изменения данных:

```
UPDATE Employee SET Salary = Salary + 1000
```

увеличит на тысячу оклад как обычным сотрудникам, так и начальникам.

Пользователь с привилегией SELECT для Employee сможет выполнить данную операцию SELECT, даже если у него нет явной привилегии SELECT для Manager. Однако такому пользователю не будет позволено выполнить операцию SELECT непосредственно на подтаблице Manager и, следовательно, не будет предоставлен доступ к собственным (то есть не унаследованным) столбцам таблицы Manager.

Аналогичным образом, пользователь с привилегией UPDATE для Employee сможет выполнить операцию UPDATE для Manager, затронув и рядовых сотрудников, и начальников, даже если у него нет явной привилегии UPDATE для таблицы Manager. Однако такому пользователю не будет позволено

выполнять операции UPDATE непосредственно на подтаблице Manager и, следовательно, не будет позволено изменять собственные (не унаследованные) столбцы таблицы Manager.

Понятия, связанные с данным:

- “Привилегии индексов” на стр. 258

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260
- “Отзыв привилегий” на стр. 261

Ссылки, связанные с данной темой:

- “CREATE VIEW statement” в *SQL Reference, Том 2*
- “SELECT statement” в *SQL Reference, Том 2*

Привилегии пакетов

Пакет - это объект базы данных, содержащий информацию, позволяющую менеджеру баз данных выбрать наиболее эффективный способ доступа к данным для конкретной прикладной программы. Привилегии пакетов позволяют пользователю создавать пакеты и управлять ими. Пользователю нужна привилегия CONNECT для базы данных, чтобы использовать какие-либо из следующих привилегий:

- Привилегия CONTROL позволяет пользователю повторно связать, отбросить и выполнить пакет, а также давать эти привилегии другим. Создатель пакета автоматически получает эту привилегию. Пользователь с привилегией CONTROL получает привилегии BIND и EXECUTE и может также давать привилегии BIND и EXECUTE другим пользователям. Чтобы предоставить привилегию CONTROL, пользователь должен иметь полномочия SYSADM или DBADM.
- Привилегия BIND дает пользователю право выполнять связывание и повторное связывание пакета, а также добавлять новые версии пакета с тем же именем и создателем.
- Привилегия EXECUTE позволяет пользователю запускать пакет.

Примечание: Привилегии пакета относятся ко всем версиям пакета с одинаковым именем и создателем.

Помимо этих привилегий пакетов, привилегия BINDADD для базы данных позволяет пользователям создать новые пакеты или повторно связать существующий пакет в базе данных.

Пользователям с полномочиями выполнять пакеты, содержащие псевдонимы, не нужны дополнительные привилегии и уровень полномочий для псевдонимов в

пакете; однако им надо будет пройти процедуру аутентификации на источниках данных, содержащих объекты, обозначенные псевдонимами. Кроме того, пользователи пакетов должны иметь соответствующие привилегии или уровни полномочий для объектов источников данных на источнике данных.

Если пакет содержит псевдонимы, то может потребоваться выполнить дополнительные действия для авторизации, так как DB2® применяет динамический SQL при подключении к источникам данных семейства DB2. ID авторизации, запустивший пакет в источнике данных, должен располагать соответствующими полномочиями для динамического выполнения пакета в источнике данных.

Понятия, связанные с данным:

- “Привилегии базы данных” на стр. 250

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260
- “Отзыв привилегий” на стр. 261

Привилегии индексов

Создатель индекса или спецификации индекса автоматически получает привилегию CONTROL для индекса. Привилегия CONTROL для индекса сводится к праву отбросить индекс. Чтобы давать привилегию CONTROL для индекса, пользователь должен иметь полномочия SYSADM или DBADM.

Привилегия INDEX уровня таблицы позволяет пользователю создавать индексы для этой таблицы.

Понятия, связанные с данным:

- “Привилегии таблиц и производных таблиц” на стр. 254

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260
- “Отзыв привилегий” на стр. 261

Привилегии последовательностей

Создатель последовательности автоматически получает привилегию USAGE. Привилегия USAGE позволяет использовать выражения NEXTVAL и PREVVAL для этой последовательности. Чтобы разрешить другим пользователям использовать выражения NEXTVAL и PREVVAL, привилегии последовательности надо предоставить всем (то есть группе PUBLIC). Это позволит всем пользователям применять выражения для заданной последовательности.

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260
- “Отзыв привилегий” на стр. 261

Привилегии процедур, функций и методов

Привилегия выполнения дает право на выполнение действий надо всеми типами подпрограмм базы данных, в число которых входят функции, процедуры и методы. Если у пользователя есть привилегия выполнения, он может вызвать подпрограмму, создать функцию на основе этой подпрограммы (относится только к функциям) и указать подпрограмму в любом операторе DDL, в том числе CREATE VIEW и CREATE TRIGGER, либо при определении ограничения.

Привилегия EXECUTE WITH GRANT предоставляется пользователю при определении внешне хранимой процедуры, функции или метода.

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260
- “Отзыв привилегий” на стр. 261

Управление доступом к объектам базы данных

Управление доступом к данным требует понимания прямых и косвенных привилегий, полномочий управления и пакетов. В этом разделе излагаются эти темы и приводятся некоторые примеры.

Прямо предоставленные привилегии хранятся в системном каталоге.

Есть три способа управлять авторизацией:

- Явной авторизацией можно управлять через привилегии, управляемые операторами GRANT и REVOKE
- Неявной авторизацией можно управлять, создавая и отбрасывая объекты
- Неявные привилегии связаны с пакетами.

Примечание: В операторах GRANT и REVOKE, а также в Центре управления разрешено указывать имена групп, содержащие не более 8 символов. Хотя можно задать и более длинное имя группы, при обращении пользователя из этой группы к объекту базы данных будет выдано сообщение об ошибке.

Понятия, связанные с данным:

- “Применение системного каталога для защиты” на стр. 274

Задачи, связанные с данной темой:

Сведения о контроле доступа к объектам баз данных

Управлять доступом к базам данных можно с помощью операторов GRANT и REVOKE. В этом разделе также обсуждаются неявные права доступа и привилегии.

Предоставление привилегий

Ограничения:

Для большинства объектов право предоставлять привилегии принадлежит пользователям с полномочиями SYSADM или DBADM, либо привилегией CONTROL для данного объекта; кроме того, привилегию может предоставить пользователь, обладающий этой привилегией с опцией WITH GRANT OPTION. Привилегии можно предоставлять только для существующих объектов. Чтобы предоставлять привилегию CONTROL другим, пользователь должен иметь полномочия SYSADM или DBADM. Чтобы предоставлять полномочия DBADM, пользователь должен иметь полномочия SYSADM.

Процедура:

С помощью оператора GRANT авторизованный пользователь может предоставлять привилегии. В одном операторе привилегия может предоставляться одному или нескольким именам авторизации или группе PUBLIC, то есть всем пользователям. Обратите внимание на то, что имя авторизации может быть именем не только отдельного пользователя, но и группы.

В операционных системах, в которых имена пользователей и групп могут совпадать, следует указывать, кому предоставляется привилегия - пользователю или группе. И оператор GRANT, и оператор REVOKE поддерживают ключевые слова USER (пользователь) и GROUP (группа). Если этих необязательных ключевых слов нет, менеджер баз данных обращается к утилите защиты операционной системы и определяет, к кому относится имя авторизации - к пользователю или к группе. Если имя авторизации может быть и именем пользователя, и именем группы, возвращается ошибка.

В следующем примере пользователю HERON предоставляются привилегии SELECT для таблицы EMPLOYEE:

```
GRANT SELECT
   ON EMPLOYEE TO USER HERON
```

В следующем примере привилегии SELECT для таблицы EMPLOYEE предоставляются группе HERON:

```
GRANT SELECT
   ON EMPLOYEE TO GROUP HERON
```

Понятия, связанные с данным:

- “Управление доступом к объектам базы данных” на стр. 259

Задачи, связанные с данной темой:

- “Отзыв привилегий” на стр. 261

Ссылки, связанные с данной темой:

- “GRANT (Database Authorities) statement” в *SQL Reference, Том 2*
- “GRANT (Index Privileges) statement” в *SQL Reference, Том 2*
- “GRANT (Package Privileges) statement” в *SQL Reference, Том 2*
- “GRANT (Schema Privileges) statement” в *SQL Reference, Том 2*
- “GRANT (Table, View, or Nickname Privileges) statement” в *SQL Reference, Том 2*
- “GRANT (Server Privileges) statement” в *SQL Reference, Том 2*
- “GRANT (Table Space Privileges) statement” в *SQL Reference, Том 2*
- “GRANT (Sequence Privileges) statement” в *SQL Reference, Том 2*
- “GRANT (Routine Privileges) statement” в *SQL Reference, Том 2*

Отзыв привилегий

Оператор REVOKE позволяет авторизованным пользователям отзывать привилегии, ранее предоставленные другим пользователям.

Ограничения:

Чтобы отозвать привилегии для объектов баз данных, нужно иметь полномочия DBADM, полномочия SYSADM или привилегию CONTROL для этих объектов. Обратите внимание на то, что предоставление привилегии с опцией WITH GRANT OPTION не дает права отозвать эту привилегию. Чтобы отозвать привилегию CONTROL у другого пользователя, нужно иметь полномочия SYSADM или DBADM. Чтобы отозвать полномочия DBADM, нужно иметь полномочия SYSADM. Привилегии можно отзывать только для существующих объектов.

Примечание: Пользователь без полномочий DBADM и привилегии CONTROL для таблицы или производной таблицы не может отозвать привилегию, которую предоставил посредством опции WITH GRANT OPTION. Кроме того, отзыв привилегии не вызывает каскада отзыва этой привилегии у тех, кто получил ее от лишенного теперь этой привилегии пользователя.

Если у пользователя с полномочиями DBADM отзывается предоставленная явным образом привилегия таблицы (или производной таблицы), привилегии для производных таблиц, определенных на данной таблице, **не отзываются**. Это связано с тем, что привилегии производных таблиц доступны в рамках полномочий DBADM и не зависят от явных привилегий для базовых таблиц.

Если вы определили производную таблицу на базе одной или нескольких таблиц или производных таблиц, а затем утратили привилегию SELECT для одной или нескольких из этих таблиц, вы не сможете использовать эту производную таблицу.

Процедура:

Если привилегия была предоставлена пользователю и группе с одним и тем же именем, при ее отзыве нужно использовать ключевое слово GROUP или USER. В следующем примере отзывается привилегия SELECT для таблицы EMPLOYEE у пользователя HERON:

```
REVOKE SELECT
   ON EMPLOYEE FROM USER HERON
```

В следующем примере отзывается привилегия SELECT для таблицы EMPLOYEE у группы HERON:

```
REVOKE SELECT
   ON EMPLOYEE FROM GROUP HERON
```

Обратите внимание на то, что отзыв привилегии у группы не обязательно лишает этой привилегии всех членов группы. Если привилегия была непосредственно предоставлена отдельному пользователю, он сохранит ее, пока не произойдет непосредственного отзыва.

Если у пользователя отзывается привилегия таблицы, отзываются также привилегии для всех зависимых от нее производных таблиц, созданных этим пользователем. Однако отзываются только привилегии, неявно предоставленные системой. Если привилегия для производной таблицы была непосредственно предоставлена другим пользователем, эта привилегия сохранится.

Возможна ситуация, при которой вы захотите предоставить привилегию группе, а затем отозвать эту привилегию только у одного члена группы. Это можно сделать только двумя способами, не получая сообщение об ошибке SQL0556N:

- Можно удалить данного члена из группы или создать новую группу с меньшим числом пользователей и предоставить данную привилегию новой группе.
- Можно отозвать привилегию у группы, а затем предоставить ее отдельным пользователям (ID авторизации).

Примечание: Когда у пользователя отзывается привилегия CONTROL для таблицы или производной таблицы, у него сохраняется право передавать привилегии другим. Получая привилегию CONTROL, пользователь также получает все остальные привилегии с опцией WITH GRANT OPTION. Когда CONTROL отзывается, все остальные привилегии сохраняются с опцией WITH GRANT OPTION, пока они не будут отозваны явным образом.

Все пакеты, зависящие от отозванных полномочий, помечаются как запрещенные; их можно сделать разрешенными повторным связыванием, выполненным пользователем с соответствующими полномочиями. Кроме того, пакеты можно восстановить, если вернуть привилегии связывателю прикладной программы; при выполнении прикладной программы произойдет успешное неявное повторное связывание. Если привилегии отзываются у группы PUBLIC, окажутся запрещены все пакеты, связанные пользователями, права которых на связывание базировались на привилегиях группы PUBLIC. Если у пользователя отзываются полномочия DBADM, запрещаются все пакеты, связанные этим пользователем, включая пакеты, относящиеся к утилитам баз данных. При обращении к пакету, помеченному как запрещенный, система пытается повторно связать этот пакет. Если повторное связывание не проходит, возникает ошибка (SQLCODE -727). В таком случае требуется явное повторное связывание пакетов пользователем с:

- Полномочиями повторно связывать пакеты
- Соответствующими полномочиями для объектов, используемых в пакетах

Эти пакеты следует повторно связать при отзыве привилегий.

Если вы определили триггер на базе одной или нескольких привилегий, а затем утратили одну или несколько из этих привилегий, вы не сможете использовать этот триггер.

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260

Ссылки, связанные с данной темой:

- “REVOKE (Database Authorities) statement” в *SQL Reference, Том 2*
- “REVOKE (Index Privileges) statement” в *SQL Reference, Том 2*
- “REVOKE (Package Privileges) statement” в *SQL Reference, Том 2*
- “REVOKE (Schema Privileges) statement” в *SQL Reference, Том 2*
- “REVOKE (Table, View, or Nickname Privileges) statement” в *SQL Reference, Том 2*
- “REVOKE (Server Privileges) statement” в *SQL Reference, Том 2*
- “REVOKE (Table Space Privileges) statement” в *SQL Reference, Том 2*

- “REVOKE (Routine Privileges) statement” в *SQL Reference, Том 2*

Управление неявной авторизацией путем создания и отбрасывания пакетов

Процедура:

Менеджер баз данных неявно предоставляет определенные привилегии пользователю, выдающему операторы CREATE SCHEMA, CREATE TABLESPACE, CREATE TABLE, CREATE VIEW и CREATE INDEX или создающему новый пакет командой PREP или BIND. Кроме того, привилегии предоставляются создающим объекты пользователям с полномочиями SYSADM или DBADM. Эти привилегии аннулируются при отбрасывании объекта.

Если создаваемый объект - табличное пространство, таблица, индекс или пакет, пользователь получает привилегию CONTROL для объекта. При создании производной таблицы привилегия CONTROL для этой таблицы неявно предоставляется лишь в том случае, если у пользователя есть привилегия CONTROL для всех таблиц и производных таблиц, заданных в определении новой производной таблицы.

Если явно создаваемый объект - схема, владелец получает привилегии ALTERIN, CREATEIN и DROPIN с опцией WITH GRANT OPTION. Для неявно созданной схемы привилегия CREATEIN предоставляется группе PUBLIC.

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260
- “Отзыв привилегий” на стр. 261

Настройка принадлежности пакета

Процедура:

Команды BIND и PRECOMPILE создают или изменяют пакет прикладной программы. В любой из них можно задать имя владельца итогового пакета с помощью опции OWNER. Есть несложные правила именования владельца пакета:

- Всякий пользователь может назвать владельцем себя самого. Это значение используется по умолчанию, если для опция `OWNER` не задана.
- Пользователь с полномочиями SYSADM или DBADM может назначить владельцем объекта любой ID авторизации, указав его в опции OWNER.

Опция OWNER поддерживается не всеми операционными системами, которые позволяют связать пакет с помощью продуктов базы данных DB2.

Ссылки, связанные с данной темой:

- “BIND Command” в *Command Reference*
- “PRECOMPILE Command” в *Command Reference*

Неявно запрашиваемые привилегии для пакета

Доступ к данным в базе данных может быть затребован прикладными программами, а также лицами, участвующими в диалоговых сеансах рабочей станции. Пакет содержит операторы, позволяющие пользователям выполнять ряд действий для многих объектов баз данных. Каждое из этих действий требует одной или нескольких привилегий.

Привилегии, предоставленные группе PUBLIC и отдельным пользователям и группам, связывающим пакет, используются для проведения авторизации при связывании статического SQL. Привилегии, предоставленные через группы, *не* используются для проведения авторизации при связывании статического SQL. Пользователь с допустимым *authID*, связывающий пакет, либо должен до этого явно получить все требуемые привилегии для выполнения статических операторов SQL в пакете, либо должен неявно получить необходимые привилегии через группу PUBLIC, если только при связывании пакета не было задано VALIDATE RUN. Если при запуске BIND было задано VALIDATE RUN, неудачи авторизации для каких-либо статических операторов SQL в пакете не сорвут выполнение BIND, эти операторы SQL будут повторно разрешены во время запуска. *Все* привилегии пользователя, группы и группы PUBLIC используются при проверке прав пользователя (привилегии BIND или BINDADD) связать пакет.

Пакеты могут включать как статические, так и динамические операторы SQL. Чтобы обработать пакет со статическими операторами SQL, пользователю достаточно иметь привилегию EXECUTE для этого пакета. Этот пользователь может затем косвенно получить привилегии связывателя пакета для всех статических операторов SQL в пакете, но только в пределах ограничений, налагаемых пакетом.

Чтобы обработать пакет какими-либо динамическими операторами SQL, пользователю необходимо иметь привилегию EXECUTE для этого пакета. Пользователю нужна привилегия EXECUTE для пакета плюс все привилегии, требуемые для выполнения динамических операторов SQL в пакете. Полномочия и привилегии связывателя используются для всех статических операторов SQL в пакете.

Понятия, связанные с данным:

- “Неявно запрашиваемые привилегии для пакета со ссылками на псевдонимы” на стр. 266

Ссылки, связанные с данной темой:

- “BIND Command” в *Command Reference*

Неявно запрашиваемые привилегии для пакета со ссылками на псевдонимы

Когда пакет содержит ссылки на псевдонимы, процесс авторизации для создателей пакета и пользователей пакета несколько усложняется. Когда создатель пакета успешно свяжет пакеты, содержащие псевдонимы, создатель пакета не должен проходить аутентификацию и проверку привилегий для обозначенных псевдонимами таблиц и производных таблиц на источниках данных. Однако исполнитель пакета должен пройти аутентификацию и проверку полномочий на источниках данных.

Например, допустим, что файл .SQC создателя пакета содержит несколько операторов SQL. Один оператор, статический, содержит ссылку на локальную таблицу. Другой оператор, динамический, ссылается на псевдоним. При связывании пакета authid создателя пакета используется для проверки привилегий для локальной таблицы, но проверки для объектов источников данных, обозначенных псевдонимами, не проводятся. Если другой пользователь с привилегией EXECUTE попытается выполнить пакет, то ему не потребуется проходить дополнительную проверку привилегий в связи с наличием оператора, ссылающегося на таблицу. Однако в связи с наличием оператора со ссылкой на псевдоним пользователю придется пройти процедуру аутентификации и проверки привилегий в источнике данных.

Если файл .SQC содержит только динамические операторы SQL, некоторые из которых ссылаются на таблицы и псевдонимы, то для локальных объектов и псевдонимов будут выполнены аналогичные процедуры авторизации DB2®. Пользователи пакетов должны проходить проверку привилегий для всех локальных объектов (таблиц, производных таблиц) в операторах, а также для объектов псевдонимов (пользователи пакетов должны проходить аутентификацию и проверку привилегий на том источнике данных, который содержит объекты, обозначенные псевдонимами). В обоих случаях пользователи пакета должны иметь привилегию EXECUTE.

ID и пароль исполнителя пакета используется при всех аутентификациях и проверках привилегий на источниках данных. Эта информация может быть изменена путем создания отображения пользователей.

Примечание: В статических SQL нельзя использовать псевдонимы. Не используйте опцию DYNAMICRULES (со значением BIND) для пакетов, содержащих псевдонимы.

Может случиться, что пакеты, содержащие псевдонимы, потребуют дополнительных шагов авторизации, поскольку DB2 использует динамические SQL при соединении с источниками данных семейства DB2. ID авторизации,

запустивший пакет в источнике данных, должен располагать соответствующими полномочиями для динамического выполнения пакета в источнике данных.

Понятия, связанные с данным:

- “Неявно запрашиваемые привилегии для пакета” на стр. 265

Управление доступом к данным с помощью производных таблиц

Производная таблица дает возможность управлять доступом и распространять привилегии для таблицы, если разрешить:

- Доступ только к указанным столбцам таблицы.

  Для пользователей и прикладных программ, требующих доступа лишь к определенным столбцам таблицы, полномочный пользователь может создать производную таблицу, ограничив доступные столбцы необходимыми для работы.

- Доступ только к подмножеству строк таблицы.

  Задавая условие WHERE в подзапросе определения производной таблицы, полномочный пользователь может ограничить доступные через производную таблицу строки.

- Доступ только к подмножеству строк или столбцов в таблицах или производных таблицах источников данных. Если для доступа к источникам данных применяются псевдонимы, то можно создать локальные производные таблицы DB2®, ссылающиеся на псевдонимы. Эти производные таблицы могут ссылаться на псевдонимы из одного или нескольких источников данных.

  Примечание: Поскольку вы можете создать производную таблицу с псевдонимами, ссылающимися на несколько источников данных, ваши пользователи смогут обращаться из одной производной таблицы к данным во многих источниках данных. Такие производные таблицы называются *распределенными производными таблицами*.Они полезны при объединении информации из столбцов конфиденциальных таблиц в распределенной среде или при недостаточности привилегий отдельных пользователей для определенных объектов в источниках данных.

Чтобы создать производную таблицу, пользователю нужны полномочия SYSADM или DBADM или привилегии CONTROL или SELECT для всех таблиц и производных таблиц, упоминаемых в определении производной таблицы. Пользователь также должен иметь право создать объект в схеме, заданной для производной таблицы. Речь идет о привилегии CREATEIN для существующей схемы или полномочиях IMPLICIT_SCHEMA для базы данных, если схема еще не существует.

Если вы создаете производные таблицы, ссылающиеся на псевдонимы, вам не нужны дополнительные полномочия для объектов источников данных (таблиц и производных таблиц), обозначенных в производной таблице псевдонимами; однако вашим пользователям, когда они обратятся к производной таблице, понадобятся полномочия SELECT или эквивалентный уровень авторизации для базовых объектов источников данных.

Если у ваших пользователей не будет нужных полномочий в источниках данных для базовых объектов (таблиц и производных таблиц), вы можете:

1. Создать производную таблицу источника данных с теми столбцами из таблицы источника данных, которые доступны для данного пользователя
2. Предоставить пользователям привилегию SELECT для этой производной таблицы
3. Создать псевдоним, ссылающийся на производную таблицу

Тогда пользователи смогут обращаться к столбцам, выдавая оператор SELECT с новым псевдонимом.

В следующем сценарии подробно показан пример использования производной таблицы для ограничения доступа к информации.

Многим людям может по разным причинам потребоваться доступ к информации в таблице STAFF (сотрудники). Например:

- Отделу кадров нужны права на изменение и просмотр всей таблицы.

  Это требование легко удовлетворить, предоставив привилегии SELECT и UPDATE для таблицы STAFF группе PERSONNL (отделу кадров):

  ```
  GRANT SELECT,UPDATE ON TABLE STAFF TO GROUP PERSONNL
  ```

- Начальникам отделов нужно просматривать информацию о зарплатах их подчиненных.

  Это требование можно удовлетворить, создав по производной таблице для каждого начальника отдела. Например, для начальника отдела номер 51 можно создать следующую производную таблицу:

  ```
  CREATE VIEW EMP051 AS
     SELECT NAME,SALARY,JOB FROM STAFF
     WHERE DEPT=51
  GRANT SELECT ON TABLE EMP051 TO JANE
  ```

  Начальник отдела с именем авторизации JANE запрашивать производную таблицу EMP051, как и таблицу STAFF. Обращаясь к производной таблице EMP051 таблицы STAFF, она увидит следующую информацию:

NAME	SALARY	JOB
Fraye	45150.0	Менеджер
Williams	37156.5	Продавец

NAME	SALARY	JOB
Smith	35654.5	Продавец
Lundquist	26369.8	Клерк
Wheeler	22460.0	Клерк

- Всем пользователям нужно право искать других сотрудников. Это требование можно удовлетворить, создав производную таблицу из столбца NAME таблицы STAFF и столбца LOCATION таблицы ORG, и объединив эти две таблицы по их столбцам DEPT и DEPTNUMB:

```
CREATE VIEW EMPLOCS AS
   SELECT NAME, LOCATION FROM STAFF, ORG
   WHERE STAFF.DEPT=ORG.DEPTNUMB
GRANT SELECT ON TABLE EMPLOCS TO PUBLIC
```

  Пользователи, которые обратятся к производной таблице мест работы сотрудников, увидят следующую информацию:

NAME	LOCATION
Molinare	New York
Lu	New York
Daniels	New York
Jones	New York
Hanes	Boston
Rothman	Boston
Ngan	Boston
Kermisch	Boston
Sanders	Washington
Pernal	Washington
James	Washington
Sneider	Washington
Marenghi	Atlanta
O’Brien	Atlanta
Quigley	Atlanta
Naughton	Atlanta
Abrahams	Atlanta
Koonitz	Chicago
Plotz	Chicago
Yamaguchi	Chicago

NAME	LOCATION
Scoutten	Chicago
Fraye	Dallas
Williams	Dallas
Smith	Dallas
Lundquist	Dallas
Wheeler	Dallas
Lea	San Francisco
Wilson	San Francisco
Graham	San Francisco
Gonzales	San Francisco
Burke	San Francisco
Quill	Denver
Davis	Denver
Edwards	Denver
Gafney	Denver

Задачи, связанные с данной темой:

- “Создание производной таблицы” на стр. 137
- “Предоставление привилегий” на стр. 260

Отслеживание доступа к данным с помощью утилиты аудита

Утилита аудита DB2® создает файл аудита для ряда заранее определенных событий базы данных, который может применяться пользователем. Не запрещая доступ к данным, утилита аудита обеспечивает мониторинг и протоколирование попыток обращения и изменения объектов данных.

Для использования утилиты аудита **db2audit** требуются полномочия SYSADM.

Понятия, связанные с данным:

- “Утилита аудита DB2 - Введение” на стр. 283

Шифрование данных

Составной частью плана защиты может быть шифрование данных. Для шифрования и расшифровки данных можно воспользоваться встроенными функциями: ENCRYPT, DECRYPT_BIN, DECRYPT_CHAR и GETHINT.

Функция ENCRYPT шифрует данные при помощи метода шифрации на основе паролей. Эти функции также позволяют задать подсказку к паролю. Подсказка к паролю встраивается в зашифрованные данные. После шифрования единственная возможность расшифровки данных - это использование точного пароля. При использовании этих функций следует разработать план действий, который будет применяться в случае забытых паролей и недоступных данных.

Результат функций ENCRYPT имеет тот же тип данных, что и первый аргумент.

Шифрование доступно только для переменных типа VARCHAR.

Объявляемая длина результата - одна из следующих:

- Длина аргумента данных плюс 42, если задан необязательный параметр hint (подсказка).
- Длина аргумента данных плюс 10, если не задан необязательный параметр hint.

Функции DECRYPT_BIN и DECRYPT_CHAR расшифровывают данные на основе пароля.

Результат функций DECRYPT_BIN и DECRYPT_CHAR имеет тот же тип данных, что и первый аргумент.

Объявляемая длина результата равна длине первоначальных данных.

Функция GETHINT возвращает встроенную подсказку к паролю. Подсказка к паролю - это фраза, которая помогает владельцам данных вспоминать пароли. Например, слово “океан” можно использовать как подсказку, помогающую вспомнить пароль ″Тихий″.

Пароль, применяемый для шифрования данных, определяется одним из двух способов:

- С помощью аргумента пароля. Пароль - это строка, явно передаваемая при вызове функции ENCRYPT. Данные будут шифроваться и дешифроваться при помощи данного пароля.
- С помощью пароля из специального регистра. Оператор SET ENCRYPTION PASSWORD зашифровывает значение пароля и отправляет зашифрованный пароль менеджеру баз данных, который сохраняет его в специальном регистре. Функции ENCRYPT, DECRYPT_BIN и DECRYPT_CHAR, вызванные без параметра пароля, используют значение специального регистра ENCRYPTION PASSWORD.

  Первоначальное значение специального регистра (значение по умолчанию) - пустая строка.

Допустимые длины паролей - от 6 до 127 включительно. Длина подсказки может составлять от 0 до 32 символов включительно.

Если клиент задал значение специального регистра ENCRYPTION PASSWORD, то пароль зашифровывается клиентом, отправляется на сервер базы данных, а затем расшифровывается. Чтобы не оставлять пароль в читаемом виде, его повторно шифруют на сервере базы данных. Перед применением значения специального регистра функции DECRYPT_BIN и DECRYPT_CHAR должны его расшифровать. Значение ENCRYPTION PASSWORD также хранится в зашифрованном виде.

Ссылки, связанные с данной темой:

- “SET ENCRYPTION PASSWORD statement” в *SQL Reference, Том 2*
- “DECRYPT_BIN and DECRYPT_CHAR scalar functions” в *SQL Reference, Том 1*
- “ENCRYPT scalar function” в *SQL Reference, Том 1*
- “GETHINT scalar function” в *SQL Reference, Том 1*

Полномочия и привилегии, необходимые для выполнения различных задач

Не все организации одинаково распределяют обязанности между должностями. В Табл. 6 показан список некоторых распространенных названий должностей, обязанностей, обычно соответствующих этим должностям, и полномочий и привилегий, необходимых для выполнения этих обязанностей.

Таблица 6. Типичные названия должностей, обязанности и требуемая авторизация

ДОЛЖНОСТЬ	ОБЯЗАННОСТИ	ТРЕБУЕМАЯ АВТОРИЗАЦИЯ
Администратор отдела	Руководит системой отдела; создает базы данных	Полномочия SYSCTRL. Полномочия SYSADM, если для отдела используется собственный экземпляр.
Администратор безопасности	Дает другим пользователям некоторые или все авторизации и привилегии	Полномочия SYSADM или DBADM.
Администратор баз данных	Проектирует, строит, обрабатывает, охраняет и обслуживает одну или несколько баз данных	Полномочия DBADM и SYSMAINT для одной или нескольких баз данных. В некоторых случаях полномочия SYSCTRL.
Системный оператор	Ведет мониторинг базы данных и выполняет резервное копирование	Полномочия SYSMAINT.

Таблица 6. Типичные названия должностей, обязанности и требуемая авторизация (продолжение)

ДОЛЖНОСТЬ	ОБЯЗАННОСТИ	ТРЕБУЕМАЯ АВТОРИЗАЦИЯ
Прикладной программист	Разрабатывает и тестирует прикладные программы менеджера баз данных; может также создавать таблицы с тестовыми данными	BINDADD, BIND для существующего пакета, CONNECT и CREATETAB для одной или нескольких баз данных, привилегии некоторой конкретной схемы и список привилегий для некоторых таблиц.
Пользователь-аналитик	Определяет требования к данным для прикладной программы, исследуя производные таблицы системного каталога	SELECT для производных таблиц каталога; CONNECT для одной или нескольких баз данных.
Конечный пользователь программы	Запускает прикладную программу	EXECUTE для пакета; CONNECT для одной или нескольких баз данных. Смотрите примечание после этой таблицы.
Консультант информационного центра	Определяет требования к данным для пользователя с правом запроса; предоставляет данные, создавая таблицы и производные таблицы и предоставляя доступ к объектам баз данных	Полномочия DBADM для одной или нескольких баз данных.
Пользователь с правом запроса	Выдает операторы SQL, чтобы получить, добавить, удалить или изменить данные; может сохранять результаты в виде таблиц	CONNECT для одной или нескольких баз данных; CREATEIN для схемы создаваемых таблиц и производных таблиц; SELECT, INSERT, UPDATE, DELETE для некоторых таблиц и производных таблиц.

Примечание: Если прикладная программа содержит динамические операторы SQL, конечному пользователю программы могут понадобиться и другие привилегии, помимо EXECUTE и CONNECT - такие, как SELECT, INSERT, DELETE и UPDATE.

Понятия, связанные с данным:

- “Полномочия системного администратора (SYSADM)” на стр. 246
- “Полномочия управления системой (SYSCTRL)” на стр. 247
- “Полномочия обслуживания системы (SYSMAINT)” на стр. 248
- “Полномочия управления базой данных (DBADM)” на стр. 248
- “Полномочия LOAD” на стр. 249

- “Привилегии базы данных” на стр. 250

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260
- “Отзыв привилегий” на стр. 261

Применение системного каталога для защиты

Информация о каждой базе данных автоматически регистрируется в наборе производных таблиц, называемом системным каталогом, который создается при построении базы данных. В этом системном каталоге описаны таблицы, столбцы, индексы, программы, привилегии и другие объекты.

В перечисленных ниже производных таблицах хранится информация о привилегиях различных пользователей и идентификаторы пользователей, предоставивших привилегии:

SYSCAT.DBAUTH	Содержит список привилегий базы данных
SYSCAT.TABAUTH	Содержит список привилегий таблиц и производных таблиц
SYSCAT.COLAUTH	Содержит список привилегий столбцов
SYSCAT.PACKAGEAUTH	Содержит список привилегий пакетов
SYSCAT.INDEXAUTH	Содержит список привилегий индексов
SYSCAT.SCHEMAAUTH	Содержит список привилегий схем
SYSCAT.PASSTHRUAUTH	Содержит список привилегий сервера
SYSCAT.ROUTINEAUTH	Содержит список привилегий подпрограмм (функций, методов и хранимых процедур)

Для привилегий, предоставленных пользователям системой, в качестве лица, предоставившего привилегии, указывается SYSIBM. SYSADM, SYSMAINT и SYSCTRL в системном каталоге не описаны.

Операторы CREATE и GRANT заносят привилегии в системный каталог. Пользователи с полномочиями SYSADM и DBADM могут предоставлять и отзывать привилегию SELECT для производных таблиц системного каталога.

Задачи, связанные с данной темой:

- “Получение имен авторизации, которым предоставлены привилегии” на стр. 275
- “Получение всех имен с полномочиями DBADM” на стр. 276
- “Получение имен с правом доступа к таблице” на стр. 276

- “Получение всех привилегий, предоставленных пользователям” на стр. 277
- “Защита производных таблиц системного каталога” на стр. 278

Ссылки, связанные с данной темой:

- “SYSCAT.COLAUTH catalog view” в *SQL Reference, Том 1*
- “SYSCAT.DBAUTH catalog view” в *SQL Reference, Том 1*
- “SYSCAT.INDEXAUTH catalog view” в *SQL Reference, Том 1*
- “SYSCAT.PACKAGEAUTH catalog view” в *SQL Reference, Том 1*
- “SYSCAT.SCHEMAAUTH catalog view” в *SQL Reference, Том 1*
- “SYSCAT.TABAUTH catalog view” в *SQL Reference, Том 1*
- “SYSCAT.PASSTHRUAUTH catalog view” в *SQL Reference, Том 1*
- “SYSCAT.ROUTINEAUTH catalog view” в *SQL Reference, Том 1*

Сведения об использовании системного каталога для хранения данных защиты

В этом разделе описаны способы определения пользователей, у которых есть определенные права доступа к базе данных.

Получение имен авторизации, которым предоставлены привилегии

Процедура:

Ни одна из производных таблиц системного каталога не содержит информации обо всех привилегиях. Следующий оператор получает все имена авторизации с привилегиями:

```
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'DATABASE' FROM SYSCAT.DBAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'TABLE   ' FROM SYSCAT.TABAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'PACKAGE ' FROM SYSCAT.PACKAGEAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'INDEX   ' FROM SYSCAT.INDEXAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'COLUMN  ' FROM SYSCAT.COLAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SCHEMA  ' FROM SYSCAT.SCHEMAAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SERVER  ' FROM SYSCAT.PASSTHRUAUTH
ORDER BY GRANTEE, GRANTEETYPE, 3
```

Время от времени следует сравнивать список, полученный этим оператором, со списками имен пользователей и групп, определенных в системной утилите защиты. Это позволит выявить устаревшие имена авторизации.

Примечание: Если вы поддерживаете удаленных клиентов базы данных, может оказаться, что имя авторизации определено только на удаленном клиенте и отсутствует на сервере вашей базы данных.

Понятия, связанные с данным:

- “Применение системного каталога для защиты” на стр. 274

Получение всех имен с полномочиями DBADM

Процедура:

Следующий оператор получает все имена авторизации, которым были непосредственно предоставлены полномочия DBADM:

```
SELECT DISTINCT GRANTEE FROM SYSCAT.DBAUTH
   WHERE DBADMAUTH = 'Y'
```

Понятия, связанные с данным:

- “Полномочия управления базой данных (DBADM)” на стр. 248
- “Применение системного каталога для защиты” на стр. 274

Получение имен с правом доступа к таблице

Процедура:

Следующий оператор получает все имена авторизации, которым были непосредственно предоставлены права доступа к таблице EMPLOYEE со спецификатором JAMES:

```
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
   WHERE TABNAME = 'EMPLOYEE'
     AND TABSCHEMA = 'JAMES'
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
   WHERE TABNAME = 'EMPLOYEE'
     AND TABSCHEMA = 'JAMES'
```

Чтобы узнать, кто может изменять таблицу EMPLOYEE с квалификатором JAMES, выполните следующий оператор:

```
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
   WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
      (CONTROLAUTH  = 'Y' OR
       UPDATEAUTH   = 'Y' OR  UPDATEAUTH   = 'G')
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.DBAUTH
   WHERE DBADMAUTH = 'Y'
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
   WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
   PRIVTYPE = 'U'
```

Этот оператор даст все имена авторизации с полномочиями DBADM, а также имена, которым были непосредственно предоставлены привилегии CONTROL и UPDATE. Однако он не даст имена авторизации тех пользователей, которые обладают только полномочиями SYSADM.

Не забывайте, что некоторые имена авторизации могут быть группами, а не отдельными пользователями.

Понятия, связанные с данным:

- “Привилегии таблиц и производных таблиц” на стр. 254
- “Применение системного каталога для защиты” на стр. 274

Получение всех привилегий, предоставленных пользователям

Процедура:

Посылая запросы на производные таблицы системного каталога, пользователь может получать список своих привилегий и список привилегий, которые он передал другим пользователям. Например, следующий оператор получит список привилегий базы данных, непосредственно предоставленных отдельному имени авторизации:

```
SELECT * FROM SYSCAT.DBAUTH
   WHERE GRANTEE = USER AND GRANTEETYPE = 'U'
```

Следующий оператор получит список привилегий таблицы, непосредственно предоставленных заданным пользователем:

```
SELECT * FROM SYSCAT.TABAUTH
   WHERE GRANTOR  = USER
```

Следующий оператор получит список привилегий отдельного столбца, непосредственно предоставленных заданным пользователем:

```
SELECT * FROM SYSCAT.COLAUTH
   WHERE GRANTOR  = USER
```

Ключевое слово USER в этих операторах всегда равно значению имени авторизации пользователя.USER является специальным регистром только для чтения.

Понятия, связанные с данным:

- “Привилегии, полномочия и авторизация” на стр. 243
- “Привилегии базы данных” на стр. 250
- “Применение системного каталога для защиты” на стр. 274

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260

- “Отзыв привилегий” на стр. 261

Защита производных таблиц системного каталога

Процедура:

При создании базы данных привилегия SELECT для производных таблиц системного каталога предоставляется группе PUBLIC. В большинстве случаев это не угрожает безопасности. Однако для особо конфиденциальных данных это может оказаться неприемлемо, поскольку в этих таблицах описаны все объекты базы данных. В таком случае можно аннулировать привилегию SELECT у группы PUBLIC и затем предоставить привилегию SELECT нужным пользователям. Предоставление и отзыв привилегии SELECT для производных таблиц системного каталога делается также, как для любой производной таблицы, но требует полномочий SYSADM или DBADM.

Как минимум, стоит ограничить доступ к следующим производным таблицам каталога:

- SYSCAT.DBAUTH
- SYSCAT.TABAUTH
- SYSCAT.PACKAGEAUTH
- SYSCAT.INDEXAUTH
- SYSCAT.COLAUTH
- SYSCAT.PASSTHRUAUTH
- SYSCAT.SCHEMAAUTH

Тогда информация о привилегиях пользователей перестанет быть доступной для всех, кто обращается к базе данных. С этой информацией неаутентифицированный пользователь мог бы получить несанкционированный доступ к базе данных.

Кроме того, нужно проверить, для каких столбцов собирается статистическая информация. Некоторая статистическая информация, хранящаяся в системном каталоге, может содержать конфиденциальную информацию. В этом случае можно отозвать привилегию SELECT у группы PUBLIC для производных таблиц каталога SYSCAT.COLUMNS и SYSCAT.COLDIST.

Если вы хотите ограничить доступ к производным таблицам системного каталога, можно определить производные таблицы, которые позволят каждому имени авторизации получать информацию о своих собственных привилегиях.

Например, следующая производная таблица MYSELECTS включает владельца и имена всех таблиц, для которых имени авторизации пользователя была предоставлена привилегия SELECT:

```
CREATE VIEW MYSELECTS AS
   SELECT TABSCHEMA, TABNAME FROM SYSCAT.TABAUTH
   WHERE GRANTEETYPE = 'U'
     AND GRANTEE = USER
     AND SELECTAUTH = 'Y'
```

Ключевое слово USER в этом операторе всегда равно значению имени авторизации.

Следующий оператор сделает производную таблицу доступной каждому имени авторизации:

```
GRANT SELECT ON TABLE MYSELECTS TO PUBLIC
```

И, наконец, не забудьте отозвать привилегию SELECT для следующей базовой таблицы:

```
REVOKE SELECT ON TABLE SYSCAT.TABAUTH FROM PUBLIC
```

Понятия, связанные с данным:

- “Статистика из каталога” в *Руководство администратора: Производительность*
- “Привилегии базы данных” на стр. 250
- “Применение системного каталога для защиты” на стр. 274

Задачи, связанные с данной темой:

- “Предоставление привилегий” на стр. 260
- “Отзыв привилегий” на стр. 261

Поддержка брандмауэра - Введение

Брандмауэр - это набор программ, расположенных на сервере шлюза, которые применяются для предотвращения несанкционированного доступа к системе или сети.

Существует четыре типа брандмауэров:

1. Брандмауэры уровня сети, брандмауэры с фильтрами пакетов, или брандмауэры маршрутизатора с фильтром
2. Обычные брандмауэры Proxy уровня прикладной программы
3. Брандмауэры уровня канала передачи данных, или прозрачные брандмауэры Proxy
4. Брандмауэры SMLI (Stateful multi-layer inspection)

Существует ряд продуктов брандмауэров, относящихся к одному из указанных выше типов. Кроме того, существует ряд продуктов, сочетающих в себе брандмауэры нескольких типов.

Понятия, связанные с данным:

- “Брандмауэры маршрутизатора с фильтром” на стр. 280
- “Брандмауэры Proxy уровня прикладной программы” на стр. 280
- “Брандмауэры уровня канала связи” на стр. 281
- “Брандмауэры SMLI (Stateful multi-layer inspection)” на стр. 281

Брандмауэры маршрутизатора с фильтром

Этот тип брандмауэра также называется брандмауэром уровня сети или брандмауэром с фильтрами пакетов. Такие брандмауэры выполняют фильтрацию поступающих пакетов в соответствии со значениями атрибутов протокола. В число таких атрибутов может входить адрес отправителя, адрес получателя, тип протокола, исходный и целевой порт, а также некоторые другие атрибуты протокола.

Для применения любого брандмауэра (за исключением SOCKS) необходимо убедиться, что все порты DB2® могут применяться для приема и отправки пакетов. Порт 523 выделен в DB2 для сервера администратора DB2 (DAS), который применяется различными инструментами DB2. С помощью файла служб определите номера портов, применяемые различными экземплярами сервера, и свяжите номера портов с именами служб в файле конфигурации менеджера баз данных.

Понятия, связанные с данным:

- “Поддержка брандмауэра - Введение” на стр. 279

Брандмауэры Proxy уровня прикладной программы

Сервер Proxy выполняет роль промежуточного узла между клиентом и сервером Web. Брандмауэр Proxy играет роль шлюза для запросов, поступающих от клиентов. При получении запроса клиента на брандмауэре программное обеспечение Proxy определяет адрес целевого сервера. Proxy преобразует адрес, выполняет необходимые операции управления доступом, заносит информацию в журнал и подключается к серверу от имени клиента.

Продукт DB2® Connect, установленный на компьютере брандмауэра, может играть роль Proxy для целевого сервера. Кроме того, сервер DB2, установленный на компьютере брандмауэра и играющий роль сервера маршрутизации для целевого сервера, может выполнять функции Proxy прикладной программы.

Понятия, связанные с данным:

- “Поддержка брандмауэра - Введение” на стр. 279

Брандмауэры уровня канала связи

Этот тип брандмауэра также называется прозрачным брандмауэром Proxy. Прозрачный брандмауэр Proxy изменяет в запросах и ответах только те поля, которые необходимы для аутентификации и идентификации Proxy. Примером прозрачного брандмауэра Proxy может служить SOCKS.

DB2® поддерживает протокол SOCKS версии 4.

Понятия, связанные с данным:

- “Поддержка брандмауэра - Введение” на стр. 279

Брандмауэры SMLI (Stateful multi-layer inspection)

Этот тип брандмауэра представляет собой более сложную версию брандмауэра с фильтрацией пакетов. Он проверяет все семь уровней модели взаимодействия открытых систем (OSI). Каждый пакет сравнивается с известными дружественными пакетами. В отличие от брандмауэров маршрутизатора с фильтром, которые проверяют только заголовок пакета, брандмауэры SMLI проверяют весь пакет, включая поле данных.

Понятия, связанные с данным:

- “Поддержка брандмауэра - Введение” на стр. 279

Глава 5. Аудит действий DB2

Утилита аудита DB2 - Введение

Для управления известным или ожидаемым доступом к данным можно использовать аутентификацию, полномочия и привилегии, но этого может оказаться недостаточно, чтобы предотвратить несанкционированный или незапланированный доступ к данным. Для обнаружения такого рода доступа к данным в DB2® предусмотрена утилита аудита. Мониторинг и последующий анализ нежелательного доступа к данным позволяет улучшить управление доступом и полностью исключить злонамеренные и случайные неавторизованные обращения к данным. Мониторинг доступа программ и индивидуального доступа пользователей к данным, включая действия по управлению системой, обеспечивает возможность записи хронологии действий в системах баз данных.

Утилита аудита DB2 создает файл аудита для ряда заранее определенных событий базы данных. При необходимости вы можете изменить этот файл. Записи, генерируемые этой утилитой, хранятся в файле журнала аудита. Анализ этих записей позволяет выявить типичные случаи несанкционированного использования системы. Выявив такие случаи, можно предпринять действия по уменьшению или исключению неверного применения системы.

Утилита аудита действует на уровне экземпляра и записывает все действия уровня экземпляра и все действия уровня баз данных.

При работе в среде многораздельной базы данных многие события, которые можно отслеживать, происходят в разделе, с которым соединен пользователь (узел координатора), или же на узле каталога (если они не находятся в одном разделе). В связи с этим записи аудита могут генерироваться в нескольких разделах. В каждую запись аудита входят идентификаторы узла координатора и исходного узла.

Журнал аудита (db2audit.log) и файл конфигурации аудита (db2audit.cfg) расположены в подкаталоге `security` экземпляра. При создании экземпляра операционная система устанавливает для этих файлов разрешение на чтение/запись (где это возможно). По умолчанию разрешение чтение/запись устанавливается только для экземпляра владельца. Изменять эти разрешения не рекомендуется.

Пользователи утилиты аудита (db2audit) должны обладать полномочиями SYSADM.

Утилиту аудита надо останавливать и запускать явно. При запуске утилита аудита использует существующую информацию о своей конфигурации. Поскольку утилита аудита не зависит от сервера DB2, она остается активной даже при завершении работы экземпляра. Фактически, когда экземпляр остановлен, записи аудита могут генерироваться и помещаться в файл журнала.

Управляя действиями утилиты аудита, ее авторизованные пользователи могут:

- Запустить запись событий аудита на экземпляре DB2.
- Остановить запись событий аудита на экземпляре DB2.
- Задать параметры работы утилиты аудита, в том числе выбрать категории событий, сведения о которых сохраняются утилитой аудита.
- Запросить описание текущей конфигурации аудита.
- Принудительно записать любые отложенные записи аудита из экземпляра в журнал аудита.
- Извлечь записи аудита, сформатировав и скопировав их из журнала аудита в плоский файл или в файлы ASCII с ограничителями. Есть две причины для извлечения записей журнала: подготовка к анализу или подготовка к сокращению.
- Удалить часть записей из текущего журнала аудита.

Примечание: Перед применением функций аудита запустите утилиту аудита с помощью команды db2audit start.

Можно генерировать записи аудита различных категорий. Обратите внимание на то, что в описании категорий событий, доступных для аудита (смотрите ниже), за именем каждой категории следует одиночное ключевое слово, используемое для идентификации типа категории. Для аудита доступны следующие категории событий:

- Аудит (AUDIT). Генерирует записи при изменении параметров аудита или при обращении к журналу аудита.
- Проверка авторизации (CHECKING). Генерирует записи попыток доступа или управления объектами или функциями DB2 при проверке авторизации.
- Поддержка объектов (OBJMAINT). Генерирует записи при создании или отбрасывании объектов данных.
- Поддержка защиты (SECMAINT). Генерирует записи при предоставлении или отзыве привилегий для объектов и баз данных или полномочий DBADM. Кроме того, записи генерируются при изменении параметров конфигурации защиты менеджера баз данных SYSADM_GROUP, SYSCTRL_GROUP и SYSMAINT_GROUP.
- Управление системой (SYSADMIN). Генерирует записи при выполнении операций, требующих полномочий SYSADM, SYSMAINT ил SYSCTRL.
- Проверка пользователя (VALIDATE). Генерирует записи при аутентификации пользователей или при восстановлении информации защиты системы.

- Контекст операции (CONTEXT). Генерирует записи для вывода контекста операции при выполнении операций базы данных. Эта категория позволяет лучше интерпретировать файл журнала аудита. При использовании поля коррелятора событий журнала группу событий можно связать с одной операцией базы данных. Например, при анализе результатов аудита необходимый контекст может обеспечить оператор SQL для динамического SQL, идентификатор пакета для статического SQL или индикатор типа такой выполняемой операции, как CONNECT.

  Примечание: Оператор SQL, обеспечивающий контекст операции, может оказаться очень длинным, но он будет полностью показан в записи CONTEXT. В результате запись CONTEXT может оказаться очень большой.
- Утилиту аудита можно использовать для успешных событий, для неудачных событий, либо для тех и других.

Для любой операции по базе данных может быть сгенерировано несколько записей. Фактическое число записей, генерируемых и помещаемых в журнал аудита, определяется числом категорий событий, которые нужно записать, заданных в конфигурации утилиты аудита. Кроме того, оно зависит от того, какие события отслеживаются утилитой аудита: успешные события, неудачные события или оба типа событий. По этой причине важно правильно отбирать события для аудита.

Понятия, связанные с данным:

- “Применение утилиты аудита” на стр. 285
- “Форматы записей утилиты аудита (введение)” на стр. 293
- “Рекомендации по работе с утилитой аудита” на стр. 312

Задачи, связанные с данной темой:

- “Управление работой утилиты аудита DB2” на стр. 314

Ссылки, связанные с данной темой:

- “Сценарий применения утилиты аудита” на стр. 288
- “Сообщения утилиты аудита” на стр. 292

Применение утилиты аудита

Утилита аудита записывает возможные для аудита события, включая события, влияющие на экземпляры баз данных. По этой причине утилита аудита представляет собой независимый компонент DB2®, который продолжает работать даже после завершения работы экземпляра DB2. В такой ситуации,

если снова запустить остановленный экземпляр, утилита аудита продолжит аудит событий базы данных на этом экземпляре.

Время записи результатов обработки утилиты в журнал аудита может существенно влиять на производительность баз данных в экземпляре. Записи утилиты аудита могут записываться синхронно или асинхронно относительно событий, для которых генерируются эти записи. Время записей утилиты аудита определяется параметром конфигурации менеджера баз данных *audit_buf_sz*.

Если значение этого параметра равно нулю (0), запись осуществляется синхронно. Событие, генерирующее запись, ожидает, пока эта запись не будет помещена на диск. Ожидание, связанное с каждой записью, приводит к снижению производительности DB2.

При значении параметра *audit_buf_sz* больше нуля запись осуществляется асинхронно. Значение параметра *audit_buf_sz* больше нуля - это число страниц по 4 Кбайта, используемых для внутреннего буфера. Во внутреннем буфере накапливается определенное число записей, а затем вся группа записей записывается на диск. Оператор, сгенерировавший результат события аудита, не должен ждать, пока эта запись будет помещена на диск, и может продолжать обработку.

В асинхронном случае записи аудита могут некоторое время оставаться в незаполненном буфере. Чтобы они не хранились в буфере долго, менеджер баз данных регулярно записывает записи аудита принудительно. Кроме того, буфер аудита можно записать на диск по явному требованию авторизованного пользователя.

Если происходит ошибка, ее последствия при синхронной записи отличаются от последствий при асинхронной записи. В асинхронном режиме можно потерять несколько записей, так как записи аудита перед помещением на диск хранятся в буфере. В синхронном режиме ошибка может привести к потере максимум одной записи аудита.

Значение параметра утилиты аудита ERRORTYPE (тип ошибки) задает способ обработки ошибок DB2 и утилитой аудита. Если утилита аудита активна, и для параметра ERRORTYPE установлено значение `AUDIT`, утилита аудита рассматривается, как любая другая часть DB2. Запись аудита должна записываться (на диск - в синхронном режиме и в буфер аудита - в асинхронном режиме) для события аудита, связанного с успешно обработанным оператором. Если при работе в этом режиме происходит ошибка, программе возвращается отрицательный SQLCODE для оператора, генерирующего записи аудита. Если для параметра типа ошибки установлено значение `NORMAL`, все ошибки db2audit игнорируются, и возвращается SQLCODE данной операции.

В зависимости от API или от оператора SQL и настроек экземпляра DB2 для определенного события может генерироваться одна, ни одной или несколько записей аудита. Например, для оператора SQL UPDATE с подзапросом SELECT может быть сгенерирована одна запись аудита, содержащая результаты проверки авторизации привилегии на таблицу UPDATE, и другая запись, содержащая результаты проверки авторизации привилегии на таблицу SELECT.

Для операторов DML (dynamic data manipulation language - язык динамического управления данными) записи аудита генерируются для каждой проверки авторизации во время подготовки оператора. Повторное использование таких операторов одним и тем же пользователем не будет обрабатываться утилитой аудита, так как в этот момент авторизация не проверяется. Однако если в одну из таблиц каталога, содержащих информацию о привилегиях, вносится изменение, то в следующей единице работы привилегии оператора для кэшируемых операторов динамического SQL проверяются заново, и создается одна или несколько новых записей аудита.

Для пакета, содержащего только статические операторы DML, единственное событие аудита, которое может генерировать запись аудита - это проверка авторизации, выясняющая наличие у пользователя привилегии на выполнение данного пакета. Проверка авторизации и необязательное создание записи аудита, необходимые для операторов статического SQL в пакете, выполняются во время предварительной компиляции или связывания пакета. Выполнение операторов статического SQL в пределах пакета утилитой аудита не обрабатывается. При повторном связывании пакета (либо явно пользователем, либо неявно системой) записи аудита генерируются для проверок авторизации, требуемых операторами статического SQL.

Для операторов, где проверка авторизации проводится во время выполнения оператора (например, для операторов DDL GRANT и REVOKE), записи аудита генерируются при каждом использовании этих операторов.

Примечание: При выполнении DDL номер раздела, записанный для всех событий в записи аудита (кроме событий контекста), будет нулевым (0) независимо от того, какой номер раздела был у оператора.

Понятия, связанные с данным:

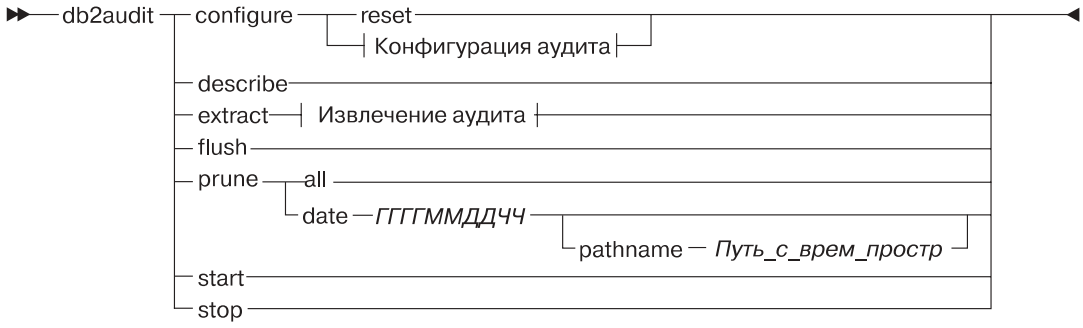
- “Утилита аудита DB2 - Введение” на стр. 283

Ссылки, связанные с данной темой:

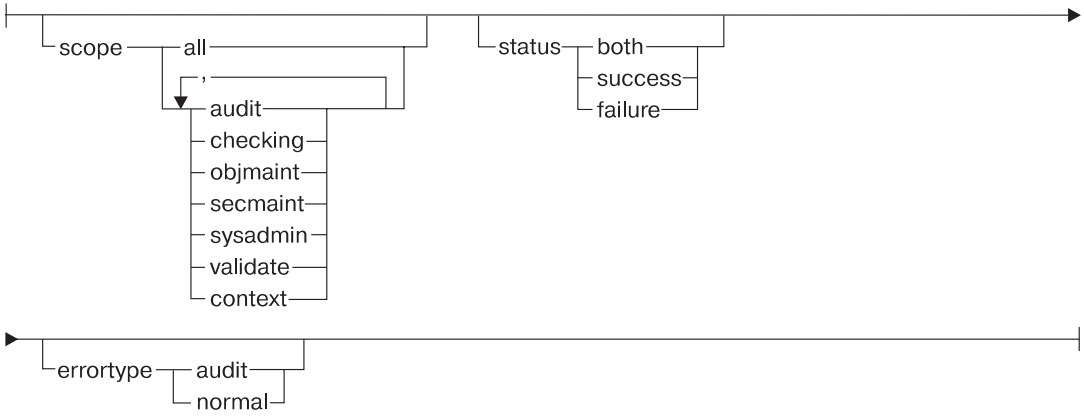
- “Параметр конфигурации Размер буфера аудита - audit_buf_sz” в *Руководство администратора: Производительность*
- “Сценарий применения утилиты аудита” на стр. 288

Сценарий применения утилиты аудита

Посмотрите каждую часть приведенных ниже синтаксических диаграмм - это поможет вам понять, как использовать возможность аудита.



Конфигурация аудита:



Извлечение аудита:

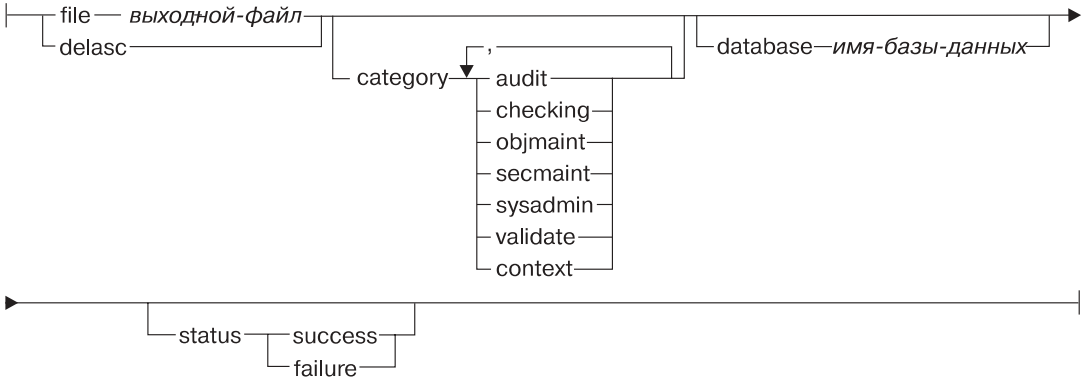


Рисунок 6. Синтаксис DB2AUDIT

Ниже приводится описание и предполагаемое использование каждого параметра:

configure

Этот параметр позволяет изменять файл конфигурации db2audit.cfg в

подкаталоге `security` экземпляра. Этот файл можно изменить даже после завершения работы экземпляра.Изменения файла, выполняемые при активном экземпляре, динамически воздействуют на процесс аудита, выполняемый DB2 по всем разделам. Если утилита аудита запущена, и выполняется аудит событий категории *аудит*, конфигурирование файла конфигурации приводит к созданию записи аудита.

Ниже приводятся возможные действия в файле конфигурации:

- RESET. Это действие восстанавливает начальную конфигурацию файла (в которой SCOPE означает все категории, кроме CONTEXT, для STATUS установлено значение FAILURE, для ERRORTYPE - значение NORMAL, а возможность аудита отключена). Если оригинал файла конфигурации аудита был потерян или поврежден, это действие создает новый файл конфигурации.
- SCOPE. Это действие задает категорию или категории событий для аудита. При этом можно сосредоточить внимание на конкретных событиях и ограничить рост объема журнала. Рекомендуется ограничить число регистрируемых событий и их типы, насколько это возможно, иначе объем журнала аудита будет быстро расти.

  Примечание: Помните, что значение по умолчанию SCOPE означает все категории, кроме CONTEXT, что может служить причиной генерации большого числа записей. Выбор категорий совместно с выбором режима (синхронного или асинхронного) может значительно уменьшить производительность и существенно повысить требования к дисковому пространству.
- STATUS. Это действие задает регистрацию в журнале только успешных событий, только неудачных событий или и тех, и других.

  Примечание: События контекста происходят до определения состояния операции. Поэтому эти события регистрируются в журнале независимо от значения для данного параметра.
- ERRORTYPE. Это действие задает, возвращать ошибки аудита пользователю или игнорировать. Для этого параметра допустимы следующие значения:
  - AUDIT. Все ошибки, включая ошибки утилиты аудита, передаются DB2, и все отрицательные SQLCODE возвращаются в вызывающую программу.
  - NORMAL. Любые ошибки, генерируемые утилитой db2audit, игнорируются, и в программу возвращаются только SQLCODE для ошибок, связанных с выполняемой операцией.

describe

Этот параметр служит для стандартного вывода текущей информации о конфигурации и состояния аудита.

extract Этот параметр позволяет переместить записи из журнала аудита в указанное место назначения. Если не задать дополнительных условий, все записи аудита извлекаются и помещаются в плоский файл отчета. Если параметр " extract" не задан, записи аудита помещаются в файл db2audit.log, расположенный в каталоге `security`. Если *выходной_файл* уже существует, возвращается сообщение об ошибке.

При извлечении могут использоваться следующие допустимые опции:

- FILE. Извлеченные записи аудита помещаются в файл (выходной_файл).
- DELASC. Извлеченные записи аудита переводятся в формат ASCII с ограничителями, удобный для загрузки в реляционные таблицы DB2. Вывод для каждой категории помещается в отдельный файл. Используются следующие имена файлов:
  - audit.del
  - checking.del
  - objmaint.del
  - secmaint.del
  - sysadmin.del
  - validate.del
  - context.del

  В операции извлечения из журнала аудита значение DELASC также позволяет переопределить ограничитель по умолчанию символьной строки аудита ("0xff"). После DELASC DELIMITER можно задать новый ограничитель, который вам хотелось бы использовать для подготовки к загрузке в таблицу, куда будут помещаться записи аудита. Новый ограничитель может быть либо одиночным символом (например !), либо четырехбайтной строкой, представляющей шестнадцатеричное число (например 0xff).
- CATEGORY. Извлекаются записи аудита для заданных категорий событий аудита. Если не задать категории для извлечения, выбираются все категории.
- DATABASE. Извлекаются записи аудита для заданной базы данных. Если не задать базу данных, выбираются все базы данных.
- STATUS. Извлекаются записи аудита для заданного состояния. Если не задать состояние, выбираются все состояния.

flush Этот параметр служит для принудительной записи в журнал всех отложенных записей аудита. Кроме того, если утилита аудита

находится в состоянии ошибки, внутреннее состояние аудита изменяется с “unable to log” (запись невозможна) меняется на “ready to log” (готово к записи).

prune Этот параметр позволяет удалять записи из журнала аудита. Если утилита аудита активна, и для аудита была выбрана категория событий “audit”, после сокращения журнала аудита в него будет помещена запись об этом.

При сокращении могут использоваться следующие допустимые опции:

- ALL. В журнале аудита подлежат удалению все записи.
- DATE ггггммддчч. Пользователь задает удаление из журнала аудита всех записей, появившихся до заданной даты/времени включительно. Пользователь может задать необязательный путь

  `путь`

  который утилита аудита будет использовать как временное пространство во время сокращения журнала аудита. Это временное пространство позволяет сокращать журнал аудита, когда диск расположения журнала переполнен, и на нем недостаточно места для операции сокращения.

start Этот параметр запускает аудит событий утилитой аудита на основе содержания файла db2audit.cfg. При задании этого условия в экземпляре многораздельной DB2 аудит начнется по всем разделам. Если для аудита была выбрана категория событий “audit”, при запуске утилиты аудита запись об этом будет помещена в журнал аудита.

stop Этот параметр останавливает аудит событий утилитой аудита. При задании этого условия в экземпляре многораздельной DB2 аудит будет остановлен на всех разделах. Если для аудита была выбрана категория событий “audit”, при остановке утилиты аудита запись об этом будет помещена в журнал аудита.

Понятия, связанные с данным:

- “Утилита аудита DB2 - Введение” на стр. 283
- “Рекомендации по работе с утилитой аудита” на стр. 312

Ссылки, связанные с данной темой:

- “db2audit - Audit Facility Administrator Tool Command” в *Command Reference*

Сообщения утилиты аудита

SQL1322N Ошибка при записи в файл журнала аудита.

Объяснение: Утилита аудита DB2 обнаружила ошибку при вызове для записи события аудита в файл журнала. В файловой системе, содержащей журнал аудита, недостаточно памяти.

Действия пользователя: Системный администратор должен освободить место в файловой системе или уменьшить журнал аудита, сократив его.

Освободив пространство, воспользуйтесь db2audit для удаления всех данных из памяти и переустановки функции контроля в состояние готовности. Перед сокращением журнала убедитесь, что из него были сделаны необходимые извлечения или была создана его копия, поскольку восстановить удаленные записи нельзя.

sqlcode: -1322

sqlstate: 50830

Понятия, связанные с данным:

- “Утилита аудита DB2 - Введение” на стр. 283

SQL1323N Ошибка при доступе к файлу конфигурации функции аудита.

Объяснение: Не удалось открыть файл конфигурации аудита (db2audit.cfg), либо этот файл содержит неверные данные. Возможные причины этой ошибки: файл db2audit.cfg либо не существует, либо поврежден.

Действия пользователя: Выполните одно из следующих действий:

- Восстановите сохраненную версию файла.
- Переустановите файл конфигурации утилиты аудита, введя команду

  ```
  db2audit reset
  ```

sqlcode: -1323

sqlstate: 57019

Форматы записей утилиты аудита (введение)

При извлечении записи из журнала аудита с использованием опции извлечения DELASC эта запись будет иметь один из показанных в нижеприведенных таблицах форматов. В начале каждой таблицы приводится содержание образца записи. Содержание каждого пункта записи выводится в соответствующей таблице в одной строке. Названия важных пунктов выделяются **жирным шрифтом**. В таких пунктах содержится самая интересная для вас информация.

Примечания:

1. В образцах записей не все поля имеют значения.
2. Некоторые поля, например “Попытка доступа”, хранятся в формате ASCII с ограничителями как битовые образы. Однако в данном плоском файле отчета такие поля будут выводиться как наборы строк, представляющие значения битовых образов.
3. В формат записей о событиях CHECKING, OBJMAINT, SECMAINT, SYSADMIN, VALIDATE и CONTEXT добавлено новое поле “Версия пакета”.

Ссылки, связанные с данной темой:

- “Формат записей аудита для событий AUDIT” на стр. 294

- “Формат записей аудита для событий CHECKING” на стр. 295
- “Формат записей аудита для событий OBJMAINT” на стр. 299
- “Формат записей аудита для событий SECMAINT” на стр. 301
- “Формат записей аудита для событий SYSADMIN” на стр. 306
- “Формат записей аудита для событий VALIDATE” на стр. 309
- “Формат записей аудита для событий CONTEXT” на стр. 310

Сведения о структуре записи аудита

В этом разделе показаны сведения о различных структурах записей аудита.

Формат записей аудита для событий AUDIT

Таблица 7. Формат записей аудита для событий AUDIT

`timestamp=1998-06-24-11.54.05.151232;category=AUDIT;audit event=START;`
`event correlator=0;event status=0;`
`userid=boss;authid=BOSS;`		

ИМЯ	**ФОРМАТ**	**ОПИСАНИЕ**
Timestamp	CHAR(26)	Дата и время события аудита.
Category	CHAR(8)	Категория события аудита. Возможны следующие значения: AUDIT
Audit Event	VARCHAR(32)	Конкретное событие аудита. Возможны следующие значения: CONFIGURE, DB2AUD, EXTRACT, FLUSH, PRUNE, START, STOP и UPDATE_ADMIN_CFG
Event Correlator	INTEGER	Идентификатор коррелятора событий для операции аудита. Может использоваться для идентификации записей аудита, связываемых с отдельным событием.
Event Status	INTEGER	Состояние события аудита, представленное SQLCODE: для успешного события > = 0 Для неудачного события < 0
User ID	VARCHAR(1024)	ID пользователя на момент события.
Authorization ID	VARCHAR(128)	ID авторизации на момент события.

Понятия, связанные с данным:

- “Форматы записей утилиты аудита (введение)” на стр. 293

Формат записей аудита для событий CHECKING

Таблица 8. Формат записей аудита для событий CHECKING

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;audit event=CHECKING_OBJECT;
  event correlator=2;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  object name=FOO;object type=DATABASE;
  access approval reason=DATABASE;access attempted=CONNECT;
```

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Timestamp	CHAR(26)	Дата и время события аудита.
Category	CHAR(8)	Категория события аудита. Возможны следующие значения: CHECKING
Audit Event	VARCHAR(32)	Конкретное событие аудита. Допустимые значения: CHECKING_OBJECT и CHECKING_FUNCTION
Event Correlator	INTEGER	Идентификатор коррелятора событий для операции аудита. Может использоваться для идентификации записей аудита, связываемых с отдельным событием.
Event Status	INTEGER	Состояние события аудита, представленное SQLCODE: для успешного события > = 0 Для неудачного события < 0
Database Name	CHAR(8)	Имя базы данных, для которой было сгенерировано событие. Поле пустое, если было сгенерировано событие аудита уровня экземпляра.
User ID	VARCHAR(1024)	ID пользователя на момент события.
Authorization ID	VARCHAR(128)	ID авторизации на момент события.
Origin Node Number	SMALLINT	Номер узла, на котором произошло событие аудита.
Coordinator Node Number	SMALLINT	Номер узла координатора.
Application ID	VARCHAR (255)	ID программы, используемый на момент события аудита.
Application Name	VARCHAR (1024)	Имя программы, используемое на момент события аудита.
Package Schema	VARCHAR (128)	Схема пакета, используемая на момент события аудита.
Package Name	VARCHAR (128)	Имя пакета, используемое на момент события аудита.
Package Section Number	SMALLINT	Номер раздела в пакете, используемого на момент события аудита.
Object Schema	VARCHAR (128)	Схема объекта, для которого было сгенерировано событие аудита.
Object Name	VARCHAR (128)	Имя объекта, для которого было сгенерировано событие аудита.

Таблица 8. Формат записей аудита для событий CHECKING (продолжение)

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;audit event=CHECKING_OBJECT;
  event correlator=2;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  object name=FOO;object type=DATABASE;
  access approval reason=DATABASE;access attempted=CONNECT;
```

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Object Type	VARCHAR (32)	Тип объекта, для которого было сгенерировано событие аудита. Возможны следующие значения: TABLE, VIEW, ALIAS, FUNCTION, INDEX, PACKAGE, DATA_TYPE, NODEGROUP, SCHEMA, STORED_PROCEDURE, METHOD, BUFFERPOOL, TABLESPACE, EVENT_MONITOR, TRIGGER, DATABASE, INSTANCE, FOREIGN_KEY, PRIMARY_KEY, UNIQUE_CONSTRAINT, CHECK_CONSTRAINT, WRAPPER, SERVER, NICKNAME, USER MAPPING, SERVER OPTION, TRANSFORM, TYPE MAPPING, FUNCTION MAPPING, SUMMARY TABLES и NONE.
Access Approval Reason	CHAR(18)	Указывает причину, по которой был предоставлен доступ для данного события аудита. Допустимые значения показаны в первом списке после данной таблицы.
Access Attempted	CHAR(18)	Указывает тип выполненного обращения. Возможные значения показаны во втором списке после данной таблицы.
Package Version	VARCHAR (64)	Версия пакета, применявшегося в момент возникновения события.

Понятия, связанные с данным:

- “Форматы записей утилиты аудита (введение)” на стр. 293

Ссылки, связанные с данной темой:

- “Список возможных причин предоставления доступа CHECKING” на стр. 296
- “Список возможных типов обращений CHECKING” на стр. 297

Список возможных причин предоставления доступа CHECKING

Ниже приведен список возможных причин предоставления доступа CHECKING:

0x0000000000000001 ACCESS DENIED
Доступ не предоставлен; требование отклонено.

0x0000000000000002 SYSADM
Доступ предоставлен; программа/пользователь обладает полномочиями SYSADM.

0x0000000000000004 SYSCTRL
Доступ предоставлен; программа/пользователь обладает полномочиями SYSCTRL.

0x0000000000000008 SYSMAINT
Доступ предоставлен; программа/пользователь обладает полномочиями SYSMAINT.

0x0000000000000010 DBADM
Доступ предоставлен; программа/пользователь обладает полномочиями DBADM.

0x0000000000000020 DATABASE PRIVILEGE
Доступ предоставлен; программа/пользователь обладает явной привилегией на базу данных.

0x0000000000000040 OBJECT PRIVILEGE
Доступ предоставлен; программа/пользователь обладает явной привилегией на данный объект или функцию.

0x0000000000000080 DEFINER
Доступ предоставлен; данный объект или функция была определена программой или пользователем.

0x0000000000000100 OWNER
Доступ предоставлен; программа/пользователь - владелец данного объекта или функции.

0x0000000000000200 CONTROL
Доступ предоставлен; программа/пользователь обладает привилегией CONTROL на данный объект или функцию.

0x0000000000000400 BIND
Доступ предоставлен; программа/пользователь обладает привилегией связывания на данный объект или функцию.

0x0000000000000800 SYSQUIESCE
Доступ предоставлен; если экземпляр или база данных находится в режиме стабилизации, то программа или пользователь может установить соединение, или подключиться.

Ссылки, связанные с данной темой:

- “Формат записей аудита для событий CHECKING” на стр. 295
- “Список возможных типов обращений CHECKING” на стр. 297

Список возможных типов обращений CHECKING

Ниже приведен список возможных типов обращений CHECKING:

0x0000000000000002 ALTER
Попытка изменить объект.

0x0000000000000004 DELETE
Попытка удалить объект.

0x0000000000000008 INDEX
Попытка использовать индекс.

0x0000000000000010 INSERT
Попытка вставки в объект.

0x0000000000000020 SELECT
Попытка запроса к таблице или производной таблице.

0x0000000000000040 UPDATE
Попытка изменить данные в объекте.

0x0000000000000080 REFERENCE
Попытка установить реляционные связи между объектами.

0x0000000000000100 CREATE
Попытка создать объект.

0x0000000000000200 DROP
Попытка отбросить объект.

0x0000000000000400 CREATEIN
Попытка создать объект в другой схеме.

0x0000000000000800 DROPIN
Попытка отбросить объект в другой схеме.

0x0000000000001000 ALTERIN
Попытка изменить объект в другой схеме.

0x0000000000002000 EXECUTE
Попытка запустить программу, вызвать подпрограмму, создать функцию, источником которой служит подпрограмма (относится только к функциям), либо задать подпрограмму в операторе DDL.

0x0000000000004000 BIND
Попытка связывания или подготовки программы.

0x0000000000008000 SET EVENT MONITOR
Попытка установки переключателей монитора событий.

0x0000000000010000 SET CONSTRAINTS
Попытка установить ограничения на объект.

0x0000000000020000 COMMENT ON
Попытка создать комментарий для объекта.

0x0000000000040000 GRANT
Попытка предоставить привилегии на объект другому ID пользователя.

0x0000000000080000 REVOKE
Попытка отозвать привилегии на объект у ID пользователя.

0x0000000000100000 LOCK
Попытка блокировки объекта.

0x0000000000200000 RENAME
Попытка переименовать объект.

0x0000000000400000 CONNECT
Попытка соединения с объектом.

0x0000000000800000 Member of SYS Group
Попытка доступа или использования элемента группы SYS.

0x0000000001000000 Access All
Попытка выполнить оператор со всеми требуемыми привилегиями на объекты (используется только для DBADM/SYSADM).

0x0000000002000000 Drop All
Попытка отбросить несколько объектов.

0x0000000004000000 LOAD
Попытка загрузить таблицу в табличное пространство.

0x0000000008000000 USE
Попытка создать таблицу в табличном пространстве.

Ссылки, связанные с данной темой:

- “Формат записей аудита для событий CHECKING” на стр. 295
- “Список возможных причин предоставления доступа CHECKING” на стр. 296

Формат записей аудита для событий OBJMAINT

Таблица 9. Формат записей аудита для событий OBJMAINT

```
timestamp=1998-06-24-08.42.41.957524;category=OBJMAINT;audit event=CREATE_OBJECT;
  event correlator=3;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=AUDIT;object type=TABLE;
```

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Timestamp	CHAR(26)	Дата и время события аудита.
Category	CHAR(8)	Категория события аудита. Возможны следующие значения: OBJMAINT
Audit Event	VARCHAR(32)	Конкретное событие аудита. Возможны следующие значения: CREATE_OBJECT, RENAME_OBJECT и DROP_OBJECT

Таблица 9. Формат записей аудита для событий OBJMAINT (продолжение)

```
timestamp=1998-06-24-08.42.41.957524;category=OBJMAINT;audit event=CREATE_OBJECT;
  event correlator=3;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=AUDIT;object type=TABLE;
```

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Event Correlator	INTEGER	Идентификатор коррелятора событий для операции аудита. Может использоваться для идентификации записей аудита, связываемых с отдельным событием.
Event Status	INTEGER	Состояние события аудита, представленное SQLCODE: для успешного события > = 0 Для неудачного события < 0
Database Name	CHAR(8)	Имя базы данных, для которой было сгенерировано событие. Поле пустое, если было сгенерировано событие аудита уровня экземпляра.
User ID	VARCHAR(1024)	ID пользователя на момент события.
Authorization ID	VARCHAR(128)	ID авторизации на момент события.
Origin Node Number	SMALLINT	Номер узла, на котором произошло событие аудита.
Coordinator Node Number	SMALLINT	Номер узла координатора.
Application ID	VARCHAR (255)	ID программы, используемый на момент события аудита.
Application Name	VARCHAR (1024)	Имя программы, используемое на момент события аудита.
Package Schema	VARCHAR (128)	Схема пакета, используемая на момент события аудита.
Package Name	VARCHAR (128)	Имя пакета, используемое на момент события аудита.
Package Section Number	SMALLINT	Номер раздела в пакете, используемого на момент события аудита.
Object Schema	VARCHAR (128)	Схема объекта, для которого было сгенерировано событие аудита.
Object Name	VARCHAR (128)	Имя объекта, для которого было сгенерировано событие аудита.
Object Type	VARCHAR (32)	Тип объекта, для которого было сгенерировано событие аудита. Возможны следующие значения: TABLE, VIEW, ALIAS, FUNCTION, INDEX, PACKAGE, DATA_TYPE, NODEGROUP, SCHEMA, STORED_PROCEDURE, METHOD, BUFFERPOOL, TABLESPACE, EVENT_MONITOR, TRIGGER, DATABASE, INSTANCE, FOREIGN_KEY, PRIMARY_KEY, UNIQUE_CONSTRAINT, CHECK_CONSTRAINT, WRAPPER, SERVER, NICKNAME, USER MAPPING, SERVER OPTION, TRANSFORM, TYPE MAPPING, FUNCTION MAPPING, SUMMARY TABLES и NONE.

Таблица 9. Формат записей аудита для событий OBJMAINT (продолжение)

```
timestamp=1998-06-24-08.42.41.957524;category=OBJMAINT;audit event=CREATE_OBJECT;
  event correlator=3;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=AUDIT;object type=TABLE;
```

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Package Version	VARCHAR (64)	Версия пакета, применявшегося в момент возникновения события.

Понятия, связанные с данным:

- “Утилита аудита DB2 - Введение” на стр. 283

Формат записей аудита для событий SECMAINT

Таблица 10. Формат записей аудита для событий SECMAINT

```
timestamp=1998-06-24-11.57.45.188101;category=SECMAINT;audit event=GRANT;
  event correlator=4;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.boss.980624155728;application name=db2bp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=T1;object type=TABLE;
  grantor=BOSS;grantee=WORKER;grantee type=USER;privilege=SELECT;
```

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Timestamp	CHAR(26)	Дата и время события аудита.
Category	CHAR(8)	Категория события аудита. Возможны следующие значения: SECMAINT
Audit Event	VARCHAR(32)	Конкретное событие аудита. Допустимые значения: GRANT, REVOKE, IMPLICIT_GRANT, IMPLICIT_REVOKE и UPDATE_DBM_CFG.
Event Correlator	INTEGER	Идентификатор коррелятора событий для операции аудита. Может использоваться для идентификации записей аудита, связываемых с отдельным событием.
Event Status	INTEGER	Состояние события аудита, представленное SQLCODE: для успешного события > = 0 Для неудачного события < 0
Database Name	CHAR(8)	Имя базы данных, для которой было сгенерировано событие. Поле пустое, если было сгенерировано событие аудита уровня экземпляра.
User ID	VARCHAR(1024)	ID пользователя на момент события.
Authorization ID	VARCHAR(128)	ID авторизации на момент события.

Таблица 10. Формат записей аудита для событий SECMAINT (продолжение)

```
timestamp=1998-06-24-11.57.45.188101;category=SECMAINT;audit event=GRANT;
  event correlator=4;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.boss.980624155728;application name=db2bp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=T1;object type=TABLE;
  grantor=BOSS;grantee=WORKER;grantee type=USER;privilege=SELECT;
```

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Origin Node Number	SMALLINT	Номер узла, на котором произошло событие аудита.
Coordinator Node Number	SMALLINT	Номер узла координатора.
Application ID	VARCHAR (255)	ID программы, используемый на момент события аудита.
Application Name	VARCHAR (1024)	Имя программы, используемое на момент события аудита.
Package Schema	VARCHAR (128)	Схема пакета, используемая на момент события аудита.
Package Name	VARCHAR (128)	Имя пакета, используемое на момент события аудита.
Package Section Number	SMALLINT	Номер раздела в пакете, используемого на момент события аудита.
Object Schema	VARCHAR (128)	Схема объекта, для которого было сгенерировано событие аудита.
Object Name	VARCHAR (128)	Имя объекта, для которого было сгенерировано событие аудита.
Object Type	VARCHAR (32)	Тип объекта, для которого было сгенерировано событие аудита. Возможны следующие значения: TABLE, VIEW, ALIAS, FUNCTION, INDEX, PACKAGE, DATA_TYPE, NODEGROUP, SCHEMA, STORED_PROCEDURE, METHOD, BUFFERPOOL, TABLESPACE, EVENT_MONITOR, TRIGGER, DATABASE, INSTANCE, FOREIGN_KEY, PRIMARY_KEY, UNIQUE_CONSTRAINT, CHECK_CONSTRAINT, WRAPPER, SERVER, NICKNAME, USER MAPPING, SERVER OPTION, TRANSFORM, TYPE MAPPING, FUNCTION MAPPING, SUMMARY TABLES и NONE.
Grantor	VARCHAR (128)	ID предоставляющего полномочия или привилегии.
Grantee	VARCHAR (128)	ID получателя, которому предоставляется привилегия или полномочия или у которого они отзываются.
Grantee Type	VARCHAR (32)	Тип получателя, для которого предоставляется или отзывается привилегия или полномочия). Допустимые значения: USER, GROUP или BOTH.
Privilege или Authority	CHAR(18)	Указывает тип предоставляемой или отзываемой привилегии или полномочий. Допустимые значения перечислены в списке, приведенном после таблицы.

Таблица 10. Формат записей аудита для событий SECMAINT (продолжение)

```
timestamp=1998-06-24-11.57.45.188101;category=SECMAINT;audit event=GRANT;
  event correlator=4;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.boss.980624155728;application name=db2bp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=T1;object type=TABLE;
  grantor=BOSS;grantee=WORKER;grantee type=USER;privilege=SELECT;
```

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Package Version	VARCHAR (64)	Версия пакета, применявшегося в момент возникновения события.

Понятия, связанные с данным:

- “Форматы записей утилиты аудита (введение)” на стр. 293

Ссылки, связанные с данной темой:

- “Список возможных привилегий или полномочий SECMAINT” на стр. 303

Список возможных привилегий или полномочий SECMAINT

Ниже приведен список возможных привилегий или полномочий SECMAINT:

0x0000000000000001 Control Table
Предоставлена или отозвана привилегия на управление таблицей.

0x0000000000000002 ALTER TABLE
Предоставлена или отозвана привилегия на изменение таблицы.

0x0000000000000004 ALTER TABLE with GRANT
Предоставлена или отозвана привилегия на изменение таблицы с правом предоставлять привилегии.

0x0000000000000008 DELETE TABLE
Предоставлена или отозвана привилегия на удаление таблицы.

0x0000000000000010 DELETE TABLE with GRANT
Предоставлена или отозвана привилегия на отбрасывание таблицы с правом предоставлять привилегии.

0x0000000000000020 Table Index
Предоставлена или отозвана привилегия на индекс.

0x0000000000000040 Table Index with GRANT
Предоставлена или отозвана привилегия на индекс с правом предоставлять привилегии.

0x0000000000000080 Table INSERT
Предоставлена или отозвана привилегия на вставку в таблицу.

0x0000000000000100 Table INSERT with GRANT
Предоставлена или отозвана привилегия на вставку в таблицу с правом предоставлять привилегии.

0x0000000000000200 Table SELECT
Предоставлена или отозвана привилегия выбора таблицы.

0x0000000000000400 Table SELECT with GRANT
Предоставлена или отозвана привилегия выбора таблицы с правом предоставлять привилегии.

0x0000000000000800 Table UPDATE
Предоставлена или отозвана привилегия на изменение таблицы.

0x0000000000001000 Table UPDATE with GRANT
Предоставлена или отозвана привилегия на изменение таблицы с правом предоставлять привилегии.

0x0000000000002000 Table REFERENCE
Предоставлена или отозвана привилегия для ссылок на таблицу.

0x0000000000004000 Table REFERENCE with GRANT
Предоставлена или отозвана привилегия для ссылок на таблицу с правом предоставлять привилегии.

0x0000000000020000 CREATEIN Schema
Предоставлена или отозвана привилегия CREATEIN для схемы.

0x0000000000040000 CREATEIN Schema with GRANT
Предоставлена или отозвана привилегия CREATEIN для схемы с правом предоставлять ее.

0x0000000000080000 DROPIN Schema
Предоставлена или отозвана привилегия DROPIN для схемы.

0x0000000000100000 DROPIN Schema with GRANT
Предоставлена или отозвана привилегия DROPIN для схемы с правом предоставлять ее.

0x0000000000200000 ALTERIN Schema
Предоставлена или отозвана привилегия ALTERIN для схемы.

0x0000000000400000 ALTERIN Schema with GRANT
Предоставлена или отозвана привилегия ALTERIN для схемы с правом предоставлять ее.

0x0000000000800000 DBADM Authority
Предоставлены или отозваны полномочия DBADM.

0x0000000001000000 CREATETAB Authority
Предоставлены или отозваны полномочия создавать таблицы.

0x0000000002000000 BINDADD Authority
Предоставлены или отозваны полномочия на связывание и добавление.

0x0000000004000000 CONNECT Authority
Предоставлены или отозваны полномочия CONNECT.

0x0000000008000000 Create not fenced Authority
Предоставлены или отозваны полномочия на создание неизолированных объектов.

0x0000000010000000 Implicit Schema Authority
Предоставлены или отозваны полномочия на неявные схемы.

0x0000000020000000 Server PASSTHRU
Предоставлена или отозвана привилегия на возможность непосредственной работы, минуя данный сервер (для источника данных базы данных объединения).

0x0000000100000000 Table Space USE
Предоставлена или отозвана привилегия создавать таблицы в табличном пространстве.

0x0000000200000000 Table Space USE with GRANT
Предоставлена или отозвана привилегия создавать таблицы в табличном пространстве с правом предоставлять привилегии.

0x0000000400000000 Column UPDATE
Предоставлена или отозвана привилегия на изменение одного или нескольких конкретных столбцов таблицы.

0x0000000800000000 Column UPDATE with GRANT
Предоставлена или отозвана привилегия на изменение одного или нескольких конкретных столбцов таблицы с правом предоставлять привилегии.

0x0000001000000000 Column REFERENCE
Предоставлена или отозвана привилегия для ссылок на один или несколько конкретных столбцов таблицы.

0x0000002000000000 Column REFERENCE with GRANT
Предоставлена или отозвана привилегия для ссылок на один или несколько конкретных столбцов таблицы с правом предоставлять привилегии.

0x0000004000000000 LOAD Authority
Предоставлены или отозваны полномочия на загрузку.

0x0000008000000000 Package BIND
Предоставлена или отозвана привилегия BIND для пакета.

0x0000010000000000 Package BIND with GRANT
Предоставлена или отозвана привилегия BIND для пакета с правом предоставлять эту привилегию.

0x0000020000000000 EXECUTE
Предоставлена или отозвана привилегия EXECUTE для пакета или подпрограммы.

0x0000040000000000 EXECUTE with GRANT
Предоставлена или отозвана привилегия EXECUTE для пакета или подпрограммы с правом предоставлять эту привилегию.

0x0000080000000000 EXECUTE IN SCHEMA
Предоставлена или отозвана привилегия EXECUTE для всех подпрограмм схемы.

0x0000100000000000 EXECUTE IN SCHEMA with GRANT
Предоставлена или отозвана привилегия EXECUTE для всех подпрограмм схемы с правом предоставлять эту привилегию.

0x0000020000000000 EXECUTE IN TYPE
Предоставлена или отозвана привилегия EXECUTE для всех подпрограмм с типом.

0x0000400000000000 EXECUTE IN TYPE with GRANT
Предоставлена или отозвана привилегия EXECUTE для всех подпрограмм с типом и правом предоставлять эту привилегию.

0x0000080000000000 CREATE EXTERNAL ROUTINE
Предоставлена или отозвана привилегия CREATE EXTERNAL ROUTINE.

0x0001000000000000 QUIESCE_CONNECT
Предоставлена или отозвана привилегия QUIESCE_CONNECT.

Ссылки, связанные с данной темой:

- “Формат записей аудита для событий SECMAINT” на стр. 301

Формат записей аудита для событий SYSADMIN

Таблица 11. Формат записей аудита для событий SYSADMIN

`timestamp=1998-06-24-11.54.04.129923;`**`category=SYSADMIN;audit event=DB2AUDIT;`**
`event correlator=1;event status=0;`
`userid=boss;authid=BOSS;`
`application id=*LOCAL.boss.980624155404;application name=db2audit;`

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Timestamp	CHAR(26)	Дата и время события аудита.
Category	CHAR(8)	Категория события аудита. Возможны следующие значения: SYSADMIN

Таблица 11. Формат записей аудита для событий SYSADMIN (продолжение)

```
timestamp=1998-06-24-11.54.04.129923;category=SYSADMIN;audit event=DB2AUDIT;
  event correlator=1;event status=0;
  userid=boss;authid=BOSS;
  application id=*LOCAL.boss.980624155404;application name=db2audit;
```

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Audit Event	VARCHAR(32)	Конкретное событие аудита. Допустимые значения перечислены в списке, приведенном после таблицы.
Event Correlator	INTEGER	Идентификатор коррелятора событий для операции аудита. Может использоваться для идентификации записей аудита, связываемых с отдельным событием.
Event Status	INTEGER	Состояние события аудита, представленное SQLCODE: для успешного события > = 0 Для неудачного события < 0
Database Name	CHAR(8)	Имя базы данных, для которой было сгенерировано событие. Поле пустое, если было сгенерировано событие аудита уровня экземпляра.
User ID	VARCHAR(1024)	ID пользователя на момент события.
Authorization ID	VARCHAR(128)	ID авторизации на момент события.
Origin Node Number	SMALLINT	Номер узла, на котором произошло событие аудита.
Coordinator Node Number	SMALLINT	Номер узла координатора.
Application ID	VARCHAR (255)	ID программы, используемый на момент события аудита.
Application Name	VARCHAR (1024)	Имя программы, используемое на момент события аудита.
Package Schema	VARCHAR (128)	Схема пакета, используемая на момент события аудита.
Package Name	VARCHAR (128)	Имя пакета, используемое на момент события аудита.
Package Section Number	SMALLINT	Номер раздела в пакете, используемого на момент события аудита.
Package Version	VARCHAR (64)	Версия пакета, применявшегося в момент возникновения события.

Понятия, связанные с данным:

- “Форматы записей утилиты аудита (введение)” на стр. 293

Ссылки, связанные с данной темой:

- “Список возможных событий аудита SYSADMIN” на стр. 308

Список возможных событий аудита SYSADMIN

Ниже приведен список возможных событий аудита SYSADMIN:

Таблица 12. События аудита SYSADMIN

START_DB2
STOP_DB2
CREATE_DATABASE
DROP_DATABASE
UPDATE_DBM_CFG
UPDATE_DB_CFG
CREATE_TABLESPACE
DROP_TABLESPACE
ALTER_TABLESPACE
RENAME_TABLESPACE
CREATE_NODEGROUP
DROP_NODEGROUP
ALTER_NODEGROUP
CREATE_BUFFERPOOL
DROP_BUFFERPOOL
ALTER_BUFFERPOOL
CREATE_EVENT_MONITOR
DROP_EVENT_MONITOR
ENABLE_MULTIPAGE
MIGRATE_DB_DIR
DB2TRC
DB2SET
ACTIVATE_DB
ADD_NODE
BACKUP_DB
CATALOG_NODE
CATALOG_DB
CATALOG_DCS_DB
CHANGE_DB_COMMENT
DEACTIVATE_DB
DROP_NODE_VERIFY
FORCE_APPLICATION
GET_SNAPSHOT
LIST_DRDA_INDOUBT_TRANSACTIONS
MIGRATE_DB
RESET_ADMIN_CFG
RESET_DB_CFG
RESET_DBM_CFG
RESET_MONITOR
RESTORE_DB
ROLLFORWARD_DB
SET_RUNTIME_DEGREE
SET_TABLESPACE_CONTAINERS
UNCATALOG_DB
UNCATALOG_DCS_DB
UNCATALOG_NODE
UPDATE_ADMIN_CFG
UPDATE_MON_SWITCHES
LOAD_TABLE
DB2AUDIT
SET_APPL_PRIORITY
CREATE_DB_AT_NODE
KILLDBM
MIGRATE_SYSTEM_DIRECTORY
DB2REMOT
DB2AUD
MERGE_DBM_CONFIG_FILE
UPDATE_CLI_CONFIGURATION
OPEN_TABLESPACE_QUERY
SINGLE_TABLESPACE_QUERY
CLOSE_TABLESPACE_QUERY
FETCH_TABLESPACE
OPEN_CONTAINER_QUERY
FETCH_CONTAINER_QUERY
CLOSE_CONTAINER_QUERY
GET_TABLESPACE_STATISTICS
DESCRIBE_DATABASE
ESTIMATE_SNAPSHOT_SIZE
READ_ASYNC_LOG_RECORD
PRUNE_RECOVERY_HISTORY
UPDATE_RECOVERY_HISTORY
QUIESCE_TABLESPACE
UNLOAD_TABLE
UPDATE_DATABASE_VERSION
CREATE_INSTANCE
DELETE_INSTANCE
SET_EVENT_MONITOR
GRANT_DBADM
REVOKE_DBADM
GRANT_DB_AUTHORITIES
REVOKE_DB_AUTHORITIES
REDIST_NODEGROUP

Ссылки, связанные с данной темой:

• “Формат записей аудита для событий SYSADMIN” на стр. 306

Формат записей аудита для событий VALIDATE

Таблица 13. Формат записей аудита для событий VALIDATE

```
timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;audit event=CHECK_GROUP_MEMBERSHIP;
  event correlator=2;event status=-1092;
  database=FOO;userid=boss;authid=BOSS;execution id=newton;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  auth type=SERVER;
```

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Timestamp	CHAR(26)	Дата и время события аудита.
Category	CHAR(8)	Категория события аудита. Возможны следующие значения: VALIDATE
Audit Event	VARCHAR(32)	Конкретное событие аудита. Допустимые значения: GET_GROUPS, GET_USERID, AUTHENTICATE_PASSWORD и VALIDATE_USER.
Event Correlator	INTEGER	Идентификатор коррелятора событий для операции аудита. Может использоваться для идентификации записей аудита, связываемых с отдельным событием.
Event Status	INTEGER	Состояние события аудита, представленное SQLCODE: для успешного события > = 0 Для неудачного события < 0
Database Name	CHAR(8)	Имя базы данных, для которой было сгенерировано событие. Поле пустое, если было сгенерировано событие аудита уровня экземпляра.
User ID	VARCHAR(1024)	ID пользователя на момент события.
Authorization ID	VARCHAR(128)	ID авторизации на момент события.
Execution ID	VARCHAR(1024)	ID выполнения, используемый на момент события аудита.
Origin Node Number	SMALLINT	Номер узла, на котором произошло событие аудита.
Coordinator Node Number	SMALLINT	Номер узла координатора.
Application ID	VARCHAR (255)	ID программы, используемый на момент события аудита.
Application Name	VARCHAR (1024)	Имя программы, используемое на момент события аудита.
Authentication Type	VARCHAR (32)	Тип аутентификации на момент события аудита.
Package Schema	VARCHAR (128)	Схема пакета, используемая на момент события аудита.
Package Name	VARCHAR (128)	Имя пакета, используемое на момент события аудита.
Package Section Number	SMALLINT	Номер раздела в пакете, используемого на момент события аудита.

Таблица 13. Формат записей аудита для событий VALIDATE (продолжение)

```
timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;audit event=CHECK_GROUP_MEMBERSHIP;
  event correlator=2;event status=-1092;
  database=FOO;userid=boss;authid=BOSS;execution id=newton;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  auth type=SERVER;
```

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Package Version	VARCHAR (64)	Версия пакета, применявшегося в момент возникновения события.

Понятия, связанные с данным:

- “Форматы записей утилиты аудита (введение)” на стр. 293

Формат записей аудита для событий CONTEXT

Таблица 14. Формат записей аудита для событий CONTEXT

```
timestamp=1998-06-24-08.42.41.476840;category=CONTEXT;audit event=EXECUTE_IMMEDIATE;
  event correlator=3;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SQLC28A1;
  package section=203;text=create table audit(c1 char(10), c2 integer);
```

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Timestamp	CHAR(26)	Дата и время события аудита.
Category	CHAR(8)	Категория события аудита. Возможны следующие значения: CONTEXT
Audit Event	VARCHAR(32)	Конкретное событие аудита. Допустимые значения перечислены в списке, приведенном после таблицы.
Event Correlator	INTEGER	Идентификатор коррелятора событий для операции аудита. Может использоваться для идентификации записей аудита, связываемых с отдельным событием.
Database Name	CHAR(8)	Имя базы данных, для которой было сгенерировано событие. Поле пустое, если было сгенерировано событие аудита уровня экземпляра.
User ID	VARCHAR(1024)	ID пользователя на момент события.
Authorization ID	VARCHAR(128)	ID авторизации на момент события.
Origin Node Number	SMALLINT	Номер узла, на котором произошло событие аудита.
Coordinator Node Number	SMALLINT	Номер узла координатора.
Application ID	VARCHAR (255)	ID программы, используемый на момент события аудита.

Таблица 14. Формат записей аудита для событий CONTEXT (продолжение)

```
timestamp=1998-06-24-08.42.41.476840;category=CONTEXT;audit event=EXECUTE_IMMEDIATE;
  event correlator=3;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SQLC28A1;
  package section=203;text=create table audit(c1 char(10), c2 integer);
```

ИМЯ	ФОРМАТ	ОПИСАНИЕ
Application Name	VARCHAR (1024)	Имя программы, используемое на момент события аудита.
Package Schema	VARCHAR (128)	Схема пакета, используемая на момент события аудита.
Package Name	VARCHAR (128)	Имя пакета, используемое на момент события аудита.
Package Section Number	SMALLINT	Номер раздела в пакете, используемого на момент события аудита.
Statement Text (оператор)	CLOB (32 Кбайта)	Текст оператора SQL, если он есть. Пустое значение, если текст оператора SQL недоступен.
Package Version	VARCHAR (64)	Версия пакета, применявшегося в момент возникновения события.

Понятия, связанные с данным:

- “Форматы записей утилиты аудита (введение)” на стр. 293

Ссылки, связанные с данной темой:

- “Список возможных событий аудита CONTEXT” на стр. 311

Список возможных событий аудита CONTEXT

Ниже приведен список возможных событий аудита CONTEXT:

Таблица 15. События аудита CONTEXT

CONNECT	SET_APPL_PRIORITY
CONNECT_RESET	RESET_DB_CFG
ATTACH	GET_DB_CFG
DETACH	GET_DFLT_CFG
DARI_START	UPDATE_DBM_CFG
DARI_STOP	SET_MONITOR
BACKUP_DB	GET_SNAPSHOT
RESTORE_DB	ESTIMATE_SNAPSHOT_SIZE
ROLLFORWARD_DB	RESET_MONITOR
OPEN_TABLESPACE_QUERY	OPEN_HISTORY_FILE
FETCH_TABLESPACE	CLOSE_HISTORY_FILE
CLOSE_TABLESPACE_QUERY	FETCH_HISTORY_FILE
OPEN_CONTAINER_QUERY	SET_RUNTIME_DEGREE
CLOSE_CONTAINER_QUERY	UPDATE_AUDIT
FETCH_CONTAINER_QUERY	DBM_CFG_OPERATION
SET_TABLESPACE_CONTAINERS	DISCOVER
GET_TABLESPACE_STATISTIC	OPEN_CURSOR
READ_ASYNC_LOG_RECORD	CLOSE_CURSOR
QUIESCE_TABLESPACE	FETCH_CURSOR
LOAD_TABLE	EXECUTE
UNLOAD_TABLE	EXECUTE_IMMEDIATE
UPDATE_RECOVERY_HISTORY	PREPARE
PRUNE_RECOVERY_HISTORY	DESCRIBE
SINGLE_TABLESPACE_QUERY	BIND
LOAD_MSG_FILE	REBIND
UNQUIESCE_TABLESPACE	RUNSTATS
ENABLE_MULTIPAGE	REORG
DESCRIBE_DATABASE	REDISTRIBUTE
DROP_DATABASE	COMMIT
CREATE_DATABASE	ROLLBACK
ADD_NODE	REQUEST_ROLLBACK
FORCE_APPLICATION	IMPLICIT_REBIND

Ссылки, связанные с данной темой:

- “Формат записей аудита для событий CONTEXT” на стр. 310

Рекомендации по работе с утилитой аудита

В большинстве случаев при работе с событиями CHECKING в записи аудита в поле тип объекта выводится объект, для которого проверяется наличие требуемой привилегии или полномочий у ID пользователя, пытающегося получить доступ к объекту. Например, если пользователь пытается изменить (ALTER) таблицу, добавив в нее столбец, в записи аудита события CHECKING

будет указано, что предпринят доступ “ALTER” и тип проверяемого объекта - “TABLE” (таблица, а не столбец, поскольку должны проверяться привилегии на таблицу).

Однако если в ходе проверки для ID пользователя проверяется наличие полномочий на создание (CREATE), связывание (BIND) или удаление объекта базы данных, в поле типа объекта будет задан создаваемый, связываемый или отбрасываемый объект, а не сама база данных (хотя проверка и производится для базы данных).

При создании индекса на таблице требуется привилегия на создание индекса, поэтому в записи события аудита CHECKING типом предпринимаемого доступа будет “index”, а не “create”.

При связывании уже существующего пакета создается запись аудита события OBJMAINT для отбрасывания (DROP) пакета, а затем другая запись аудита события OBJMAINT - для создания (CREATE) новой копии базы данных.

DDL (Data Definition Language - язык определения данных) SQL может генерировать события OBJMAINT или SECMAINT, которые записываются в журнал как успешные. Однако ошибка, последовавшая после регистрации события в журнале, может привести к откату (ROLLBACK). В результате объект не будет создан, либо действие GRANT или REVOKE не будет выполнено. В этом случае имеет смысл использовать события CONTEXT. Записи аудита события CONTEXT, особенно оператор в конце события, укажут, чем завершилась предпринятая операция.

При получении записей аудита в формате ASCII с разделителями, пригодном для загрузки в реляционную таблицу DB2®, необходимо указать ограничитель, используемый в текстовом поле оператора. Это можно сделать при извлечении файла ASCII с ограничителями, воспользовавшись следующей командой:

```
db2audit extract delasc delimiter <разделитель загрузки>
```

Разделитель загрузки может быть одиночным символом (например ") или четырехбайтной строкой, представляющей шестнадцатеричное значение (например “0xff”). Примеры допустимых команд:

```
db2audit extract delasc
db2audit extract delasc delimiter !
db2audit extract delasc delimiter 0xff
```

Если для извлечения используется не разделитель загрузки по умолчанию (“"”), команду LOAD надо использовать с опцией MODIFIED BY. Приведем конкретный пример команды LOAD с использованием в качестве разделителя “0xff”:

```
db2 load from context.del of del modified by chardel0xff replace into ...
```

В этой команде разделитель по умолчанию символьной строки загрузки заменяется разделителем “0xff”.

Понятия, связанные с данным:

- “Форматы записей утилиты аудита (введение)” на стр. 293

Ссылки, связанные с данной темой:

- “Сценарий применения утилиты аудита” на стр. 288

Управление работой утилиты аудита DB2

Процедура:

В обсуждении управления работой утилиты аудита будет использоваться простой сценарий: пользователь *newton* запускает программу *testapp*, которая соединяется и создает таблицу. Эта программа используется во всех рассматриваемых ниже примерах.

Начнем с примера крайней ситуации: вы решили провести ревизию всех успешных и неудачных событий аудита и для этого конфигурируете утилиту аудита следующим образом:

```
db2audit configure scope all status both
```

Примечание: Эта команда создает записи аудита для каждого возможного события аудита. В результате в журнал аудита записывается большое количество записей, что уменьшает производительность менеджера баз данных. Этот крайний случай рассмотрен здесь только в качестве примера; в обычном случае утилиту аудита не рекомендуется настраивать с помощью этой команды.

После запуска утилиты аудита в этой конфигурации (командой “db2audit start”) и последующего запуска программы *testapp* генерируются и помещаются в журнал аудита приведенные ниже записи. Если извлечь записи аудита из журнала, для двух действий, выполненных программой, вы увидите следующие записи:

Действие
Тип созданной записи

CONNECT

```
timestamp=1998-06-24-08.42.10.555345;category=CONTEXT;
audit event=CONNECT;event correlator=2;database=FOO;
application id=*LOCAL.newton.980624124210;
application name=testapp;

timestamp=1998-06-24-08.42.10.944374;category=VALIDATE;
audit event=AUTHENTICATION;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;execution id=newton;
```

```
application id=*LOCAL.newton.980624124210;application name=testapp;
auth type=SERVER;

timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
имя программы=testapp;тип аутентификации=SERVER;

timestamp=1998-06-24-08.42.11.561187;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
имя программы=testapp;тип аутентификации=SERVER;

timestamp=1998-06-24-08.42.11.594620;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
имя программы=testapp;тип аутентификации=SERVER;

timestamp=1998-06-24-08.42.11.622984;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
object name=FOO;object type=DATABASE;access approval reason=DATABASE;
access attempted=CONNECT;

timestamp=1998-06-24-08.42.11.801554;category=CONTEXT;
audit event=COMMIT;event correlator=2;database=FOO;userid=boss;
authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;

timestamp=1998-06-24-08.42.41.450975;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;object schema=NULLID;
object name=SQLC28A1;object type=PACKAGE;
access approval reason=OBJECT;access attempted=EXECUTE;
```

CREATE TABLE

```
timestamp=1998-06-24-08.42.41.476840;category=CONTEXT;
audit event=EXECUTE_IMMEDIATE;event correlator=3;database=FOO;
userid=boss;authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;package schema=NULLID;package name=SQLC28A1;
package section=203;text=create table audit(c1 char(10), c2 integer);

timestamp=1998-06-24-08.42.41.539692;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;
access approval reason=DATABASE;access attempted=CREATE;
```

```
timestamp=1998-06-24-08.42.41.570876;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object name=BOSS;object type=SCHEMA;access approval reason=DATABASE;
access attempted=CREATE;

timestamp=1998-06-24-08.42.41.957524;category=OBJMAINT;
audit event=CREATE_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;

timestamp=1998-06-24-08.42.42.018900;category=CONTEXT;
audit event=COMMIT;event correlator=3;database=FOO;userid=boss;
authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;package schema=NULLID;package name=SQLC28A1;
```

Как видите, при такой конфигурации, когда запрашиваются все возможные события и типы объектов, генерируется большое число записей.

В большинстве случаев в конфигурации утилиты аудита предусматривается более конкретный просмотр контролируемых событий. Например, можно провести аудит только тех событий, которые завершаются неудачно. Для этого утилиту аудита можно настроить следующим образом:

```
db2audit configure scope audit,checking,objmaint,secmaint,sysadmin,
   validate status failure
```

Примечание: Такая конфигурация устанавливается изначально и при сбросе конфигурации.

После запуска утилиты аудита в этой конфигурации и последующего запуска программы *testapp* генерируются и помещаются в журнал аудита приведенные ниже записи. (Предполагается, что программа *testapp* ранее не была запущена.) Если извлечь записи аудита из журнала, для двух действий, выполненных программой, вы увидите следующие записи:

Действие
Тип созданной записи

CONNECT

```
timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
имя программы=testapp;тип аутентификации=SERVER;

timestamp=1998-06-24-08.42.11.561187;category=VALIDATE;
```

```
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
имя программы=testapp;тип аутентификации=SERVER;

timestamp=1998-06-24-08.42.11.594620;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;
```

CREATE TABLE

```
(нет)
```

При такой конфигурации, когда запрашиваются все возможные события аудита (кроме CONTEXT), но только для событий, завершающихся неудачно, генерируется гораздо меньше записей. Путем настройки утилиты аудита можно изменять тип и характер создаваемых записей аудита.

Утилита аудита позволяет создать записи аудита, когда будут контролироваться только успешно предоставленные привилегии для объекта. В этом случае утилиту аудита можно сконфигурировать так:

```
db2audit configure scope checking status success
```

После запуска утилиты аудита в этой конфигурации и последующего запуска программы *testapp* генерируются и помещаются в журнал аудита приведенные ниже записи. (Предполагается, что программа *testapp* ранее не была запущена.) Если извлечь записи аудита из журнала, для двух действий, выполненных программой, вы увидите следующие записи:

Действие

Тип созданной записи

CONNECT

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;

timestamp=1998-06-24-08.42.41.450975;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;object schema=NULLID;
object name=SQLC28A1;object type=PACKAGE;
access approval reason=OBJECT;access attempted=EXECUTE;

timestamp=1998-06-24-08.42.41.539692;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
```

```
object schema=BOSS;object name=AUDIT;object type=TABLE;
access approval reason=DATABASE;access attempted=CREATE;

timestamp=1998-06-24-08.42.41.570876;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object name=BOSS;object type=SCHEMA;access approval reason=DATABASE;
access attempted=CREATE;
```

CREATE TABLE

```
(нет)
```

Понятия, связанные с данным:

- “Форматы записей утилиты аудита (введение)” на стр. 293

Ссылки, связанные с данной темой:

- “Сценарий применения утилиты аудита” на стр. 288

Часть 3. Приложения

Приложение A. Правила именования

Правила именования

Если не сказано иного, все имена могут содержать следующие символы:

- Буквы от A до Z. В большинстве имен буквы преобразуются из строчных в прописные.
- Цифры от 0 до 9
- @, #, $ и _ (подчеркивание)

Имена не должны начинаться с цифры или символа подчеркивания.

Не используйте зарезервированные слова SQL для имен таблиц, производных таблиц, столбцов, индексов или ID авторизации.

Есть и другие специальные символы, которые могут рассматриваться по-разному в разных операционных системах, откуда вы работаете с DB2. Даже если они работают, нет гарантии, что они будут работать и дальше. Использование этих дополнительных специальных символов в именах объектов базы данных не рекомендуется.

Кроме того, ознакомьтесь с правилами именования рабочих станций, а также правилами именования, действующими в среде NLS и среде Unicode.

Понятия, связанные с данным:

- “Общие правила именования объектов и пользователей” на стр. 237
- “Правила именования объектов DB2” на стр. 321
- “Правила именования рабочих станций” на стр. 326
- “Правила именования пользователей, групп и ID пользователей” на стр. 324
- “Правила именования объектов базы данных объединения” на стр. 325

Правила именования объектов DB2

На все объекты распространяются общие правила именования. Кроме того, для некоторых объектов есть дополнительные ограничения, описанные ниже.

Таблица 16. Правила именования баз данных, алиасов баз данных и экземпляров

Объекты	Рекомендации
• Базы данных • Алиасы баз данных • Экземпляры	• Имена баз данных должны быть уникальными в пределах того расположения, в котором они занесены в каталог. В реализациях DB2 для систем на базе UNIX расположение представляет собой каталог, а в Windows® - логический диск. • Имена алиасов баз данных должны быть уникальными в пределах системного каталога баз данных. При создании новой базы данных ее алиасом по умолчанию считается имя базы данных. Это означает, что нельзя создать базу данных, имя которой совпадало бы с существующим алиасом, даже если базы данных с этим именем не существует. • Длина имен баз данных, алиасов баз данных и экземпляров не может быть больше 8 байт. • В Windows NT, Windows 2000, Windows XP и Windows .NET имена экземпляров не должны совпадать с именами служб. **Примечание:** Во избежание возникновения конфликтов не используйте специальные символы @, # и $ в имени базы данных, если вы планируете применять эту базу данных в сетевой среде. Кроме того, поскольку эти символы имеются не на всех клавиатурах, не используйте их, если с базой данных будут работать на другом языке.

Таблица 17. Правила именования объектов баз данных

Объекты	Рекомендации
• Алиасы • Пулы буферов • Столбцы • Мониторы событий • Индексы • Методы • Группы узлов • Пакеты • Версии пакетов • Схемы • Хранимые процедуры • Таблицы • Табличные пространства • Триггеры • Пользовательские функции • Пользовательские типы • Производные таблицы	Могут содержать до 18 байтов, *кроме*: • Имен таблиц (в том числе производных, таблиц сводок, алиасов и внутриоператорных имен), которые могут содержать до 128 байтов • Имен пакетов, которые могут содержать до 8 байт • Имен схем, которые могут содержать до 30 байт • Версий пакетов, которые могут содержать до 64 байт • Имена объектов могут также содержать: – допустимые символы национальных алфавитов (например, ö) – многобайтные символы, за исключением многобайтных пробелов (в многобайтных средах) • Имена и версии пакетов могут содержать точки (.), дефисы (-) и двоеточия (:).

Понятия, связанные с данным:

• “Правила именования” на стр. 321

Идентификаторы и имена объектов с ограничителями

Можно использовать ключевые слова. Если ключевое слово используется в контексте, где его можно принять за ключевое слово SQL, для него надо использовать ограничители.

Используя идентификаторы с ограничителями, можно создать объекты, имена которых нарушают приведенные правила; однако при последующем использовании такого объекта могут возникнуть ошибки. Например, если создать столбец, в имя которого входят знаки + или −, и впоследствии использовать этот столбец в индексе, при попытке реорганизации таблицы возникнут трудности.

Понятия, связанные с данным:

- “Правила именования” на стр. 321

Правила именования пользователей, групп и ID пользователей

Таблица 18. Правила именования пользователей, групп и ID пользователей

Объекты	Рекомендации
• Имена групп • Имена пользователей • ID пользователей	• Имена групп могут содержать до 8 байтов. • ID пользователей в системах на основе UNIX могут содержать до 8 символов. • Имена пользователей в Windows® могут содержать до 30 символов. В Windows NT, Windows 2000, Windows XP и Windows .NET имена должны содержать не более 20 символов. • Если не применяется аутентификация типа Client, то при подключении клиентов, отличных от 32-разрядных систем Windows, к системам Windows NT, Windows 2000, Windows XP и Windows .NET разрешено указывать имена пользователей, содержащие более 8 символов, если имя и пароль заданы явно. • Имена и ID не должны: – Совпадать с именами USERS, ADMINS, GUESTS, PUBLIC, LOCAL и другими зарезервированными словами SQL – Начинаться с IBM, SQL или SYS. – Содержать символы национальных алфавитов.

Примечания:

1. В некоторых операционных системах ID пользователей и пароли регистрозависимы. Посмотрите в документации правила для вашей операционной системы.
2. ID авторизации, возвращаемый успешно выполненным оператором CONNECT или ATTACH, усекается до 8 символов. К ID авторизации добавляется многоточие (...), а поля SQLWARN содержат предупреждения, указывающие на усечение.

Понятия, связанные с данным:

- “Правила именования” на стр. 321

- “Правила именования объектов базы данных объединения” на стр. 325

Правила именования объектов базы данных объединения

Таблица 19. Правила именования объектов базы данных объединения

Объекты	Рекомендации
• Отображения функций • Спецификации индексов • Псевдонимы • Серверы • Отображения типов • Отображения пользователей • Оболочки	• Псевдонимы, имена отображений, спецификаций индексов, серверов и оболочек не должны быть длиннее 128 байтов. • Опции сервера и псевдонима, а также значения опций не должны быть длиннее 255 байтов. • Имена объектов баз данных объединения могут также содержать: – Допустимые буквы национальных алфавитов (например, ö) – Многобайтовые символы, за исключением многобайтовых пробелов (в среде с поддержкой наборов многобайтовых символов)

Понятия, связанные с данным:

- “Правила именования” на стр. 321

Дополнительная информация об именах схем

- Имена схем пользовательских типов не могут быть длиннее 8 байтов.
- Следующие имена схем зарезервированы и использовать их нельзя: SYSCAT, SYSFUN, SYSIBM, SYSSTAT.
- Во избежание возможных проблем с перенастройкой не следует использовать имена схем, начинающиеся с SYS. менеджер баз данных не позволит вам создавать триггеры, пользовательские типы и пользовательские функции, имена схем которых начинаются с SYS.
- Не рекомендуется использовать имя схемы SESSION. К этой схеме должны относиться объявленные временные таблицы. Поэтому возможна ситуация, когда программа объявит временную таблицу с именем, идентичным имени одной из постоянных таблиц, что может привести к сложностям в работе. Не используйте схему SESSION, если вы не работаете с объявленными временными таблицами.

Понятия, связанные с данным:

- “Правила именования” на стр. 321

Дополнительная информация о паролях

Вам может потребоваться менять пароли. Обычно такие действия выполняются на сервере, но многие пользователи слабо знакомы с работой в среде сервера, и для них подобная задача может представлять значительные трудности. В DB2® UDB предусмотрена возможность обновлять и изменять пароли на клиенте. Например, DB2 for OS/390® версии 5 поддерживает следующий способ изменения пароля пользователя. Если вы получили сообщение SQL1404N “Срок действия пароля истек”, измените пароль при помощи оператора CONNECT так:

```
CONNECT TO <база_данных> USER <id_пользователя> USING <пароль>
   NEW <новый_пароль> CONFIRM <новый_пароль>
```

Кроме того, пароль можно изменить с помощью окна “ Изменение пароля” Ассистента конфигурирования DB2 (CA).

Понятия, связанные с данным:

- “Правила именования” на стр. 321
- “Правила именования объектов DB2” на стр. 321
- “Правила именования рабочих станций” на стр. 326
- “Правила именования пользователей, групп и ID пользователей” на стр. 324
- “Правила именования объектов базы данных объединения” на стр. 325
- “Идентификаторы и имена объектов с ограничителями” на стр. 323
- “Дополнительная информация об именах схем” на стр. 325

Правила именования рабочих станций

Имя рабочей станции задает имя NetBIOS для сервера базы данных, клиента базы данных или DB2® Personal Edition, расположенного на локальной рабочей станции. Это имя хранится в файле конфигурации менеджера баз данных. Имя рабочей станции также обозначается как *nname*.

Кроме этого, задаваемое имя:

- Должно содержать от 1 до 8 символов
- Не должно содержать символов &, # или @
- Должно быть уникальным в сети

В многораздельной системе баз данных вся многораздельная система обозначается одним *nname* рабочей станции, однако у каждого узла есть свое производное уникальное *nname* NetBIOS.

nname рабочей станции, обозначающее всю многораздельную систему, хранится в файле конфигурации менеджера баз данных на сервере разделения базы данных, которому принадлежит текущий экземпляр.

Уникальное *nname* узла представляет собой сочетание *nname* рабочей станции и номера этого узла.

Если узел не является владельцем экземпляра, его *nname* NetBIOS образуется так:

1. Первый символ *nname* рабочей станции компьютера-владельца экземпляра используется как первый символ *nname* NetBIOS.
2. Следующие 1 - 3 символа - это номер узла. Диапазон номеров - от 1 до 999.
3. Оставшиеся символы берутся из *nname* рабочей станции компьютера-владельца экземпляра. Число оставшихся символов зависит от длины *nname* рабочей станции компьютера-владельца экземпляра. Это может быть от 0 до 4 символов.

Например:

***nname* рабочей станции компьютера-владельца экземпляра**	**Номер узла**	**Производное *nname* узла NetBIOS**
GEORGE	3	G3ORGE
A	7	A7
B2	94	B942
N0076543	21	N216543
GEORGE5	1	G1RGE5

Если во время установки вы изменили *nname* рабочей станции по умолчанию, убедитесь, что последние четыре символа *nname* уникальны в сети NetBIOS, для того чтобы минимизировать риск возникновения конфликта имен *nname* NetBIOS.

Понятия, связанные с данным:

- “Правила именования” на стр. 321

Правила присвоения имен в среде NLS

Основной набор символов, который можно использовать в именах баз данных, состоит из однобайтных прописных и строчных латинских букв (A...Z, a...z), арабских цифр (0...9) и символа подчеркивания (_). К этому списку для обеспечения совместимости с программными продуктами баз данных хоста добавляется три специальных символа (#, @ и $) . Используйте специальные

символы #, @ и $ в среде NLS с осторожностью, поскольку они не включены в инвариантный набор символов хоста NLS (EBCDIC). В зависимости от применяемой кодовой страницы можно использовать и символы из расширенного набора символов. Если вы используете базу данных в среде с несколькими кодовыми страницами, необходимо убедиться, что все кодовые страницы поддерживают все элементы из того расширенного набора символов, который вы планируете использовать.

При именовании объектов баз данных (таких, как таблицы и производные таблицы), меток программ, переменных хоста и указателей могут быть также использованы элементы из расширенного набора символов (например, русские буквы). Какие конкретно символы доступны, зависит от используемой кодовой страницы.

Определение расширенного набора символов для идентификаторов DBCS:

В среде DBCS расширенный набор символов состоит из всех символов основного набора плюс:

- Все двухбайтные символы всех кодовых страниц DBCS, кроме двухбайтного пробела, являются допустимыми буквами.
- Двухбайтный пробел является специальным символом.
- Однобайтные символы, доступные на каждой из страниц со смешанной кодировкой, распадаются на несколько категорий:

Категория	Действительные значения кодов на каждой из страниц со смешанной кодировкой
Цифры	x30-39
Буквы	x23-24, x40-5A, x61-7A, xA6-DF (A6-DF только для кодовых страниц 932 и 942)
Специальные символы	Все другие действительные значения кодов однобайтных символов

Понятия, связанные с данным:

- “Правила именования” на стр. 321
- “Правила именования объектов DB2” на стр. 321
- “Правила именования рабочих станций” на стр. 326

Правила присвоения имен в среде Unicode

В базе данных UCS-2 все идентификаторы находятся в многобайтном UTF-8. Поэтому можно использовать любые символы UCS-2 в идентификаторах, в которых использование символа из расширенного набора символов (например, русской буквы или многобайтного символа) разрешено DB2® UDB.

Клиенты могут вводить любые символы, поддерживаемые их средой; все эти символы в идентификаторах будут преобразованы менеджером баз данных в UTF-8. При задании символов национальных языков в идентификаторах для базы данных UCS-2 следует учитывать два обстоятельства:

- Для каждого символа, кроме символов ASCII, требуется от двух до четырех байт. Поэтому *n*-байтный идентификатор может содержать только от *n*/4 до *n* символов в зависимости доли символов ASCII в нем. Если у вас только один-два символа не относятся к ASCII (например, в слове есть символ с диакритическим значком), значение ближе к *n* символам, а идентификатор, ни один символ которого не входит в ASCII (например, на японском языке), может содержать от *n*/4 до *n*/3 символов.
- Если идентификаторы будут вводиться из разных сред клиентов, они должны быть определены с использованием общего подмножества символов, доступных этим клиентам. Например, если обращение к базе данных UCS-2 производится из сред с латиницей-1, арабским и японским языками, все идентификаторы реально должны ограничиваться ASCII.

Понятия, связанные с данным:

- “Правила именования” на стр. 321
- “Правила именования объектов DB2” на стр. 321
- “Правила именования рабочих станций” на стр. 326

Приложение B. Службы каталога протокола LDAP (Lightweight Directory Access Protocol)

Простой протокол доступа к каталогам (LDAP) - Введение

Простой протокол доступа к каталогам (LDAP) - это стандартный протокол доступа к каталогам. Служба каталогов хранит информацию о ресурсах для множества систем и служб в распределенной среде и предоставляет клиенту и серверу доступ к этим ресурсам. Каждый экземпляр сервера баз данных публикует информацию о себе на сервере LDAP и сохраняет информацию о базе данных в каталоге LDAP при ее создании. Когда клиент соединяется с базой данных, он может получить информацию о нужном сервере из каталога LDAP. Следовательно, клиенту не нужно хранить каталог на локальном компьютере. Прикладные программы клиента используют каталог LDAP для поиска информации, необходимой для соединения с базой данных.

Есть механизм кэширования, позволяющий клиенту проводить поиск в каталоге LDAP только один раз, в локальных каталогах. Найденная информация сохраняется или кэшируется на локальном компьютере. Последующий доступ к ней зависит от значения параметра конфигурации менеджера баз данных *dir_cache* и значения реестра DB2LDAPCACHE.

- Если DB2LDAPCACHE=NO и *dir_cache*=NO, информация всегда ищется в LDAP.
- Если DB2LDAPCACHE=NO и *dir_cache*=YES, то информация, найденная в каталоге LDAP, помещается в кэш DB2®.
- Если DB2LDAPCACHE=YES или значение не задано, а локальный кэш не содержит нужной информации, она ищется в каталоге LDAP, после чего локальный кэш обновляется.

Примечание: Значение реестра DB2LDAPCACHE применимо только к базам данных и каталогам узлов.

Понятия, связанные с данным:

- “Служба каталога протокола LDAP (Lightweight Directory Access Protocol)” на стр. 79
- “Поддержка Active Directory операционной системы Windows” на стр. 333
- “Поддержка LDAP и DB2 Connect” на стр. 350
- “Особенности организации защиты в среде LDAP” на стр. 350
- “Особенности организации защиты в Active Directory Windows 2000” на стр. 351

- “Переменные реестра и среды DB2” в *Руководство администратора: Производительность*
- “Дополнение схемы каталога LDAP классами и атрибутами объектов DB2” на стр. 352

Задачи, связанные с данной темой:

- “Настройка DB2 для применения Active Directory” на стр. 334
- “Настройка DB2 в среде LDAP фирмы IBM” на стр. 335
- “Создание пользователя LDAP” на стр. 336
- “Настройка пользователя LDAP для применения программ DB2” на стр. 337
- “Регистрация серверов DB2 после установки” на стр. 338
- “Обновление протокола для сервера DB2” на стр. 340
- “Создание алиаса узла для команды ATTACH” на стр. 340
- “Отмена регистрации сервера DB2” на стр. 341
- “Регистрация баз данных в каталоге LDAP” на стр. 341
- “Подключение к удаленному серверу в среде LDAP” на стр. 342
- “Отмена регистрации базы данных в каталоге LDAP” на стр. 343
- “Обновление записей LDAP в каталоге локальной базы данных и узлов” на стр. 344
- “Поиск в разделах каталога LDAP или доменах” на стр. 345
- “Регистрация баз данных хоста в LDAP” на стр. 346
- “Настройка переменных реестра DB2 на уровне пользователя в среде LDAP” на стр. 347
- “Включение поддержки LDAP после завершения установки” на стр. 348
- “Выключение поддержки LDAP” на стр. 349
- “Дополнение схемы каталога в Active Directory Windows 2000” на стр. 353

Ссылки, связанные с данной темой:

- “Поддерживаемые конфигурации клиента и сервера LDAP” на стр. 332
- “Объекты DB2 в Active Directory Windows 2000” на стр. 354
- “Классы объектов и атрибуты LDAP, применяемые DB2” на стр. 355
- “Поддержка каталога LDAP Netscape и определения атрибутов” на стр. 367

Поддерживаемые конфигурации клиента и сервера LDAP

В следующей таблице перечислены поддерживаемые конфигурации клиента и сервера LDAP:

Таблица 20. Поддерживаемые конфигурации клиента и сервера LDAP

	IBM SecureWay Directory	**Microsoft Active Directory**	**Сервер LDAP Netscape**
Клиент LDAP IBM	Поддерживается	Не поддерживается	Поддерживается
Клиент LDAP/ADSI Microsoft	Поддерживается	Поддерживается	Поддерживается

IBM SecureWay Directory, Версия 3.1 - сервер LDAP версии 3 для систем Windows NT, AIX и Solaris. SecureWay Directory поставляется в составе базовой операционной системы в AIX и iSeries (AS/400), а также вместе с сервером защиты OS/390.

DB2 поддерживает клиент LDAP IBM в системах AIX, Solaris, Windows NT и Windows 98.

Microsoft Active Directory - сервер LDAP версии 3, входящий в операционную систему Windows 2000 Server.

Клиент LDAP Microsoft входит в состав операционной системы Windows.

При работе в операционной системе Windows DB2 позволяет обращаться к серверу каталогов IBM SecureWay как с помощью клиента LDAP IBM, так и с помощью клиента LDAP Microsoft. Чтобы явно указать выбор клиента LDAP IBM, с помощью команды **db2set** задайте для переменной реестра DB2LDAP_CLIENT_PROVIDER значение “IBM”.

Поддержка Active Directory операционной системы Windows

DB2® применяет Active Directory следующим образом:

1. Серверы баз данных DB2 заносятся в каталог Active Directory как объекты ibm_db2Node. Класс объектов ibm_db2Node является подклассом ServiceConnectionPoint (SCP). Каждый объект ibm_db2Node содержит информацию о конфигурации протокола, позволяющую клиентским программам установить соединение с соответствующим сервером баз данных DB2. Когда создается новая база данных, она заносится в каталог Active Directory как объект ibm_db2Database, подчиненный объекту ibm_db2Node.
2. При соединении с удаленной базой данных клиент DB2 через интерфейс LDAP ищет в каталоге Active Directory объект ibm_db2Database. Информация о протоколе, применяемом для подключения к серверу баз данных, (информация о связывании) считывается из объекта ibm_db2Node, на основе которого создан объект ibm_db2Database.

Свойства объектов ibm_db2Node и ibm_db2Database можно просмотреть и изменить с помощью консоли управления *Active Directory - Пользователи и компьютеры* (MMC), предусмотренной на контроллере домена. Для того чтобы задать страницу свойств, вызовите команду regsrv32 для регистрации страниц свойств объектов DB2, как показано ниже:

```
regsvr32 %DB2PATH%\bin\db2ads.dll
```

Для просмотра объектов откройте консоль управления *Active Directory - Пользователи и компьютеры* (MMC) на контроллере домена. Для этого выберите Пуск—> Программы—> Администрирование—> Active Directory - Пользователи и компьютеры.

Примечание: Для просмотра объектов DB2, расположенных в объектах компьютеров, выберите в меню View пункт *Users, Groups, and Computers as containers*.

Примечание: Если DB2 не установлен на контроллере домена, то для просмотра свойств объектов DB2 необходимо скопировать файл db2ads.dll из каталога %DB2PATH%\bin и DLL ресурсов db2adsr.dll из каталога %DB2PATH%\msg\имя-локали в локальный каталог контроллера домена. После этого нужно вызвать команду regsrv32 из локального каталога для регистрации DLL.

Понятия, связанные с данным:

- “Особенности организации защиты в Active Directory Windows 2000” на стр. 351

Задачи, связанные с данной темой:

- “Настройка DB2 для применения Active Directory” на стр. 334
- “Дополнение схемы каталога в Active Directory Windows 2000” на стр. 353

Ссылки, связанные с данной темой:

- “Объекты DB2 в Active Directory Windows 2000” на стр. 354

Настройка DB2 для применения Active Directory

Процедура:

Для доступа к Microsoft Active Directory должны выполняться следующие условия:

1. Компьютер, где работает DB2, должен принадлежать к домену Windows 2000.

2. Должен быть установлен клиент LDAP Microsoft. Клиент LDAP Microsoft входит в операционную систему Windows 2000. В системах Windows 98 и Windows NT надо убедиться, что в системном каталоге имеется `wldap32.dll`.
3. Должна быть включена поддержка LDAP. В Windows 2000 поддержка LDAP обеспечивается программой установки. В Windows 98/NT следует разрешить LDAP в явном виде, задав для переменной реестра DB2_ENABLE_LDAP значение “YES” при помощи команды **db2set**.
4. Для получения информации из каталога Active Directory следует войти в систему в качестве пользователя домена во время работы DB2.

Понятия, связанные с данным:

- “Поддержка Active Directory операционной системы Windows” на стр. 333
- “Переменные реестра и среды DB2” в *Руководство администратора: Производительность*

Задачи, связанные с данной темой:

- “Настройка пользователя LDAP для применения программ DB2” на стр. 337

Настройка DB2 в среде LDAP фирмы IBM

Процедура:

Чтобы использовать DB2 в среде LDAP IBM, необходимо на каждом компьютере сконфигурировать:

- Поддержку LDAP. В Windows 2000 поддержка LDAP обеспечивается программой установки. В Windows 98/NT следует разрешить LDAP в явном виде, задав для переменной реестра DB2_ENABLE_LDAP значение “YES” при помощи команды **db2set**.
- Имя хоста TCP/IP и номер порта сервера LDAP. Эти значения можно ввести при автоматической установке с помощью ключевого слова файла ответов DB2LDAPHOST или позже задать вручную командой DB2SET:

  ```
  db2set DB2LDAPHOST=<имя_хоста[:порт]>
  ```

  где `имя_хоста` - имя хоста TCP/IP сервера LDAP, а `[:порт]` - номер порта. Если номер порта не указан, DB2 использует порт LDAP по умолчанию (389).

  Объекты DB2 размещаются в базовом определенном имени LDAP (baseDN). При использовании сервера каталогов LDAP IBM SecureWay версии 3.1 не требуется конфигурировать базовое определенное имя, поскольку DB2 может динамически получать эту информацию с сервера. Однако при использовании сервера IBM eNetwork Directory Server версии 2.1 базовое определенное имя LDAP необходимо сконфигурировать на каждом компьютере с помощью команды DB2SET:

```
db2set DB2LDAP_BASEDN=<baseDN>
```

где baseDN - имя суффикса LDAP, определенного на сервере LDAP. Этот суффикс LDAP применяется для хранения объектов DB2.

- Имя (DN) и пароль пользователя LDAP. Они требуются только для хранения с помощью LDAP информации DB2, относящейся к отдельным пользователям.

Понятия, связанные с данным:

- "Переменные реестра и среды DB2" в *Руководство администратора: Производительность*

Задачи, связанные с данной темой:

- "Настройка DB2 для применения Active Directory" на стр. 334
- "Создание пользователя LDAP" на стр. 336

Ссылки, связанные с данной темой:

- "db2set - DB2 Profile Registry Command" в *Command Reference*

Создание пользователя LDAP

Процедура:

DB2 поддерживает задание переменных реестра DB2 и конфигурирование CLI на уровне пользователя. (На платформах AIX и Solaris это недоступно.) Уровень пользователя поддерживает настройки отдельных пользователей в многопользовательской среде. Примером может служить Windows NT Terminal Server, где каждый зарегистрированный пользователь может настраивать свою среду, не влияя на системную среду или среду других пользователей.

При использовании каталога LDAP IBM для хранения в LDAP информации уровня пользователя необходимо сначала создать пользователя LDAP. Пользователя LDAP можно создать одним из следующих способов:

- Создать файл LDIF, содержащий все атрибуты объекта пользователя, и запустить утилиту импорта LDIF, чтобы импортировать этот объект в каталог LDAP. Утилита LDIF для сервера LDAP IBM - "LDIF2DB".
- Создать объект пользователя с помощью программы DMT (Directory Management Tool, есть только в IBM SecureWay LDAP Directory Server Версии 3.1).

Файл LDIF, содержащий атрибуты объекта пользователя, имеет следующий вид:

```
File name: newuser.ldif

dn: cn=Mary Burnnet, ou=DB2 UDB Development, ou=Toronto, o=ibm, c=ca
```

```
objectclass: ePerson
cn: Mary Burnnet
sn: Burnnet
uid: mburnnet
userPassword: password
telephonenumber: 1-416-123-4567
facsimiletelephonenumber: 1-416-123-4568
title: Software Developer
```

Ниже приводится пример команды LDIF, импортирующей файл LDIF с помощью утилиты импорта LDIF IBM:

```
LDIF2DB -i newuser.ldif
```

Примечания:

1. Команду LDIF2DB следует выполнить на компьютере сервера LDAP.
2. Объекту пользователя LDAP следует дать необходимые права доступа (ACL), чтобы пользователь мог добавлять, удалять, просматривать и изменять свой объект. Права ACL даются объектам пользователей с помощью программы LDAP Directory Server Web Administration.

Задачи, связанные с данной темой:

- “Настройка DB2 в среде LDAP фирмы IBM” на стр. 335
- “Настройка пользователя LDAP для применения программ DB2” на стр. 337

Настройка пользователя LDAP для применения программ DB2

Процедура:

При работе с клиентом LDAP Microsoft пользователь LDAP совпадает с учетной записью пользователя в операционной системе. Однако если применяется клиент LDAP IBM, то перед обращением к DB2 следует настроить отличительное имя (DN) и пароль пользователя LDAP для текущего пользователя операционной системы. Это можно сделать с помощью утилиты db2ldcfg:

```
db2ldcfg -u <DN-пользователя> -w <пароль> -> задает DN и пароль пользователя
         -r  -> удаляет DN и пароль пользователя
```

Например:

```
db2ldcfg -u "cn=Mary Burnnet,ou=DB2 UDB Development,ou=Toronto,o=ibm,c=ca"
         -w пароль
```

Задачи, связанные с данной темой:

- “Создание пользователя LDAP” на стр. 336

Ссылки, связанные с данной темой:

- “db2ldcfg - Configure LDAP Environment Command” в *Command Reference*

Регистрация серверов DB2 после установки

Процедура:

Каждый экземпляр сервера DB2 надо зарегистрировать в LDAP, чтобы сделать доступной информацию о протоколе, который клиентские программы используют для связи с этим экземпляром сервера DB2. При регистрации экземпляра сервера баз данных необходимо задать *имя узла*. Имя узла используется клиентскими программами, когда они подключаются к серверу или устанавливают с ним соединение. Вы можете добавить в каталог другой алиас для узла LDAP с помощью команды **CATALOG LDAP NODE**.

Примечание: В домене Windows 2000 экземпляр сервера DB2 при установке автоматически регистрируется в Active Directory со следующей информацией:

```
nodename: имя хоста TCP/IP
protocol type: TCP/IP
```

Если имя хоста TCP/IP длиннее восьми символов, оно усекается до восьми символов.

Команда **REGISTER** задается в следующем формате:

```
db2 register db2 server in ldap
   as <имя_узла_ldap>
   protocol tcpip
```

Условие `protocol` указывает, какой протокол использовать для связи с этим сервером баз данных.

При создании экземпляра DB2 Universal Database Enterprise Server Edition, включающего несколько физических компьютеров, команду **REGISTER** нужно вызвать для каждого компьютера. Для того чтобы вызвать команду **REGISTER** сразу на всех компьютерах, воспользуйтесь командой **rah**.

Примечание: Нельзя использовать одно `имя_узла_ldap` на всех компьютерах, поскольку оно должно быть уникально в LDAP. `Имя_узла_ldap` рекомендуется заменить в команде **REGISTER** на имя хоста соответствующего компьютера. Например:

```
   rah ">DB2
REGISTER DB2 SERVER IN LDAP AS <> PROTOCOL TCPIP"
```

На всех компьютерах, где выполняется команда **rah**, символ “<>” заменяется на имя хоста. В том случае, если создано несколько экземпляров DB2 Universal Database Enterprise Server Edition, в качестве имени узла в команде **rah** рекомендуется задать сочетание имени экземпляра и индекса хоста.

Команду **REGISTER** можно вызвать для удаленного сервера DB2. Для этого следует указать имя удаленного компьютера, имя экземпляра и параметры конфигурации протокола при регистрации удаленного сервера. Команда может иметь следующий вид:

```
db2 register db2 server in ldap
   as <имя_узла_ldap>
   protocol tcpip
   hostname <имя_хоста>
   svcename <имя_службы_tcpip>
   remote <имя_удаленного_компьютера>
   instance <имя_экземпляра>
```

Для имен компьютеров используется следующее соглашение:

- Если задан протокол TCP/IP, имя компьютера должно совпадать с именем хоста TCP/IP.
- Если задан протокол APPN, в качестве имени компьютера следует использовать имя LU партнера.

При работе в среде высокой доступности или с обработкой отказов с использованием протокола TCP/IP следует использовать IP-адрес *кластера*.IP-адрес кластера позволяет клиенту подключаться к серверу на всех компьютерах, не регистрируя на каждом из них отдельный узел TCP/IP. IP-адрес кластера задается в условии `hostname` так:

```
db2 register db2 server in ldap
   as <имя_узла_ldap>
   protocol tcpip
   hostname n.nn.nn.nn
```

где `n.nn.nn.nn` - IP-адрес кластера.

Понятия, связанные с данным:

- “Обзор команд rah и db2_all” на стр. 372

Задачи, связанные с данной темой:

- “Обновление протокола для сервера DB2” на стр. 340
- “Создание алиаса узла для команды ATTACH” на стр. 340
- “Отмена регистрации сервера DB2” на стр. 341
- “Подключение к удаленному серверу в среде LDAP” на стр. 342

Ссылки, связанные с данной темой:

- “REGISTER Command” в *Command Reference*
- “CATALOG LDAP NODE Command” в *Command Reference*

Обновление протокола для сервера DB2

Процедура:

Информация о сервере DB2 в LDAP должна всегда соответствовать текущему моменту. Например, изменения в параметрах конфигурации протокола или в правах доступа к серверу по сети должны вноситься в LDAP.

Следующая команда позволяет изменить описание сервера DB2 в LDAP на локальном компьютере:

```
db2 update ldap ...
```

Примеры изменяемых параметров конфигурации протокола:

- Имя хоста TCP/IP и имя службы или номер порта.
- Информация протокола APPC, такая как имя TP, LU партнера или режим.
- Имя рабочей станции NetBIOS.

Чтобы изменить параметры конфигурации протокола удаленного сервера DB2, используйте команду UPDATE LDAP с условием node:

```
db2 update ldap
   node <имя_узла>
   hostname <имя_хоста>
   svcename <имя_службы_tcpip>
```

Задачи, связанные с данной темой:

- “Регистрация серверов DB2 после установки” на стр. 338
- “Создание алиаса узла для команды ATTACH” на стр. 340
- “Подключение к удаленному серверу в среде LDAP” на стр. 342

Ссылки, связанные с данной темой:

- “UPDATE LDAP NODE Command” в *Command Reference*

Создание алиаса узла для команды ATTACH

Процедура:

При регистрации сервера DB2 в LDAP необходимо задать для него имя узла. Программы используют его для соединения с этим сервером баз данных. Если требуется другой узел, например, имя узла жестко закодировано в программе, имя узла можно изменить с помощью команды CATALOG LDAP NODE. Команда имеет следующий вид:

```
db2 catalog ldap node <имя_узла_ldap>
   as <новое_имя_алиас>
```

Удалить узел LDAP из каталога можно с помощью команды UNCATALOG LDAP NODE COMMAND. Она имеет следующий вид:

```
db2 uncatalog ldap node <имя_узла_ldap>
```

Задачи, связанные с данной темой:

- “Регистрация серверов DB2 после установки” на стр. 338
- “Подключение к удаленному серверу в среде LDAP” на стр. 342

Ссылки, связанные с данной темой:

- “CATALOG LDAP NODE Command” в *Command Reference*
- “UNCATALOG LDAP NODE Command” в *Command Reference*

Отмена регистрации сервера DB2

Процедура:

Отмена регистрации экземпляра в каталоге LDAP удаляет также все объекты узлов или алиасов и объекты баз данных, относящиеся к этому экземпляру.

При отмене регистрации сервера DB2 на локальном или удаленном компьютере необходимо указать для него имя узла LDAP:

```
db2 deregister db2 server in ldap
   node <имя_узла>
```

После того, как регистрация сервера DB2 отменена, из каталога удаляются также все записи узлов и баз данных LDAP, относящихся к этому экземпляру сервера DB2.

Задачи, связанные с данной темой:

- “Регистрация серверов DB2 после установки” на стр. 338

Ссылки, связанные с данной темой:

- “DEREGISTER Command” в *Command Reference*

Регистрация баз данных в каталоге LDAP

Процедура:

При создании базы данных на экземпляре она автоматически регистрируется в LDAP. Регистрация позволяет удаленным клиентам подключаться к ней, не внося эту базу данных и узел в каталог на компьютере клиента. Когда клиент пытается установить соединение с базой данных, отсутствующей в каталоге баз данных на локальном компьютере, производится поиск в каталоге LDAP.

Если в каталоге LDAP уже есть это имя, база данных все равно создается на локальном компьютере, но выдается предупреждение о конфликте имен в каталоге LDAP. Поэтому базу данных можно внести в каталог LDAP вручную. Пользователи могут регистрировать базы данных на удаленных серверах в LDAP с помощью команды CATALOG LDAP DATABASE. При регистрации удаленной базы данных следует указать имя узла LDAP, представляющего удаленный сервер баз данных. Вы **должны** зарегистрировать удаленный сервер баз данных в LDAP с помощью команды REGISTER DB2 SERVER IN LDAP **до** регистрации базы данных.

Ручная регистрация базы данных в LDAP выполняется с помощью команды CATALOG LDAP DATABASE:

```
db2 catalog ldap database <имя_базы_данных>
   at node <имя_узла>
   with "Моя база данных LDAP"
```

Задачи, связанные с данной темой:

- “Регистрация серверов DB2 после установки” на стр. 338
- “Отмена регистрации базы данных в каталоге LDAP” на стр. 343

Ссылки, связанные с данной темой:

- “CATALOG LDAP DATABASE Command” в *Command Reference*

Подключение к удаленному серверу в среде LDAP

Процедура:

В среде LDAP можно подключиться к удаленному серверу баз данных при помощи команды ATTACH с именем узла LDAP:

```
db2 attach to <имя_узла_ldap>
```

Когда клиентская программа подключается к узлу или устанавливает соединение с базой данных в первый раз, узла нет в локальном каталоге узлов, поэтому DB2 ищет запись, соответствующую узлу назначения, в каталоге LDAP. Если в каталоге LDAP есть такая запись, будет получена информация о протоколе удаленного сервера. При подключении к базе данных запись в каталоге LDAP дает также информацию о базе данных. DB2 автоматически вносит эту информацию для базы данных и узла в каталог на локальном компьютере. При следующем обращении клиентской программы к тому же узлу или базе данных используется информация из локального каталога баз данных, а в каталоге LDAP поиск не ведется.

Подробнее: Есть механизм кэширования, благодаря которому клиент при просмотре локальных каталогов баз данных проводит поиск в каталоге LDAP

только один раз. Найденная информация сохраняется или кэшируется на локальном компьютере. Последующий доступ к ней зависит от значения параметра конфигурации менеджера баз данных *dir_cache* и значения реестра DB2LDAPCACHE.

- Если DB2LDAPCACHE=NO и *dir_cache*=NO, информация всегда ищется в LDAP.
- Если DB2LDAPCACHE=NO, а *dir_cache*=YES, информация, найденная в LDAP, помещается в кэш DB2.
- Если DB2LDAPCACHE=YES или значение не задано, а локальный кэш не содержит нужной информации, она ищется в каталоге LDAP, после чего локальный кэш обновляется.

Примечание: Кэширование информации LDAP неприменимо к CLI уровня пользователя и к переменным реестра профилей DB2. Кроме того, для каталогов баз данных, узлов и DCS существует кэш “в памяти”. Однако для каталога узла такого кэша нет.

Понятия, связанные с данным:

- “Переменные реестра и среды DB2” в *Руководство администратора: Производительность*

Задачи, связанные с данной темой:

- “Регистрация серверов DB2 после установки” на стр. 338
- “Обновление протокола для сервера DB2” на стр. 340
- “Создание алиаса узла для команды ATTACH” на стр. 340
- “Регистрация баз данных в каталоге LDAP” на стр. 341

Ссылки, связанные с данной темой:

- “ATTACH Command” в *Command Reference*

Отмена регистрации базы данных в каталоге LDAP

Процедура:

База данных автоматически удаляется из каталога LDAP, когда:

- Эта база данных отбрасывается.
- Из каталога LDAP удаляется экземпляр-владелец.

Базу данных можно удалить из каталога LDAP следующей командой:

```
db2 uncatalog ldap database <имя_базы_данных>
```

Задачи, связанные с данной темой:

- “Регистрация баз данных в каталоге LDAP” на стр. 341

Ссылки, связанные с данной темой:

- “UNCATALOG LDAP DATABASE Command” в *Command Reference*

Обновление записей LDAP в каталоге локальной базы данных и узлов

Процедура:

Информация LDAP часто меняется, что делает необходимым обновление записей LDAP в локальных каталогах баз данных и узлов. Эти каталоги используются LDAP для кэширования записей.

Подробнее: Есть механизм кэширования, благодаря которому клиент при просмотре локальных каталогов баз данных проводит поиск в каталоге LDAP только один раз. Найденная информация сохраняется или кэшируется на локальном компьютере. Последующий доступ к ней зависит от значения параметра конфигурации менеджера баз данных *dir_cache* и значения реестра DB2LDAPCACHE.

- Если DB2LDAPCACHE=NO и *dir_cache*=NO, информация всегда ищется в LDAP.
- Если DB2LDAPCACHE=NO, а *dir_cache*=YES, информация, найденная в LDAP, помещается в кэш DB2.
- Если DB2LDAPCACHE=YES или значение не задано, а локальный кэш не содержит нужной информации, она ищется в каталоге LDAP, после чего локальный кэш обновляется.

Примечание: Кэширование информации LDAP неприменимо к CLI уровня пользователя и к переменным реестра профилей DB2. Кроме того, для каталогов баз данных, узлов и DCS существует кэш “в памяти”. Однако для каталога узла такого кэша нет.

Следующая команда позволяет обновить записи LDAP, соответствующие базам данных:

```
db2 refresh ldap database directory
```

Обновить записи LDAP, соответствующие узлам, можно с помощью следующей команды:

```
db2 refresh ldap node directory
```

При обновлении записи LDAP, сохранявшиеся в локальных каталогах баз данных и узлов, удаляются. При следующем обращении к базе данных или узлу программа возьмет информацию непосредственно от LDAP и создаст новую запись в локальном каталоге узлов или баз данных.

Некоторые способы обеспечить своевременность обновления:

- Запланировать периодическое обновление.
- Выполнять команду REFRESH при загрузке системы.
- С помощью пакета управления выполнять команду REFRESH на всех компьютерах клиентов.
- Задать значение DB2LDAPCACHE="NO", чтобы запретить кэширование информации LDAP в каталогах баз данных, узлов и DCS.

Понятия, связанные с данным:

- "Переменные реестра и среды DB2" в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- "Параметр конфигурации Поддержка кэша каталогов - dir_cache" в *Руководство администратора: Производительность*
- "REFRESH LDAP Command" в *Command Reference*

Поиск в разделах каталога LDAP или доменах

Процедура:

DB2 производит поиск в текущем разделе каталога LDAP или, в среде Windows 2000, в текущем домене Active Directory. В среде с несколькими разделами каталога LDAP или доменами область поиска можно регулировать. Например, если информация не найдена в текущем разделе или домене, можно потребовать автоматического поиска во всех остальных разделах или доменах. Можно, напротив, ограничить поиск пределами локального компьютера.

Сфера поиска регулируется переменной реестра профиля DB2 DB2LDAP_SEARCH_SCOPE. Задать сферу поиска в LDAP на глобальном уровне можно с помощью опции "-gl" ("глобально в LDAP") команды *db2set*:

```
db2set -gl db2ldap_search_scope=<значение>
```

Возможные значения: "local", "domain" и "global". Значение по умолчанию - "domain", при котором поиск ограничивается текущим разделом каталога. В LDAP можно установить сферу поиска по умолчанию для всего предприятия. Например, можно установить глобальный поиск ("global") после создания новой базы данных. Тогда компьютеры клиентов смогут просмотреть все остальные разделы или домены и найти базу данных, определенную в одном из них. После первого соединения, когда на компьютере каждого клиента создана соответствующая запись, сферу поиска можно сузить до локального - "local". Значение "local" не дает клиентам просматривать разделы или домены.

Примечание: Переменная реестра профиля DB2 DB2LDAP_SEARCH_SCOPE - единственная переменная реестра, значение которой можно задавать на глобальном уровне в LDAP.

Понятия, связанные с данным:

- “Переменные реестра и среды DB2” в *Руководство администратора: Производительность*

Задачи, связанные с данной темой:

- “Объявление переменных реестра и среды” на стр. 33

Регистрация баз данных хоста в LDAP

Процедура:

При регистрации баз данных хоста в LDAP возможны две конфигурации:

- Прямое соединение с базами данных хоста; и
- Соединение с базой данных хоста через шлюз.

В первом случае пользователь регистрирует в LDAP сервер хоста, после чего вносит в каталог LDAP базу данных хоста, указывая имя узла сервера хоста. Во втором случае пользователь регистрирует в LDAP сервер шлюза, после чего вносит в каталог LDAP базу данных хоста, указывая имя узла сервера шлюза.

В следующем примере показаны оба случая: Предположим, у вас есть база данных хоста под названием NIAGARA_FALLS. Она принимает входящие соединения APPN и TCP/IP. Клиенты, которые из-за отсутствия DB2 Connect не могут соединяться с хостом напрямую, будут устанавливать соединение через шлюз под названием “goto@niagara”.

Необходимо выполнить следующие шаги:

1. Зарегистрировать сервер баз данных хоста в LDAP для соединений APPN. Условия REMOTE и INSTANCE не обязательны. Условие NODETYPE имеет значение “DCS” в знак того, что сервер является сервером баз данных хоста.

   ```
   db2 register ldap as nfappn appn network CAIBMOML partnerlu NFLU
      mode IBMRDB remote mvssys instance msvinst nodetype dcs
   ```

2. Зарегистрировать сервер баз данных хоста в LDAP для соединений TCP/IP. Имя хоста TCP/IP сервера - “myhost”, номер порта - “446”. Как и в шаге 1, условие NODETYPE имеет значение “DCS” в знак того, что сервер является сервером баз данных хоста.

   ```
   db2 register ldap as nftcpip tcpip hostname myhost svcename 446
      remote mvssys instance mvsinst nodetype dcs
   ```

3. Зарегистрировать сервер шлюза DB2 Connect в LDAP для соединений TCP/IP. Имя хоста TCP/IP сервера шлюза - “niagara”, номер порта - “50000”.

   ```
   db2 register ldap as whasf tcpip hostname niagara svcename 50000
      remote niagara instance goto nodetype server
   ```

4. Внести базу данных хоста в каталог LDAP с указанием TCP/IP. Имя базы данных хоста - “NIAGARA_FALLS”, алиас базы данных - “nftcpip”. Имя узла сервера шлюза DB2 Connect указывается в условии GWNODE.

   ```
   db2 catalog ldap database NIAGARA_FALLS as nftcpip at node nftcpip
      gwnode whasf authentication dcs
   ```

5. Внести базу данных хоста в каталог LDAP с указанием APPN.

   ```
   db2 catalog ldap database NIAGARA_FALLS as nfappn at node nfappn
      gwnode whasf authentication dcs
   ```

После вышеописанной регистрации соединения с хостом с использованием TCPIP устанавливаются через “nftcpip”. Соединения с использованием APPN устанавливаются через “nfappn”. Если на рабочей станции клиента нет DB2 Connect, соединение пройдет через шлюз с протоколом TCPIP и далее к хосту с протоколом TCP/IP или APPN, в зависимости от того, используется “nftcpip” или “nfappn”.

В результате можно вручную сконфигурировать информацию о базе данных в LDAP, чтобы каждому клиенту не надо было вносить базу данных и узел в каталог на каждом локальном компьютере. Процесс описан ниже:

1. Зарегистрируйте сервер баз данных хоста в LDAP. Необходимо указать имя удаленного компьютера, имя экземпляра и тип узла для сервера баз данных хоста в команде REGISTER в условиях REMOTE, INSTANCE и NODETYPE соответственно. Значением REMOTE может быть имя хоста или имя LU компьютера сервера хоста. Значением INSTANCE может быть любая строка не длиннее восьми символов. (Например, в качестве имени экземпляра можно использовать “DB2”). В качестве значения NODE TYPE надо задать “DCS” в знак того, что сервер является сервером баз данных хоста.
2. Зарегистрируйте базу данных хоста в LDAP с помощью команды CATALOG LDAP DATABASE. Все дополнительные параметры DRDA можно задать в условии PARMS. Тип аутентификации базы данных должен иметь значение “DCS”.

Ссылки, связанные с данной темой:

- “REGISTER Command” в *Command Reference*
- “CATALOG LDAP DATABASE Command” в *Command Reference*

Настройка переменных реестра DB2 на уровне пользователя в среде LDAP

Процедура:

В среде LDAP значения переменных реестра профиля DB2 можно задавать на уровне пользователя, что позволяет пользователям настраивать свою среду DB2. Задать значения реестра профиля DB2 на уровне пользователя позволяет опция `-ul`:

```
db2set -ul <переменная>=<значение>
```

Примечание: Это не поддерживается в системах AIX и Solaris.

В DB2 есть механизм кэширования. Переменные реестра профиля DB2 на пользовательском уровне кэшируются на локальном компьютере. Если задан параметр `-ul`, DB2 всегда считывает значения переменных реестра DB2 из кэша. Кэш обновляется, когда:

- Значение переменной реестра DB2 изменяют или сбрасывают на уровне пользователя.
- Команда обновления переменных профиля LDAP на уровне пользователя имеет вид:

  ```
  db2set -ur
  ```

Задачи, связанные с данной темой:

- “Объявление переменных реестра и среды” на стр. 33

Ссылки, связанные с данной темой:

- “db2set - DB2 Profile Registry Command” в *Command Reference*

Включение поддержки LDAP после завершения установки

Процедура:

Чтобы включить поддержку LDAP после завершения установки, выполните следующие действия на всех компьютерах:

- Установите двоичные файлы поддержки LDAP. Запустите программу установки, выберите `Custom` (пользовательская установка) -> LDAP Directory Exploitation (Поддержка использования каталогов LDAP). Программа установки установит двоичные файлы и задаст для переменной реестра профиля DB2 DB2_ENABLE_LDAP значение “YES”.

  Примечание: На платформах Windows 98/NT и UNIX следует разрешить LDAP в явном виде, задав для переменной реестра DB2_ENABLE_LDAP значение “YES” с помощью команды **db2set**.

- (Только на платформах UNIX) Объявите имя хоста TCP/IP сервера LDAP и (необязательно) номер порта при помощи следующей команды:

  ```
  db2set DB2LDAPHOST=<базовое_имя_домена>[:номер_порта]
  ```

где базовое_имя_домена - это имя хоста TCP/IP сервера LDAP, а [:порт] - номер порта. Если номер порта не указан, DB2 использует порт LDAP по умолчанию (389).

Объекты DB2 размещаются в базовом определенном имени LDAP (baseDN). При использовании сервера каталогов LDAP IBM SecureWay версии 3.1 не требуется конфигурировать базовое определенное имя, поскольку DB2 может динамически получать эту информацию с сервера. Однако при использовании сервера IBM eNetwork Directory Server версии 2.1 базовое определенное имя LDAP необходимо сконфигурировать на каждом компьютере с помощью команды DB2SET:

```
db2set DB2LDAP_BASEDN=<baseDN>
```

где baseDN - имя суффикса LDAP, определенного на сервере LDAP. Этот суффикс LDAP содержит объекты DB2.

- Зарегистрируйте текущий экземпляр сервера DB2 в LDAP с помощью команды REGISTER LDAP AS. Например:

```
db2 register ldap as <имя-узла> protocol tcpip
```

- Чтобы зарегистрировать в LDAP базы данных, выполните команду CATALOG LDAP DATABASE. Например:

```
db2 catalog ldap database <имя_базы_данных> as <имя_алиас>
```

- Введите имя (DN) и пароль пользователя LDAP. Они требуются только для хранения с помощью LDAP информации DB2, относящейся к отдельным пользователям.

Понятия, связанные с данным:

- “Переменные реестра и среды DB2” в *Руководство администратора: Производительность*

Задачи, связанные с данной темой:

- “Выключение поддержки LDAP” на стр. 349

Ссылки, связанные с данной темой:

- “REGISTER Command” в *Command Reference*
- “db2set - DB2 Profile Registry Command” в *Command Reference*
- “CATALOG LDAP DATABASE Command” в *Command Reference*

Выключение поддержки LDAP

Процедура:

Чтобы отключить поддержку LDAP, выполните следующие действия:

- Для каждого экземпляра сервера DB2 отмените регистрацию сервера DB2 в LDAP:

  ```
  db2 deregister db2 server in ldap node <имя_узла>
  ```

- Задайте для переменной реестра профиля DB2 DB2_ENABLE_LDAP значение “NO”.

Задачи, связанные с данной темой:

- “Объявление переменных реестра и среды” на стр. 33
- “Включение поддержки LDAP после завершения установки” на стр. 348

Ссылки, связанные с данной темой:

- “DEREGISTER Command” в *Command Reference*

Поддержка LDAP и DB2 Connect

Если на шлюзе DB2® Connect поддерживается LDAP, и база данных не была найдена в каталоге баз данных шлюза, то DB2 выполнит поиск в LDAP и сохранит найденную информацию.

Особенности организации защиты в среде LDAP

Прежде чем программа или пользователь получает доступ к каталогу LDAP, они проходят аутентификацию на сервере LDAP. Процесс аутентификации называется *связыванием* с сервером LDAP.

Очень важно регулировать доступ к информации в каталоге LDAP, чтобы предотвратить добавление, удаление или изменение этой информации неизвестными пользователями.

Права доступа наследуются по умолчанию и могут применяться на уровне контейнера. Новый объект наследует тот же атрибут защиты, что и его родительский объект. Определить управление доступом к контейнеру можно с помощью доступных для сервера LDAP средств управления.

По умолчанию права доступа определяются так:

- Для записей LDAP, соответствующих базам данных и узлам, право чтения имеет каждый пользователь. Право чтения/записи имеют только администратор каталогов и владелец или создатель соответствующего объекта.
- Владелец профиля и администратор каталогов имеют право чтения/записи профиля пользователя. Один пользователь не может получить доступ к профилю другого пользователя, если у него нет полномочий администратора каталогов.

Примечание: Проверка авторизации всегда осуществляется сервером LDAP, а не DB2. Процесс авторизации LDAP не связан с авторизацией DB2®. Учетная запись или ID пользователя с полномочиями SYSADM может не иметь доступа к каталогу LDAP.

При выполнении команд LDAP и API, если не указаны определенное имя связывания (bindDN) и пароль, DB2 связывается с сервером LDAP, используя личные данные по умолчанию; если их полномочий недостаточно для выполнения этих команд, будет возвращено сообщение об ошибке.

bindDN и пароль пользователя можно задать в явном виде с помощью условий USER и PASSWORD в командах DB2 и API.

Понятия, связанные с данным:

- “Особенности организации защиты в Active Directory Windows 2000” на стр. 351

Особенности организации защиты в Active Directory Windows 2000

В Active Directory объекты узлов и баз данных DB2® создаются в объекте того компьютера, на котором установлен сервер DB2. Чтобы зарегистрировать в Active Directory сервер баз данных или внести в каталог базу данных, необходимо иметь права доступа, достаточные для создания и/или удаления объектов, подчиненных объекту компьютера.

По умолчанию право чтения объектов, подчиненных объекту компьютера, есть у всех пользователей, прошедших аутентификацию, а право их изменения - у администраторов (пользователей, принадлежащих к группам администраторов, администраторов домена и администраторов предприятия). Дать право доступа определенному пользователю или группе можно с помощью консоли управления (MMC) *Active Directory - Пользователи и компьютеры* так:

1. Запустите средство управления *Active Directory - Пользователи и компьютеры* (Пуск —> Программы —> Администрирование —> Active Directory - Пользователи и компьютеры)
2. В меню *Вид* выберите *Дополнительные возможности*
3. Выберите контейнер *Компьютеры*
4. Щелкните правой кнопкой мыши по объекту, представляющему компьютер сервера, где установлена программа DB2, и выберите *Свойства*
5. Выберите закладку *Защита* и дайте необходимые права доступа указанному пользователю или группе

Переменные реестра DB2 и параметры CLI на уровне пользователя определяются объектом свойств DB2, подчиненным объекту пользователя.

Чтобы задать значения переменных реестра DB2 или параметров CLI на уровне пользователя, пользователь должен обладать достаточными правами для создания объектов, подчиненных объекту пользователя.

По умолчанию право создавать объекты, подчиненные объекту пользователя, есть только у администраторов. Дать пользователю право задания значений переменных реестра DB2 или параметров CLI на уровне пользователя можно с помощью Консоли управления (MMC) *Active Directory - Пользователи и компьютеры* так:

1. Запустите средство управления *Active Directory - Пользователи и компьютеры* (Пуск —> Программы —> Администрирование —> Active Directory - Пользователи и компьютеры)
2. Выберите объект пользователя в контейнере Пользователи
3. Щелкните по объекту пользователя правой кнопкой мыши и выберите *Свойства*
4. Выберите закладку *Защита*
5. Добавьте в список имя пользователя при помощи кнопки Добавить
6. Предоставьте права доступа “Запись” и “Создание всех дочерних объектов”
7. При помощи дополнительных параметров задайте применение разрешений к “Этому объекту и всем дочерним объектам”
8. Включите переключатель “Распространить на объект наследуемые разрешения родительских объектов”

Понятия, связанные с данным:

- “Особенности организации защиты в среде LDAP” на стр. 350

Дополнение схемы каталога LDAP классами и атрибутами объектов DB2

Схема каталога LDAP определяет классы объектов и атрибуты информации, хранящейся в каталоге LDAP. Класс объектов состоит из набора обязательных и необязательных атрибутов. С каждой записью в каталоге LDAP связан класс объектов.

Чтобы DB2® могла записывать информацию в LDAP, схема каталога сервера LDAP должна включать используемые DB2 классы объектов и атрибуты. Процесс добавления новых классов объектов и атрибутов в базовую схему называется дополнением схемы каталога.

Примечание: При использовании IBM® SecureWay® LDAP Directory версии 3.1 все классы объектов и атрибуты, необходимые DB2, включены в базовую схему. Дополнять базовую схему не требуется.

Задачи, связанные с данной темой:

• “Дополнение схемы каталога в Active Directory Windows 2000” на стр. 353

Дополнение схемы каталога в Active Directory Windows 2000

Процедура:

Чтобы DB2 могла записывать информацию в Active Directory Windows 2000, необходимо дополнить схему каталога классами объектов и атрибутами DB2. Процесс добавления новых классов объектов и атрибутов в схему каталога называется *дополнением схемы*.

Схему Active Directory необходимо дополнить с помощью программы установки схемы DB2 **db2schex** до первой установки DB2 на любом компьютере, входящем в домен Windows 2000.

Программа **db2schex** есть на компакт-диске продукта. Она находится в каталоге `db2` в подкаталоге `common`. Например:

```
x:\db2\common
```

где `x:` - буква дисковода компакт-дисков.

Ниже приведен формат команды:

```
db2schex
```

С этой командой используются также необязательные условия:

- –b DN-пользователя

  Задает имя пользователя.
- –w Пароль

  Задает пароль связывания.
- –u

  Деинсталлирует схему.
- –k

  Принудительно продолжает деинсталляцию, несмотря на ошибки.

Примечания:

1. Если DN пользователя и пароль не указаны, **db2schex** связывается как текущий пользователь.
2. В качестве DN пользователя можно указать имя пользователя Windows NT.
3. Чтобы изменить схему, необходимо входить в группу администраторов схем или иметь права на изменение этой схемы.

Примеры:

- Для установки схемы DB2:

  ```
  db2schex
  ```

- Для установки схемы DB2 и задания DN пользователя и пароля:

  ```
  db2schex -b "cn=A Name,dc=toronto1,dc=ibm,dc=com"
           -w пароль
  ```

  Либо

  ```
  db2schex -b Administrator -w пароль
  ```

- Для деинсталляции схемы DB2:

  ```
  db2schex -u
  ```

- Для деинсталляции схемы DB2, несмотря на ошибки:

  ```
  db2schex -u -k
  ```

Программа установки схемы DB2 для Active Directory выполняет следующие задачи:

Примечания:

1. Определяет, какой сервер служит мастером схемы
2. Связывается с контроллером домена, являющимся мастером схемы
3. Проверяет, достаточно ли прав пользователя для добавления классов и атрибутов в схему
4. Проверяет, что в мастер схемы можно осуществлять запись (то есть в реестре удалена защитная блокировка)
5. Создает все новые атрибуты
6. Создает все новые классы объектов
7. Проверяет наличие ошибок и, если они обнаружены, отменяет все изменения в схеме.

Понятия, связанные с данным:

- “Дополнение схемы каталога LDAP классами и атрибутами объектов DB2” на стр. 352

Объекты DB2 в Active Directory Windows 2000

DB2 создает объекты в Active Directory в двух местах:

1. Объекты баз данных и узлов DB2 создаются как подчиненные объекту компьютера, где установлен сервер DB2. На компьютерах серверов DB2, не принадлежащих к домену Windows NT, объекты баз данных и узлов DB2 создаются в контейнере “System”.
2. Переменные реестра DB2 и параметры CLI на уровне пользователя хранятся в объектах свойств DB2, подчиненных объекту пользователя. Эти объекты содержат информацию, относящуюся к данному пользователю.

Ссылки, связанные с данной темой:

- “Классы объектов и атрибуты LDAP, применяемые DB2” на стр. 355

Классы объектов и атрибуты LDAP, применяемые DB2

Ниже в таблицах описаны классы объектов, используемые DB2:

Таблица 21. cimManagedElement

Класс	**cimManagedElement**
Имя дисплея LDAP Active Directory	Не применимо
Общее имя (cn) Active Directory	Не применимо
Описание	Служит базовым классом для многих классов объектов управления системой в схеме IBM
Надкласс	top
Обязательные атрибуты	
Необязательные атрибуты	description
Введите	abstract
OID (идентификатор объекта)	1.3.18.0.2.6.132
GUID (глобальный уникальный идентификатор)	b3afd63f-5c5b-11d3-b818-002035559151

Таблица 22. cimSetting

Класс	**cimSetting**
Имя дисплея LDAP Active Directory	Не применимо
Общее имя (cn) Active Directory	Не применимо
Описание	Служит базовым классом для конфигурации и параметров в схеме IBM
Надкласс	cimManagedElement
Обязательные атрибуты	
Необязательные атрибуты	settingID
Введите	abstract
OID (идентификатор объекта)	1.3.18.0.2.6.131
GUID (глобальный уникальный идентификатор)	b3afd64d-5c5b-11d3-b818-002035559151

Таблица 23. eProperty

Класс	**eProperty**
Имя дисплея LDAP Active Directory	ibm-eProperty
Общее имя (cn) Active Directory	ibm-eProperty

Таблица 23. eProperty (продолжение)

Класс	**eProperty**
Описание	Используется для задания значений пользовательских свойств для конкретной программы
Надкласс	cimSetting
Обязательные атрибуты	
Необязательные атрибуты	propertyType cisPropertyType cisProperty cesPropertyType cesProperty binPropertyType binProperty
Введите	structural
OID (идентификатор объекта)	1.3.18.0.2.6.90
GUID (глобальный уникальный идентификатор)	b3afd69c-5c5b-11d3-b818-002035559151

Таблица 24. DB2Node

Класс	**DB2Node**
Имя дисплея LDAP Active Directory	ibm-db2Node
Общее имя (cn) Active Directory	ibm-db2Node
Описание	Представляет сервер DB2
Надкласс	eSap / ServiceConnectionPoint
Обязательные атрибуты	db2nodeName
Необязательные атрибуты	db2nodeAlias db2instanceName db2Type host / dNSHostName (см. Примечание 2) protocolInformation/ServiceBindingInformation
Введите	structural
OID (идентификатор объекта)	1.3.18.0.2.6.116
GUID (глобальный уникальный идентификатор)	b3afd65a-5c5b-11d3-b818-002035559151

Таблица 24. DB2Node (продолжение)

Класс	**DB2Node**
Особые замечания	1. Класс *DB2Node* является производным от класса объектов *eSap* в IBM SecureWay Directory и от класса объектов *ServiceConnectionPoint* в Microsoft Active Directory. 2. Атрибут *host* используется в среде IBM SecureWay. Атрибут *dNSHostName* используется в Microsoft Active Directory. 3. Атрибут *protocolInformation* используется только в среде IBM SecureWay. В Microsoft Active Directory протокол задается атрибутом *ServiceBindingInformation*, наследуемым от класса ServiceConnectionPoint.

Атрибут *protocolInformation* (в IBM SecureWay Directory) или *ServiceBindingInformation* (в Microsoft Active Directory) объекта *DB2Node* обозначает протокол связи для связывания сервера баз данных DB2. Он состоит из элементов, описывающих поддерживаемый сетевой протокол. Элементы разделяются точками с запятыми. Между ними не ставятся пробелы. Необязательный параметр можно указать звездочкой (*).

Элементы для TCP/IP:

- “TCPIP”
- Имя хоста или IP-адрес сервера
- Имя службы (svcename) или номер порта (например, 50000)
- (Необязательный) защита (“NONE” или “SOCKS”)

Элементы для APPN:

- “APPN”
- ID сети
- LU партнера
- Имя программы транзакций (TP) (поддерживаются только прикладные TP, но не служебные TP – TP в формате HEX)
- Режим
- Защита (“NONE”, “SAME” или “PROGRAM”)
- (Необязательный) адрес сетевого адаптера
- (Необязательный) LU изменения пароля

Примечание: В клиенте DB2 for Windows NT (а также Windows 98), если в локальном стеке SNA не сконфигурирована информация APPN, но

адрес сетевого адаптера и LU изменения пароля указаны в LDAP, клиент DB2 пытается по возможности сконфигурировать стек SNA, используя эту информацию. Данная опция не поддерживается в клиентах DB2 for AIX и DB2 for Solaris.

Элементы для NetBIOS:

- “NETBIOS”
- Имя рабочей станции сервера NetBIOS

Элементы для именованных конвейеров:

- “NPIPE”
- Имя компьютера сервера
- Имя экземпляра сервера

Таблица 25. DB2Database

Класс	**DB2Database**
Имя дисплея LDAP Active Directory	ibm-db2Database
Общее имя (cn) Active Directory	ibm-db2Database
Описание	Представляет базу данных DB2
Надкласс	top
Обязательные атрибуты	db2databaseName db2nodePtr
Необязательные атрибуты	db2databaseAlias db2additionalParameter db2ARLibrary db2authenticationLocation db2gwPtr db2databaseRelease DCEPrincipalName
Введите	structural
OID (идентификатор объекта)	1.3.18.0.2.6.117
GUID (глобальный уникальный идентификатор)	b3afd659-5c5b-11d3-b818-002035559151

Таблица 26. db2additionalParameters

Атрибут	**db2additionalParameters**
Имя дисплея LDAP Active Directory	ibm-db2AdditionalParameters

Таблица 26. db2additionalParameters (продолжение)

Атрибут	**db2additionalParameters**
Общее имя (cn) Active Directory	ibm-db2AdditionalParameters
Описание	Содержит дополнительные параметры, используемые при соединении с сервером базы данных хоста
Синтаксис	Строка без учета регистра
Максимальная длина	1024
Количество значений	Одно значение
OID (идентификатор объекта)	1.3.18.0.2.4.426
GUID (глобальный уникальный идентификатор)	b3afd315-5c5b-11d3-b818-002035559151

Таблица 27. db2authenticationLocation

Атрибут	**db2authenticationLocation**
Имя дисплея LDAP Active Directory	ibm-db2AuthenticationLocation
Общее имя (cn) Active Directory	ibm-db2AuthenticationLocation
Описание	Указывает, где проводится аутентификация
Синтаксис	Строка без учета регистра
Максимальная длина	64
Количество значений	Одно значение
OID (идентификатор объекта)	1.3.18.0.2.4.425
GUID (глобальный уникальный идентификатор)	b3afd317-5c5b-11d3-b818-002035559151
Примечания	Разрешенные значения: CLIENT, SERVER, DCS, DCE, KERBEROS, SVRENCRYPT и DCSENCRYPT

Таблица 28. db2ARLibrary

Атрибут	**db2ARLibrary**
Имя дисплея LDAP Active Directory	ibm-db2ARLibrary
Общее имя (cn) Active Directory	ibm-db2ARLibrary
Описание	Имя библиотеки реквестера прикладных программ
Синтаксис	Строка без учета регистра
Максимальная длина	256
Количество значений	Одно значение
OID (идентификатор объекта)	1.3.18.0.2.4.427
GUID (глобальный уникальный идентификатор)	b3afd316-5c5b-11d3-b818-002035559151

Таблица 29. db2databaseAlias

Атрибут	**db2databaseAlias**
Имя дисплея LDAP Active Directory	ibm-db2DatabaseAlias
Общее имя (cn) Active Directory	ibm-db2DatabaseAlias
Описание	Алиас (алиасы) базы данных
Синтаксис	Строка без учета регистра
Максимальная длина	1024
Количество значений	Несколько значений
OID (идентификатор объекта)	1.3.18.0.2.4.422
GUID (глобальный уникальный идентификатор)	b3afd318-5c5b-11d3-b818-002035559151

Таблица 30. db2databaseName

Атрибут	**db2databaseName**
Имя дисплея LDAP Active Directory	ibm-db2DatabaseName
Общее имя (cn) Active Directory	ibm-db2DatabaseName
Описание	Имя базы данных
Синтаксис	Строка без учета регистра
Максимальная длина	1024
Количество значений	Одно значение
OID (идентификатор объекта)	1.3.18.0.2.4.421
GUID (глобальный уникальный идентификатор)	b3afd319-5c5b-11d3-b818-002035559151

Таблица 31. db2databaseRelease

Атрибут	**db2databaseRelease**
Имя дисплея LDAP Active Directory	ibm-db2DatabaseRelease
Общее имя (cn) Active Directory	ibm-db2DatabaseRelease
Описание	Выпуск базы данных
Синтаксис	Строка без учета регистра
Максимальная длина	64
Количество значений	Одно значение
OID (идентификатор объекта)	1.3.18.0.2.4.429
GUID (глобальный уникальный идентификатор)	b3afd31a-5c5b-11d3-b818-002035559151

Таблица 32. db2nodeAlias

Атрибут	**db2nodeAlias**
Имя дисплея LDAP Active Directory	ibm-db2NodeAlias
Общее имя (cn) Active Directory	ibm-db2NodeAlias
Описание	Алиас (алиасы) узла
Синтаксис	Строка без учета регистра
Максимальная длина	1024
Количество значений	Несколько значений
OID (идентификатор объекта)	1.3.18.0.2.4.420
GUID (глобальный уникальный идентификатор)	b3afd31d-5c5b-11d3-b818-002035559151

Таблица 33. db2nodeName

Атрибут	**db2nodeName**
Имя дисплея LDAP Active Directory	ibm-db2NodeName
Общее имя (cn) Active Directory	ibm-db2NodeName
Описание	Имя узла
Синтаксис	Строка без учета регистра
Максимальная длина	64
Количество значений	Одно значение
OID (идентификатор объекта)	1.3.18.0.2.4.419
GUID (глобальный уникальный идентификатор)	b3afd31e-5c5b-11d3-b818-002035559151

Таблица 34. db2nodePtr

Атрибут	**db2nodePtr**
Имя дисплея LDAP Active Directory	ibm-db2NodePtr
Общее имя (cn) Active Directory	ibm-db2NodePtr
Описание	Указатель на объект узла (DB2Node), представляющий сервер баз данных, который является владельцем базы данных.
Синтаксис	Определенное имя
Максимальная длина	1000
Количество значений	Одно значение
OID (идентификатор объекта)	1.3.18.0.2.4.423
GUID (глобальный уникальный идентификатор)	b3afd31f-5c5b-11d3-b818-002035559151

Таблица 34. db2nodePtr (продолжение)

Атрибут	**db2nodePtr**
Особые замечания	Это отношение позволяет клиенту находить информацию о протоколе связи для соединения с базой данных.

Таблица 35. db2gwPtr

Атрибут	**db2gwPtr**
Имя дисплея LDAP Active Directory	ibm-db2GwPtr
Общее имя (cn) Active Directory	ibm-db2GwPtr
Описание	Указатель на объект узла, который представляет сервер шлюза и обеспечивает доступ к базе данных.
Синтаксис	Определенное имя
Максимальная длина	1000
Количество значений	Одно значение
OID (идентификатор объекта)	1.3.18.0.2.4.424
GUID (глобальный уникальный идентификатор)	b3afd31b-5c5b-11d3-b818-002035559151

Таблица 36. db2instanceName

Атрибут	**db2instanceName**
Имя дисплея LDAP Active Directory	ibm-db2InstanceName
Общее имя (cn) Active Directory	ibm-db2InstanceName
Описание	Имя экземпляра сервера баз данных
Синтаксис	Строка без учета регистра
Максимальная длина	256
Количество значений	Одно значение
OID (идентификатор объекта)	1.3.18.0.2.4.428
GUID (глобальный уникальный идентификатор)	b3afd31c-5c5b-11d3-b818-002035559151

Таблица 37. db2Type

Атрибут	**db2Type**
Имя дисплея LDAP Active Directory	ibm-db2Type
Общее имя (cn) Active Directory	ibm-db2Type
Описание	Тип сервера баз данных
Синтаксис	Строка без учета регистра
Максимальная длина	64

Таблица 37. db2Type (продолжение)

Атрибут	**db2Type**
Количество значений	Одно значение
OID (идентификатор объекта)	1.3.18.0.2.4.418
GUID (глобальный уникальный идентификатор)	b3afd320-5c5b-11d3-b818-002035559151
Примечания	Разрешенные типы серверов баз данных: SERVER, MPP и DCS

Таблица 38. DCEPrincipalName

Атрибут	**DCEPrincipalName**
Имя дисплея LDAP Active Directory	ibm-DCEPrincipalName
Общее имя (cn) Active Directory	ibm-DCEPrincipalName
Описание	Имя принципала DCE
Синтаксис	Строка без учета регистра
Максимальная длина	2048
Количество значений	Одно значение
OID (идентификатор объекта)	1.3.18.0.2.4.443
GUID (глобальный уникальный идентификатор)	b3afd32d-5c5b-11d3-b818-002035559151

Таблица 39. cesProperty

Атрибут	**cesProperty**
Имя дисплея LDAP Active Directory	ibm-cesProperty
Общее имя (cn) Active Directory	ibm-cesProperty
Описание	С помощью значений этого атрибута могут задаваться параметры конфигурации, настроенные для конкретных прикладных программ. Например, значение может содержать данные в формате XML. Все значения этого атрибута должны иметь одинаковое значение атрибута cesPropertyType.
Синтаксис	Строка с учетом регистра
Максимальная длина	32700
Количество значений	Несколько значений
OID (идентификатор объекта)	1.3.18.0.2.4.307
GUID (глобальный уникальный идентификатор)	b3afd2d5-5c5b-11d3-b818-002035559151

Таблица 40. cesPropertyType

Атрибут	**cesPropertyType**
Имя дисплея LDAP Active Directory	ibm-cesPropertyType
Общее имя (cn) Active Directory	ibm-cesPropertyType
Описание	Значения этого атрибута могут описывать синтаксические, семантические и иные свойства всех значений атрибута cesProperty. Например, можно указать с помощью значения “XML”, что все значения атрибута cesProperty записаны в формате XML.
Синтаксис	Строка без учета регистра
Максимальная длина	128
Количество значений	Несколько значений
OID (идентификатор объекта)	1.3.18.0.2.4.308
GUID (глобальный уникальный идентификатор)	b3afd2d6-5c5b-11d3-b818-002035559151

Таблица 41. cisProperty

Атрибут	**cisProperty**
Имя дисплея LDAP Active Directory	ibm-cisProperty
Общее имя (cn) Active Directory	ibm-cisProperty
Описание	С помощью значений этого атрибута могут задаваться параметры конфигурации, настроенные для конкретных прикладных программ. Например, значение может содержать файл INI. Все значения этого атрибута должны иметь одинаковое значение атрибута cisPropertyType.
Синтаксис	Строка без учета регистра
Максимальная длина	32700
Количество значений	Несколько значений
OID (идентификатор объекта)	1.3.18.0.2.4.309
GUID (глобальный уникальный идентификатор)	b3afd2e0-5c5b-11d3-b818-002035559151

Таблица 42. cisPropertyType

Атрибут	**cisPropertyType**
Имя дисплея LDAP Active Directory	ibm-cisPropertyType
Общее имя (cn) Active Directory	ibm-cisPropertyType

Таблица 42. cisPropertyType (продолжение)

Атрибут	**cisPropertyType**
Описание	Значения этого атрибута могут описывать синтаксические, семантические и иные свойства всех значений атрибута cisProperty. Например, значение “INI File”, задает,что все значения атрибута cisProperty являются файлами INI.
Синтаксис	Строка без учета регистра
Максимальная длина	128
Количество значений	Несколько значений
OID (идентификатор объекта)	1.3.18.0.2.4.310
GUID (глобальный уникальный идентификатор)	b3afd2e1-5c5b-11d3-b818-002035559151

Таблица 43. binProperty

Атрибут	**binProperty**
Имя дисплея LDAP Active Directory	ibm-binProperty
Общее имя (cn) Active Directory	ibm-binProperty
Описание	С помощью значений этого атрибута могут задаваться параметры конфигурации, настроенные для конкретных прикладных программ. Например, значение может содержать набор свойств Lotus 123 в двоичной кодировке. Все значения этого атрибута должны иметь одинаковое значение атрибута binPropertyType.
Синтаксис	двоичный
Максимальная длина	250000
Количество значений	Несколько значений
OID (идентификатор объекта)	1.3.18.0.2.4.305
GUID (глобальный уникальный идентификатор)	b3afd2ba-5c5b-11d3-b818-002035559151

Таблица 44. binPropertyType

Атрибут	**binPropertyType**
Имя дисплея LDAP Active Directory	ibm-binPropertyType
Общее имя (cn) Active Directory	ibm-binPropertyType

Таблица 44. binPropertyType (продолжение)

Атрибут	**binPropertyType**
Описание	Значения этого атрибута могут описывать синтаксические, семантические и иные свойства всех значений атрибута binProperty. Например, значение “Lotus 123” задает, что все значения атрибута binProperty являются свойствами Lotus 123 в двоичной кодировке.
Синтаксис	Строка без учета регистра
Максимальная длина	128
Количество значений	Несколько значений
OID (идентификатор объекта)	1.3.18.0.2.4.306
GUID (глобальный уникальный идентификатор)	b3afd2bb-5c5b-11d3-b818-002035559151

Таблица 45. PropertyType

Атрибут	**PropertyType**
Имя дисплея LDAP Active Directory	ibm-propertyType
Общее имя (cn) Active Directory	ibm-propertyType
Описание	Значения этого атрибута описывают семантические свойства объекта eProperty
Синтаксис	Строка без учета регистра
Максимальная длина	128
Количество значений	Несколько значений
OID (идентификатор объекта)	1.3.18.0.2.4.320
GUID (глобальный уникальный идентификатор)	b3afd4ed-5c5b-11d3-b818-002035559151

Таблица 46. settingID

Атрибут	**settingID**
Имя дисплея LDAP Active Directory	Не применимо
Общее имя (cn) Active Directory	Не применимо
Описание	Атрибут, значения которого служат именами объектов, образованных от cimSetting, таких как eProperty
Синтаксис	Строка без учета регистра
Максимальная длина	256
Количество значений	Одно значение
OID (идентификатор объекта)	1.3.18.0.2.4.325

Таблица 46. settingID (продолжение)

Атрибут	settingID
GUID (глобальный уникальный идентификатор)	b3afd596-5c5b-11d3-b818-002035559151

Поддержка каталога LDAP Netscape и определения атрибутов

Для сервера LDAP Netscape поддерживаются уровни v4.12 или новее.

Сервер LDAP Netscape версии 4.12 и выше позволяет прикладным программам расширять схему путем добавления определений атрибутов и классов объектов в файлы slapd.user_oc.conf и slapd.user_at.conf. Эти два файла расположены в каталоге

```
<установочный-каталог-Netscape>\slapd-<имя-компьютера>\config
```

:NONE.

Примечание: Если применяется продукт iPlan Directory Server 5.0, то ознакомьтесь с подробными инструкциями по расширению схемы, приведенными в документации по этому продукту.

Атрибуты DB2 можно добавить в файл slapd.user_at.conf следующим образом:

```
#########################################################################
#
# IBM DB2 Universal Database
# Определения атрибутов
#
# bin -> двоичный
# ces -> строка с учетом регистра
# cis -> строка без учета регистра
# dn -> отличительное имя
#
#########################################################################

attribute binProperty                   1.3.18.0.2.4.305     bin
attribute binPropertyType               1.3.18.0.2.4.306     cis
attribute cesProperty                   1.3.18.0.2.4.307     ces
attribute cesPropertyType               1.3.18.0.2.4.308     cis
attribute cisProperty                   1.3.18.0.2.4.309     cis
attribute cisPropertyType               1.3.18.0.2.4.310     cis
attribute propertyType                  1.3.18.0.2.4.320     cis
attribute systemName                    1.3.18.0.2.4.329     cis
attribute db2nodeName                   1.3.18.0.2.4.419     cis
attribute db2nodeAlias                  1.3.18.0.2.4.420     cis
attribute db2instanceName               1.3.18.0.2.4.428     cis
attribute db2Type                       1.3.18.0.2.4.418     cis
attribute db2databaseName               1.3.18.0.2.4.421     cis
attribute db2databaseAlias              1.3.18.0.2.4.422     cis
attribute db2nodePtr                    1.3.18.0.2.4.423     dn
```

```
 attribute db2gwPtr                        1.3.18.0.2.4.424     dn
 attribute db2additionalParameters         1.3.18.0.2.4.426     cis
 attribute db2ARLibrary                    1.3.18.0.2.4.427     cis
 attribute db2authenticationLocation       1.3.18.0.2.4.425     cis
 attribute db2databaseRelease              1.3.18.0.2.4.429     cis
 attribute DCEPrincipalName                1.3.18.0.2.4.443     cis
```

Классы объектов DB2 можно добавить в файл slapd.user_oc.conf следующим образом:

```
#########################################################################
#
# IBM DB2 Universal Database
# Определения классов объектов
#
#########################################################################

objectclass eProperty
        oid 1.3.18.0.2.6.90
        requires
                objectClass
        allows
                cn,
                propertyType,
                binProperty,
                binPropertyType,
                cesProperty,
                cesPropertyType,
                cisProperty,
                cisPropertyType

objectclass eApplicationSystem
        oid 1.3.18.0.2.6.8
        requires
                objectClass,
                systemName


objectclass DB2Node
        oid 1.3.18.0.2.6.116
        requires
                objectClass,
                db2nodeName
        allows
                db2nodeAlias,
                host,
                db2instanceName,
                db2Type,
                description,
                protocolInformation

objectclass DB2Database
        oid 1.3.18.0.2.6.117
        requires
                objectClass,
```

```
        db2databaseName,
        db2nodePtr
allows
        db2databaseAlias,
        description,
        db2gwPtr,
        db2additionalParameters,
        db2authenticationLocation,
        DCEPrincipalName,
        db2databaseRelease,
        db2ARLibrary
```

После добавления определения схемы DB2 надо перезапустить сервер каталогов, чтобы внесенные изменения вступили в силу.

Ссылки, связанные с данной темой:

- “Классы объектов и атрибуты LDAP, применяемые DB2” на стр. 355

Приложение C. Выполнение команд на нескольких разделах базы данных

Вызов команд в среде многораздельной базы данных

В системе многораздельной базы данных можно выполнять команды на компьютерах экземпляра или на серверах разделов базы данных (узлах). Для этого служат команды **rah** и **db2_all**. Команда **rah** позволяет посылать команды компьютерам экземпляра. Чтобы выполнить команды на серверах разделов базы данных, пользуйтесь командой **db2_all**. Ниже приведен обзор этих команд. Все нижеизложенное относится только к системам многораздельных баз данных.

Примечания:

1. На платформах UNIX при входе в систему можно использовать как оболочку Korn, так и любую другую оболочку. Однако следует обратить внимание на то, что различные оболочки по-разному обрабатывают команды, содержащие специальные символы.
2. В Windows NT для запуска команды **rah** или **db2_all** необходимо быть зарегистрированным с учетной записью пользователя, входящего в группу администраторов.

Как определить область действия команды, описано в книге *Command Reference*. В этой книге указывается, выполняется ли команда на одном сервере раздела базы данных или на всех таких серверах. Если команда запускается на одном сервере раздела базы данных, и нужно, чтобы она выполнялась на всех серверах, воспользуйтесь командой **db2_all**. Исключение - команда **db2trc**, которая выполняется на всех логических узлах (серверах разделов базы данных) на компьютере. Чтобы выполнить **db2trc** на всех логических узлах всех компьютеров, используйте команду **rah**.

Понятия, связанные с данным:

- “Обзор команд rah и db2_all” на стр. 372
- “Вызов команд rah и db2_all” на стр. 373

Ссылки, связанные с данной темой:

- “Описание команд rah и db2_all” на стр. 372

Обзор команд rah и db2_all

Команды можно запускать последовательно на одном сервере раздела базы данных после другого или же параллельно. При параллельном выполнении команд на платформах на основе UNIX можно также по своему выбору направлять выходные данные в буфер и накапливать их там для вывода на экран (поведение по умолчанию) или выводить их на экран на компьютере, где введена команда. В Windows NT при параллельном выполнении команд выходные данные выводятся на экран на том компьютере, где введена команда.

Чтобы запустить команду **rah**, введите:

```
rah команда
```

Чтобы запустить команду **db2_all**, введите:

```
db2_all команда
```

Чтобы получить справку по синтаксису **rah**, введите:

```
rah "?"
```

В качестве команды можно ввести практически все, что допускается в приглашении командной строки, включая, например, несколько последовательно выполняемых команд. На платформах на основе UNIX последовательные команды разделяются точкой с запятой (;). В Windows NT команды разделяются знаком амперсанд (&). За последней командой символ разделения не ставится.

В следующем примере показано, как с помощью команды **db2_all** изменить конфигурацию базы данных на всех разделах базы данных, заданных в файле конфигурации узлов. Поскольку символ ; помещен в двойные кавычки, требование будет выполняться параллельно:

```
db2_all ";UPDATE DB CFG FOR sample USING LOGFILSIZ=100"
```

Понятия, связанные с данным:

- “Вызов команд в среде многораздельной базы данных” на стр. 371
- “Вызов команд rah и db2_all” на стр. 373

Ссылки, связанные с данной темой:

- “Описание команд rah и db2_all” на стр. 372

Описание команд rah и db2_all

Можно использовать следующие команды:

Команда **Описание**

rah Запускает команду на всех компьютерах.

db2_all Запускает команду на всех указанных вами серверах разделов баз данных.

db2_kill Принудительно останавливает все процессы, запущенные на нескольких серверах разделов баз данных и освобождает все ресурсы на всех серверах разделов баз данных. Эта команда оставляет базы данных в несогласованном состоянии. Ее *не следует* использовать, кроме тех случаев, когда этого требует центр обслуживания IBM.

db2_call_stack На платформах на основе UNIX заставляет все процессы, запущенные на всех серверах разделов баз данных, записывать трассировку вызовов в системный журнал.

В Windows NT заставляет все процессы, запущенные на всех серверах разделов баз данных, записывать трассировку вызовов в файл P*xxxx*.*nnn* в каталоге экземпляра, где P*xxxx* - ID процесса, a and *nnn* - номер узла.

На платформах на основе UNIX эти команды выполняют **rah** со следующими неявно заданными параметрами:

- Выполнять параллельно на всех компьютерах
- Буферизовать вывод команд, соответственно, в `/tmp/$USER/db2_kill` и `/tmp/$USER/db2_call_stack`.

В Windows NT эти команды отправляются на выполнение **rah** параллельно на всех компьютерах.

Понятия, связанные с данным:

- “Обзор команд rah и db2_all” на стр. 372
- “Вызов команд rah и db2_all” на стр. 373
- “Параллельное выполнение команд на платформах на базе UNIX” на стр. 375

Вызов команд rah и db2_all

Команду можно задать:

- Из командной строки в качестве параметра
- В ответ на приглашение, если никаких параметров не было задано.

Метод приглашения используется, если команда содержит следующие специальные символы:

```
| & ; < > ( )  { } [ ] и сам символ $ (не в качестве символа подстановки)
```

Если команда задается как параметр в командной строке, при наличии в ней любого из перечисленных здесь специальных символов ее следует заключить в двойные кавычки.

Примечание: На платформах на основе UNIX эта команда будет добавлена в ваш хронологический список команд сразу после ее ввода в ответ на приглашение.

Все специальные символы, кроме \, можно вводить в команде обычным способом (то есть не заключая в кавычки). Если в команду требуется включить \, надо ввести его дважды (\\).

Примечание: На платформах на основе UNIX, если не используется оболочка Korn, все специальные символы, кроме ", \, символа $, не используемого как символ подстановки, и одиночных кавычек ('), можно вводить в команде обычным способом (не заключая в кавычки). Если в команду требуется включить один из перечисленных символов, надо ввести перед ним три обратных косых черты (\\\). Например, если в команду нужно включить символ \, нужно ввести его четыре раза (\\\\).

Если в команду нужно включить двойные кавычки ("), перед ними надо поставить три обратных косых черты \\\".

Примечания:

1. На платформах на основе UNIX в команду нельзя включить одиночные кавычки ('), если ваша командная оболочка не допускает какой-либо способ введения этого символа в строку, взятую в одиночные кавычки.
2. В Windows NT в команду нельзя включить одиночные кавычки ('), если ваше командное окно не допускает какой-либо способ введения этого символа в строку, взятую в одиночные кавычки.

При выполнении любого сценария оболочки korn, содержащего операции чтения из stdin в фоновом режиме, надо явно перенаправить stdin в источник, откуда процесс может читать без остановки с терминала (сообщение SIGTTIN). Для перенаправления stdin надо запустить сценарий в следующем виде:

```
shell_script </dev/null &
```

если не задается ввода.

Подобным образом при запуске db2_all в фоновом режиме всегда надо задавать `</dev/null`. Например:

```
db2_all ";выполнить_эту_команду" </dev/null &
```

Таким образом вы перенаправляете stdin и избегаете остановки с терминала.

Другой вариант, если не предполагается вывода от удаленной команды - использовать опцию “daemonize” в префиксе db2_all:

```
db2_all ";daemonize_эта_команда" &
```

Понятия, связанные с данным:

- “Параллельное выполнение команд на платформах на базе UNIX” на стр. 375
- “Дополнительная информация о команде rah (для Solaris и AIX)” на стр. 377

Задачи, связанные с данной темой:

- “Настройка профиля среды по умолчанию для команды rah в Windows NT” на стр. 385

Ссылки, связанные с данной темой:

- “Описание команд rah и db2_all” на стр. 372
- “Префиксные последовательности команды rah” на стр. 378
- “Управление командой rah” на стр. 383

Параллельное выполнение команд на платформах на базе UNIX

Примечание: Информация в этом разделе относится только к платформам на основе UNIX.

По умолчанию команда запускается последовательно на каждом компьютере, но можно задать параллельное выполнение команд с помощью фоновых r-оболочек, введя перед командой определенные префиксные последовательности. Если r-оболочка запущена в фоновом режиме, каждая команда помещает вывод в файл буфера на своем удаленном компьютере. Этот процесс возвращает вывод в виде двух частей:

1. После завершения выполнения удаленной команды.
2. После завершения работы r-оболочки, которое может произойти позднее, если некоторые процессы еще выполняются.

Имя файла буфера по умолчанию - `/tmp/$USER/rahout`, но его можно изменить с помощью переменных среды $RAHBUFDIR/$RAHBUFNAME.

Если задано, что команды должны по умолчанию выполняться параллельно, в этом сценарии к команде, направляемой всем хостам, присоединяется спереди дополнительная команда, проверяющая, можно ли использовать переменные среды $RAHBUFDIR и $RAHBUFNAME для файла буфера. При этом создается переменная $RAHBUFDIR. Чтобы подавить этот процесс, экспортируйте переменную среды `RAHCHECKBUF=no`. Это позволит сэкономить время, если известно, что каталог существует и пригоден для использования.

Прежде чем использовать **rah** для параллельного выполнения какой-либо команды на нескольких компьютерах:

- Убедитесь, что каталог `/tmp/$USER` для вашего ID пользователя существует на каждом компьютере. Если такого каталога еще нет, создайте его при помощи команды:

  ```
  rah ")mkdir /tmp/$USER"
  ```

- Добавьте следующую строку к вашему файлу `.kshrc` (для синтаксиса оболочки Korn) или `.profile`, введя ее также в свой текущий сеанс:

  ```
  export RAHCHECKBUF=no
  ```

- На каждом компьютере, где выполняется удаленная команда, в файле `.rhosts` должна быть запись с ID компьютера, где запущена команда **rah**, а на компьютере, где запущена **rah**, в файле `.rhosts` должны быть записи с ID каждого компьютера, где выполняется удаленная команда.

Понятия, связанные с данным:

- “Дополнительная информация о команде rah (для Solaris и AIX)” на стр. 377

Задачи, связанные с данной темой:

- “Отслеживание процессов rah на платформах на базе UNIX” на стр. 376

Ссылки, связанные с данной темой:

- “Префиксные последовательности команды rah” на стр. 378
- “Диагностика ошибок при работе с командой rah на платформах на основе UNIX” на стр. 386

Отслеживание процессов rah на платформах на базе UNIX

Процедура:

Примечание: Информация в этом разделе относится только к платформам на основе UNIX.

Когда выполняются какие-либо удаленные команды или накапливается буферизованный вывод, процессы, запущенные `rah`, отслеживают действия системы:

- Записывают сообщения на терминал, указывая, какие команды не были запущены
- Получают буферизованный вывод.

Информационные сообщения записываются через интервал времени, который задает переменная среды RAHWAITTIME. Подробности задания этого интервала смотрите в справке. Можно полностью подавить вывод информационных сообщений, экспортировав `RAHWAITTIME=0`.

Первичный процесс мониторинга - это команда с именем (как показывает команда ps) **rahwaitfor**. Первое информационное сообщение содержит pid (идентификатор процесса) для этого процесса. Все остальные процессы мониторинга будут видны как команды **ksh**, запускающие сценарий **rah** (или имя символической связи). Если нужно, все процессы мониторинга можно остановить с помощью команды:

```
kill <pid>,
```

где <pid> - ID процесса для первичного процесса мониторинга. Не указывайте номер сигнала. Оставьте значение по умолчанию 15. Это никак не повлияет на удаленные команды, но предотвратит автоматический вывод на экран буферизованных выходных данных. Обратите внимание на то, что во время выполнения **rah** может в различные моменты выполняться два или более различных наборов процессов мониторинга. Однако если остановить в любой момент выполнение текущего набора, другие наборы не будут запущены.

Если вы не используете Korn в качестве обычной оболочки регистрации (например, `/bin/ksh`), использовать **rah** можно, но в этом случае будут несколько другими правила ввода команд, содержащих следующие специальные символы:

```
" $ (без подстановки) '
```

Чтобы получить дополнительную информацию, введите `rah "?"`. Кроме того, если в среде на основе UNIX на компьютере, выполняющем удаленные команды, в качестве оболочки регистрации не используется Korn, на компьютере, выполняющем **rah**, также не должна использоваться оболочка Korn. (**rah** определяет наличие оболочки Korn на удаленном компьютере на основе локального ID). Оболочка не должна выполнять никаких подстановок или специальной обработки строк в одинарных кавычках. Она должна оставить эту строку в точности как есть.

Понятия, связанные с данным:

- “Параллельное выполнение команд на платформах на базе UNIX” на стр. 375
- “Дополнительная информация о команде rah (для Solaris и AIX)” на стр. 377

Дополнительная информация о команде rah (для Solaris и AIX)

Для улучшения производительности в больших системах возможности rah позволяют использовать каскадный вызов. Это означает, что rah будет проверять, сколько узлов содержится в списке, и, если это число превышает заданное пороговое значение, построит подсписок узлов и пошлет этим узлам рекурсивный вызов себя самой. На этих узлах рекурсивно вызываемая rah ведет себя по тем же правилам, пока список не сократится настолько, чтобы

использовать стандартную логику отправки команд всем (конечным) узлам списка. Значение этого порога задается переменной среды RAHTREETHRESH; по умолчанию оно равно 15.

В системе с несколькими логическими узлами на одном физическом узле db2_all рекурсивно посылает вызов отдельным физическим узлам, которые затем пересылают его при помощи команды rsh другим логическим узлам на том же физическом узле; это сокращает трафик между физическими узлами. (Это относится только к команде db2_all, а не rah, поскольку rah всегда посылает вызов только различным физическим узлам.)

Понятия, связанные с данным:

- “Параллельное выполнение команд на платформах на базе UNIX” на стр. 375

Задачи, связанные с данной темой:

- “Отслеживание процессов rah на платформах на базе UNIX” на стр. 376

Префиксные последовательности команды rah

Префиксная последовательность состоит из одного или более специальных символов. Одна или несколько префиксных последовательностей вводятся непосредственно перед символами команды без всяких пробелов между ними. Если нужно задать несколько последовательностей, их можно вводить в любом порядке, но порядок символов в любой многосимвольной последовательности должен быть соблюден. При вводе любых префиксных последовательностей необходимо заключать всю команду вместе с ними в двойные кавычки, как показано в следующих примерах:

- На платформах на основе UNIX:

```
rah "};ps -F pid,ppid,etime,args -u $USER"
```

- В системе Windows NT:

```
rah "||db2 get db cfg for sample"
```

Используются следующие префиксные последовательности:

Последовательность	**Назначение**
|	Запускает команды последовательно в фоновом режиме.
|&	Запускает команды последовательно в фоновом режиме и завершает команду после того, как выполнены все удаленные команды, даже если какие-либо процессы еще выполняются. Это может произойти в том случае, если, например, продолжает выполняться дочерний процесс (на платформах на основе UNIX) или фоновый процесс (в Windows). В этом случае команда запускает отдельный фоновый процесс для получения всех

удаленных выходных данных, сгенерированных после завершения команды, и записывает их обратно на исходный компьютер.

Примечание: На платформах на основе UNIX при задании & производительность ухудшается, поскольку требуется больше команд **rsh**.

|| Запускает команды параллельно в фоновом режиме.

||& Запускает команды параллельно в фоновом режиме и завершает команду после того, как будут выполнены все удаленные команды, как описано выше для случая |&.

Примечание: На платформах на основе UNIX при задании & производительность ухудшается, поскольку требуется больше команд **rsh**.

; Эквивалентна ||&. Это альтернативная краткая форма.

Примечание: На платформах на основе UNIX при задании ; производительность по сравнению с || ухудшается, поскольку требуется больше команд **rsh**.

] Присоединяет спереди точечное выполнение профиля пользователя перед выполнением команды.

Примечание: Доступна только на платформах на основе UNIX.

} Присоединяет спереди точечное выполнение файла, указанного в $RAHENV (обычно `.kshrc`) перед выполнением команды.

Примечание: Доступна только на платформах на основе UNIX.

]} Присоединяет спереди точечное выполнение профиля пользователя и, вслед за этим, файла, указанного в $RAHENV (обычно `.kshrc`) перед выполнением команды.

Примечание: Доступна только на платформах на основе UNIX.

) Подавляет выполнение профиля пользователя и файла, указанного в $RAHENV.

Примечание: Доступна только на платформах на основе UNIX.

' Возвращает на компьютер вызов команды в виде эхо.

< Посылает вызов всем компьютерам, кроме данного.

<<-nnn< Посылает вызов всем серверам разделов баз данных, кроме *nnn* (все серверы разделов баз данных в файле `db2nodes.cfg` кроме

сервера с номером узла *nnn*, смотрите первый абзац за последней префиксной последовательностью в этой таблице).

<<+nnn< Посылает только серверу раздела базы данных *nnn* (сервер раздела базы данных в файле `db2nodes.cfg` с номером узла - *nnn*, смотрите первый абзац за последней префиксной последовательностью в этой таблице). примечание ниже).

(пробел) Запускает удаленную команду в фоновом режиме с закрытыми `stdin`, `stdout` и `stderr`. Эта опция действительна только при выполнении команды в фоновом режиме, то есть только в префиксной последовательности, включающей также \ или ;. Она позволяет завершить выполнение команды значительно быстрее (в момент инициации удаленной команды). Если задать эту префиксную последовательность в командной строке **rah**, следует также заключить команду в одинарные кавычки или же в двойные кавычки, поставив перед префиксным символом обратную косую черту. Например,

```
rah '; mydaemon'
```

или

```
rah ";\ mydaemon"
```

Команда **rah**, выполняемая как фоновый процесс, никогда не ждет возврата каких-либо выходных данных.

> Подставляет вместо <> имя компьютера.

" Подставляет вместо () индекс компьютера, а вместо ## - номер узла.

Примечания:

1. Индекс компьютера - это число, связанное с компьютером в системе баз данных. Если вы не запускаете несколько логических узлов, индекс компьютера соответствует номеру узла для этого компьютера в файле конфигурации узлов. Чтобы получить индекс компьютера в среде с несколькими логическими узлами, не следует повторно учитывать те из них, где запущено несколько логических узлов. Например, если на компьютере MACH1 работает два логических узла, и на компьютере MACH2 - два логических узла, номер узла для компьютера MACH3 в файле конфигурации узлов будет 5. Индекс компьютера для MACH3, однако, будет равен 3.

   В Windows NT файл конфигурации узлов редактировать нельзя. Чтобы получить индекс компьютера, используйте команду **db2nlist**.

2. Если задан символ ", повторы не удаляются из списка компьютеров.

При использовании префиксных последовательностей `<<-nnn<` и `<<+nnn<` *nnn* - любой номер раздела из 1, 2 или 3 цифр, который должен совпадать со значением *nodenum* в файле `db2nodes.cfg`.

Примечание: Префиксные последовательности рассматриваются как часть команды. Если префиксная последовательность задана как часть команды, всю команду вместе с префиксными последовательностями надо заключать в двойные кавычки.

Понятия, связанные с данным:

- “Вызов команд rah и db2_all” на стр. 373
- “Параллельное выполнение команд на платформах на базе UNIX” на стр. 375

Ссылки, связанные с данной темой:

- “Описание команд rah и db2_all” на стр. 372

Настройка списка компьютеров в среде многораздельной базы данных

Процедура:

По умолчанию список компьютеров берется из файла конфигурации узлов `db2nodes.cfg`. Это можно переопределить:

- Задав полное имя файла, содержащего список компьютеров, с помощью экспорта (на платформах на основе UNIX) или задания значения (в Windows NT) переменной среды RAHOSTFILE.
- Задав список явным образом как строку имен через пробелы с помощью экспорта (на платформах на основе UNIX) или задания значения (в Windows NT) переменной среды RAHOSTLIST.

  Примечание: Если заданы обе эти переменные среды, приоритет имеет RAHOSTLIST.

Примечание: В Windows NT во избежание внесения несогласованности в файл конфигурации узлов *не* редактируйте его вручную. Получить список компьютеров в экземпляре можно с помощью команды **db2nlist**.

Задачи, связанные с данной темой:

- “Исключение одинаковых записей из списка компьютеров в среде многораздельной базы данных” на стр. 382

Исключение одинаковых записей из списка компьютеров в среде многораздельной базы данных

Процедура:

Если DB2 Enterprise - Extended Edition запускается с несколькими логическими узлами (серверы разделов баз данных) на одном компьютере, в файле `db2nodes.cfg` для этого компьютера будет содержаться несколько записей. В этой ситуации команде **rah** требуется знать, должна ли ваша команда выполняться только один раз на каждом компьютере или один раз для каждого логического узла, заданного в файле `db2nodes.cfg`. Чтобы задать компьютеры, воспользуйтесь командой **rah**. Чтобы задать логические узлы, воспользуйтесь командой **db2_all**.

Примечание: На платформах на основе UNIX, если заданы компьютеры, **rah** будет, как правило, удалять повторы из списка компьютеров. Есть следующее исключение: если заданы логические узлы, **db2_all** добавит перед вашей командой следующее назначение:

```
export DB2NODE=nnn (в синтаксисе оболочки Korn)
```

где *nnn* - номер узла, взятый из соответствующей строки файл `db2nodes.cfg`, что позволяет направить команду на нужный сервер раздела базы данных.

При задании логических узлов можно ограничить список, включив в него все логические узлы, кроме одного, или же только один сервер раздела базы данных при помощи префиксных последовательностей `<<-nnn<` и `<<+nnn<`. Это может понадобиться, если сначала нужно запустить команду на узле каталога, а после этого - на всех остальных серверах разделов баз данных, возможно, в параллельном режиме. Обычно это требуется при запуске команды **db2 restart database**. Для этого вам надо знать номер узла каталога.

При выполнении **db2 restart database** с помощью команды **rah** повторные записи удаляются из списка компьютеров. Однако если задать префикс ”, повторы не будут удаляться, так как предполагается, что использование префикса ” означает отправку данных каждому серверу раздела базы данных, а не каждом компьютеру.

Задачи, связанные с данной темой:

- “Настройка списка компьютеров в среде многораздельной базы данных” на стр. 381

Ссылки, связанные с данной темой:

- “RESTART DATABASE Command” в *Command Reference*

- “Префиксные последовательности команды rah” на стр. 378

Управление командой rah

Для управления работой команды **rah** можно воспользоваться следующими переменными среды.

Таблица 47.

Имя	Смысл	По умолчанию
$RAHBUFDIR **Примечание:** Доступна только на платформах на основе UNIX.	Каталог для буфера	/tmp/$USER
$RAHBUFNAME **Примечание:** Доступна только на платформах на основе UNIX.	Имя файла для буфера	rahout
$RAHOSTFILE (на платформах на основе UNIX); RAHOSTFILE (в Windows NT)	Файл, содержащий список хостов	`db2nodes.cfg`
$RAHOSTLIST (на платформах на основе UNIX); RAHOSTLIST (в Windows NT)	Список хостов в виде строки	извлекается из $RAHOSTFILE
$RAHCHECKBUF **Примечание:** Доступна только на платформах на основе UNIX.	При значении ″no″ пропустить проверки	не задан
$RAHSLEEPTIME (на платформах на основе UNIX); RAHSLEEPTIME (в Windows NT)	Время в секундах, в течение которого данный сценарий будет дожидаться начального вывода от выполняемых параллельно команд	86400 секунд для **db2_kill**, 200 для всех остальных

Таблица 47. (продолжение)

Имя	Смысл	По умолчанию
$RAHWAITTIME (на платформах на основе UNIX); RAHWAITTIME (в Windows NT)	В Windows NT интервал в секундах между последовательными проверками, выполняются ли еще удаленные задания. На платформах на основе UNIX интервал в секундах между последовательными проверками, выполняются ли еще в данный момент удаленные задания, и сообщениями `rah: waiting for <pid> ...`. На любой платформе можно указать любое целое положительное число. Задание значения, начинающегося с нуля, например, export RAHWAITTIME=045, подавляет вывод сообщений. Задавать низкое значение нет необходимости, так как **rah** при обнаружении завершения заданий не опирается на результаты этих проверок.	45 секунд
$RAHENV **Примечание:** Доступна только на платформах на основе UNIX.	Задает имя выполняемого файла, если $RAHDOTFILES=E или K или PE или B	$ENV
$RAHUSER (на платформах на основе UNIX); RAHUSER (в Windows NT)	На платформах на основе UNIX - ID пользователя под которым должна быть выполнена удаленная команда. В Windows NT - учетная запись регистрации, связанная со службой DB2 Remote Command Service.	$USER

Примечание: На платформах на основе UNIX используется значение $RAHENV системы, где запущена **rah**, а не значение, заданное удаленной оболочкой (если оно есть).

Ссылки, связанные с данной темой:

- “Применение $RAHDOTFILES на платформах на основе UNIX” на стр. 385

Применение $RAHDOTFILES на платформах на основе UNIX

Примечание: Информация в этом разделе относится только к платформам на основе UNIX.

Ниже приводятся .-файлы, которые выполняются, если префиксная последовательность не задана:

P .profile

E Файл, указанный в $RAHENV (обычно `.kshrc`)

K Эквивалентно E

PE .profile, а затем файл, указанный в $RAHENV (обычно `.kshrc`)

B Эквивалентно PE

N Никакой

Примечание: Если вы не используете в качестве оболочки Korn, любые .-файлы, назначенные вами к выполнению, будут выполняться в процессе оболочки Korn и поэтому должны соответствовать ее синтаксису. Поэтому, если используется, например, оболочка регистрации C, чтобы настроить вашу среду `.cshrc` на команды, выполняемые **rah**, необходимо также создать `INSTHOME/.profile` оболочки Korn, эквивалентный вашему `.cshrc` и задать в вашем `INSTHOME/.cshrc`:

```
setenv RAHDOTFILES P
```

или же создать `INSTHOME/.kshrc` оболочки Korn, эквивалентный вашему `.cshrc` и задать в вашем `INSTHOME/.cshrc`:

```
setenv RAHDOTFILES E
setenv RAHENV INSTHOME/.kshrc
```

Существенно также, что ваш файл `.cshrc` не записывается в stdout, если нет tty (как при вызове командой **rsh**). Это можно обеспечить, вставив все строки, которые записывают в stdout, например, в следующие команды:

```
if { tty -s } then echo "executed .cshrc";
endif
```

Ссылки, связанные с данной темой:

• “Управление командой rah” на стр. 383

Настройка профиля среды по умолчанию для команды rah в Windows NT

Процедура:

Примечание: Информация в этом разделе относится только к Windows NT. Задать профиль среды по умолчанию для команды **rah** можно при помощи файла `db2rah.env`, который должен быть создан в каталоге экземпляра. Этот файл должен иметь следующий формат:

```
; Это строка комментария
DB2INSTANCE=имя_экземпляра
DB2DBDFT=имя_базы_данных
; Конец файла
```

Можно задать все переменные среды, которые требуются для инициализации среды для **rah**.

Понятия, связанные с данным:

- “Вызов команд rah и db2_all” на стр. 373

Диагностика ошибок при работе с командой rah на платформах на основе UNIX

Примечание: Информация в этом разделе относится только к платформам на основе UNIX.

Ниже приводятся советы относительно некоторых ошибок, которые могут иметь место при выполнении команды **rah**:

1. **rah** зависает (или выполняется слишком долго)

   Причина этой ошибки может заключаться в том, что:

   - **rah** определила, что ей требуется буферизовать вывод, а вы не экспортировали значение `RAHCHECKBUF=no`. Поэтому, прежде чем выполнять вашу команду, **rah** посылает команду всем компьютерам проверить существование каталога буфера и создать его, если такой каталог не существует.
   - Один или несколько компьютеров, которым была послана ваша команда, не отвечает. Срок ожидания для команды **rsh** в конце концов истечет, но он достаточно велик, обычно около 60 секунд.

2. Вы получили сообщения:

   - Неверное имя при регистрации
   - В разрешении отказано

   Либо ID компьютера, где запущена **rah**, не задан правильно в файле `.hosts` одного из компьютеров, либо на компьютере, где запущена **rah**, в файле `.rhosts` не задан правильно один из компьютеров.

3. При параллельном выполнении команд при помощи фоновых r-оболочек, хотя команды запускаются и выполняются на компьютерах в течение ожидаемого затраченного времени, **rah** требуется много времени, чтобы обнаружить это и выдать приглашение оболочки.

На компьютере, где запущена команда **rah**, в файле `.rhosts` не задан правильно один из компьютеров.

4. Хотя **rah** хорошо запускается из командной строки оболочки, если запустить **rah** удаленным образом с помощью, например, rsh,

   ```
   rsh somewher -l $USER db2_kill
   ```

   rah никогда не завершается.

   Это нормальная ситуация. **rah** запускает процессы фонового мониторинга, которые продолжаются и после ее завершения. Эти процессы обычно продолжаются до завершения всех процессов, связанных с выполняемой вами командой. Для команды **db2_kill** это означает завершение работы всех менеджеров баз данных. Процессы мониторинга можно завершить, отыскав процесс с командой **rahwaitfor** и введя команду kill <id_процесса>. Не указывайте номер сигнала. Вместо этого используйте значение по умолчанию (15).

5. Вывод **rah** показан неверно или при выдаче нескольких команд **rah** под одним и тем же $RAHUSER **rah** ошибочно сообщает, что переменная среды $RAHBUFNAME не существует.

   Это связано с тем, что при одновременном выполнении нескольких **rah** они пытаются использовать для буферизации вывода один и тот же файл буфера (например, $RAHBUFDIR/$RAHBUFNAME). Чтобы предотвратить эту ошибку, используйте разные $RAHBUFNAME для каждой из одновременно выполняемых команд **rah**, например, в следующей ksh:

   ```
   export RAHBUFNAME=rahout
   rah ";$command_1" &
   export RAHBUFNAME=rah2out
   rah ";$command_2" &
   ```

   или воспользуйтесь методом автоматического выбора уникального имени, например:

   ```
   RAHBUFNAME=rahout.$$ db2_all "....."
   ```

   Какой бы метод ни использовался, если пространство на диске ограничено, необходимо в какой-то момент обеспечить очистку файлов буфера. **rah** не стирает файл буфера в конце выполнения, хотя она стирает и затем использует повторно существующий файл в следующий раз, когда задается тот же файл буфера.

6. Вы ввели

   ```
   rah '"print from ()'
   ```

   и получили сообщение:

   ```
   ksh: syntax error at line 1 : (' unexpected
   ```

   Для подстановки () и ## необходимо:

- Использовать команду **db2_all**, а не **rah**.
- Задать для переменной среды RAHOSTFILE значение вашего файла `/sqllib/db2nodes.cfg` путем экспорта RAHOSTFILE или сохранения значения по умолчанию. Без этого **rah** оставит () и ## в неизменном виде. Вы получите сообщение об ошибке, так как команда **print from ()** недопустима.

Для повышения производительности при параллельном выполнении команд можно посоветовать использовать | вместо |& и || вместо ||& или ;, если только вам действительно не требуется функция, обеспечиваемая &. При задании & требуется больше команд **rsh**, что приводит к ухудшению производительности.

Ссылки, связанные с данной темой:

- “Управление командой rah” на стр. 383

Приложение D. Поддержка Windows Management Instrumentation (WMI)

Windows Management Instrumentation (WMI) - Введение

В настоящее время существует набор промышленных стандартов для инфраструктуры управления и определен способ объединения информации, полученной в различных аппаратных и программных системах управления. Этот набор стандартов называется Web-Based Enterprise Management (WBEM). WBEM основан на схеме Common Information Model (CIM), которая представляет собой промышленный стандарт, разработанный организацией Desktop Management Task Force (DMTF).

Microsoft® Windows® Management Instrumentation (WMI) - это реализация WBEM для поддерживаемых платформ Windows. WMI может применяться в корпоративной сети Windows для сокращения стоимости обслуживания ее компонентов. WMI предоставляет:

- Согласованную модель, описывающую схему работы, настройки и состояние Windows.
- API COM для получения доступа к информации управления.
- Возможность для работы с другими службами управления Windows.
- Гибкую и расширяемую архитектуру, позволяющую вендорам создавать собственные провайдеры WMI для поддержки новых устройств, приложений и других изменений.
- Язык запросов WMI (WQL) для запроса детальной информации.
- API для разработчиков управляющих программ, создающих сценарии Visual Basic и Windows Scripting Host (WSH).

Архитектура WMI состоит из двух компонентов:

1. Инфраструктуры управления, содержащей администратор объектов CIM (CIMOM) и центральную область хранения данных управления, называемую хранилищем объектов CIMOM. CIMOM предоставляет программам универсальный способ доступа к данным управления.
2. Провайдеров WMI. Провайдеры WMI играют роль посредников между CIMOM и управляемыми объектами. С помощью API WMI провайдеры WMI предоставляют данные из управляемых объектов администратору объектов CIM, обрабатывают запросы от имени управляющих программ и отправляют уведомления о событиях.

Провайдеры Windows Management Instrumentation (WMI) представляют собой стандартные серверы COM или DCOM, играющие роль посредников между управляемыми объектами и администратором объектов CIM (CIMOM). Когда управляющая программа запрашивает у CIMOM данные, которых нет в хранилище объектов CIMOM, либо информацию о событиях, CIMOM перенаправляет запрос провайдерам WMI. Провайдеры WMI предоставляют данные и уведомления о событиях, связанные с управляемыми объектами из их домена.

Понятия, связанные с данным:

- “Поддержка Windows Management Instrumentation в DB2 Universal Database” на стр. 390

Поддержка Windows Management Instrumentation в DB2 Universal Database

Windows® Management Instrumentation (WMI) может применять мониторы снимков путем просмотра счетчиков производительности DB2® и использования встроенного провайдера PerfMon.

Встроенный провайдер Registry позволяет WMI обращаться к переменным реестра профайла DB2.

Ряд встроенных провайдеров предусмотрен в продукте WMI Software Development Kit (WMI SDK):

- Провайдер PerfMon
- Провайдер Registry event
- Провайдер Registry
- Провайдер SNMP
- Провайдер Windows NT® event log
- Провайдер Win32
- Провайдер WDM

С помощью встроенного провайдера Windows NT Event Log WMI может получать информацию об ошибках DB2, хранящуюся в журналах событий.

В DB2 Universal Database™ (UDB) предусмотрен провайдер DB2 WMI Administration и примеры файлов сценариев WMI для доступа к следующим управляемым объектам:

1. Экземпляры сервера баз данных, включая экземпляры с несколькими разделами. Можно выполнять следующие операции:
   - Просматривать список экземпляров
   - Настраивать параметры менеджера баз данных

- Запускать, останавливать и запрашивать состояние службы сервера DB2
- Настраивать и устанавливать соединения

2. Базы данных. Можно выполнять следующие операции:
   - Просматривать список баз данных
   - Настраивать параметры базы данных
   - Создавать и отбрасывать базы данных
   - Создавать резервную копию, восстанавливать и повторять транзакции для баз данных

Перед применением прикладных программ WMI необходимо зарегистрировать провайдер DB2 WMI в системе. Для этого необходимо вызвать следующие команды:

- mofcomp %DB2PATH%\bin\db2wmi.mof

  Эта команда загружает определение схемы DB2 WMI в систему.
- regsvr %DB2PATH%\bin\db2wmi.dll

  Эта команда регистрирует DLL COM провайдера DB2 WMI в Windows.

В обеих командах вместо %DB2PATH% необходимо указать имя каталога, в котором установлен DB2. Файл db2wmi.mof - это файл .MOF, содержащий определение схемы DB2 WMI.

Интеграция инфраструктуры WMI дает следующие преимущества:

1. С помощью инструмента WMI можно создавать простые сценарии для управления серверами DB2 в среде Windows. Для выполнения простых задач, таких как получение списка экземпляров, создание и отбрасывание баз данных и обновление параметров конфигурации, предусмотрены примеры сценариев Visual Basic (VBS). Они поставляются вместе с продуктом DB2 Application Development for Windows.
2. Можно создать сложные управляющие программы, выполняющие многие задачи с помощью WMI. В их число могут входить следующие задачи:
   - Просмотр информации о системе
   - Отслеживание производительности DB2
   - Отслеживание потребления системных ресурсов DB2

   Создав управляющие программы, отслеживающие как системные события, так и события DB2, можно повысить эффективность управления базой данных.
3. Для создания программ достаточно иметь навыки работы с COM и Visual Basic. В результате программисты могут сэкономить время при создании прикладных программ для управления предприятием.

Понятия, связанные с данным:

- “Windows Management Instrumentation (WMI) - Введение” на стр. 389

Приложение E. Как DB2 for Windows NT работает с защитой Windows NT

DB2 for Windows NT и функции защиты Windows NT - Введение

Домен Windows® NT представляет собой группу компьютеров клиентов и серверов с уникальным именем. В домене применяется общая база данных учетных записей пользователей, которая называется Security Access Manager (SAM). Один из компьютеров домена играет роль контроллера домена. Он управляет всеми аспектами взаимодействия пользователей с доменом. Контроллер домена выполняет аутентификацию пользователей при их входе в домен с помощью базы данных учетных записей пользователей домена. В каждом домене один из контроллеров домена является главным контроллером домена (PDC). Кроме того, в домен могут входить резервные контроллеры домена (BDC), которые выполняют аутентификацию учетных записей пользователей, когда главный контроллер домена не выбран, либо недоступен. На резервных контроллерах домена хранится копия базы данных SAM, которая регулярно синхронизируются с основной копией, расположенной на PDC.

Для предоставления доступа к ресурсам домена учетные записи пользователей, ID пользователей и пароли достаточно определить только на основном контроллере домена.

Во время установки сервера Windows NT® можно указать, какую роль будет играть этот сервер:

- Роль главного контроллера домена в новом домене
- Роль резервного контроллера домена в существующем домене
- Роль автономного сервера в существующем домене.

Если вы решите создать “контроллер” в новом домене, то сервер будет назначен главным контроллером домена.

Пользователь может зарегистрироваться на локальном компьютере или, если компьютер установлен в домене Windows NT - в домене. Продукт DB2® for Windows NT поддерживает оба варианта. Для аутентификации пользователя DB2 вначале проверяет локальный компьютер, затем контроллер домена для текущего домена и, наконец, все доверенные домены, известные контроллеру домена.

Покажем, как это происходит, в предположении, что экземпляр DB2 требует аутентификации типа Server. Используется следующая конфигурация:

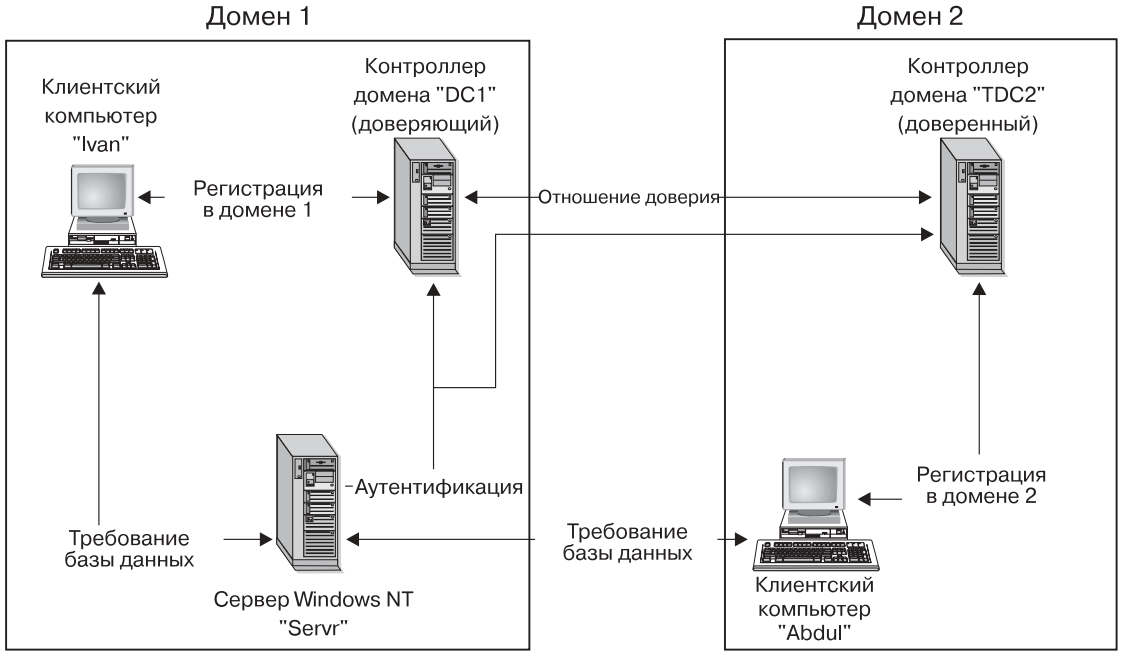


Рисунок 7. Аутентификация при помощи доменов Windows NT

У каждого компьютера, за исключением клиентских компьютеров Windows 9x, есть база данных защиты SAM (Security Access Management - управление доступом к защите). У компьютеров Windows 9x баз данных SAM нет. Пусть DC1 - контроллер домена, на котором записаны компьютер клиента Ivan и сервер DB2 for Windows NT Servr. TDC2 - доверенный домен для DC1, а компьютер клиента Abdul - член домена TDC2.

Понятия, связанные с данным:

- “Аутентификация пользователей и групп в Windows NT” на стр. 399

Задачи, связанные с данной темой:

- “Применение резервного контроллера домена в DB2” на стр. 397
- “Установка DB2 на резервном контроллере домена” на стр. 400
- “Аутентификация DB2 for Windows NT с применением функций защиты групп и доменов” на стр. 402

Сценарий аутентификации типа Server для DB2 for Windows NT

1. Abdul регистрируется в домене TDC2 (это значит, что он становится известен в базе данных SAM TDC2).
2. Затем Abdul соединяется с некоторой базой данных DB2, зарегистрированной в каталоге как находящаяся на SRV3:

   ```
   db2 connect to remotedb user Abdul using fredpw
   ```

3. SRV3 выясняет, где известен Abdul. Интерфейс API, используемый для поиска этой информации, вначале ведет поиск на локальном компьютере (SRV3), затем на контроллере домена (DC1), и только потом начинает перебирать все доверенные домены. Имя пользователя Abdul обнаруживается на TDC2. При таком порядке поиска требуется одно пространство имен для пользователей и групп.
4. Затем SRV3:
   a. Проверяет имя пользователя и пароль при помощи TDC2.
   b. Запрашивает TDC2, является ли пользователь Abdul администратором.
   c. Запрашивает у TDC2 список всех групп пользователя Abdul.

Понятия, связанные с данным:

• “DB2 for Windows NT и функции защиты Windows NT - Введение” на стр. 393

Сценарий аутентификации типа Client на клиентском компьютере Windows NT для DB2 for Windows NT

1. Администратор Dale регистрируется на SRV3 и изменяет тип аутентификации для экземпляра базы данных на Client:

   ```
   db2 update dbm cfg using authentication client
   db2stop
   db2start
   ```

2. Пользователь Ivan на компьютере клиента Windows регистрируется в домене DC1 (это значит, что он становится известен в базе данных SAM DC1).
3. Затем Ivan соединяется с некоторой базой данных DB2, зарегистрированной в каталоге как находящаяся на SRV3:

   ```
   DB2 CONNECT to remotedb user Ivan using johnpw
   ```

4. Компьютер пользователя Ivan проверяет имя пользователя и пароль. Интерфейс API, используемый для поиска этой информации, вначале ведет поиск на локальном компьютере (Ivan), затем на контроллере домена (DC1), и только потом начинает перебирать все доверенные домены. Имя пользователя Ivan обнаруживается в DC1.
5. Затем компьютер пользователя Ivan проверяет имя пользователя и пароль при помощи DC1.
6. Затем SRV3:
   a. Выясняет, известен ли Ivan.
   b. Запрашивает DC1, является ли пользователь Ivan администратором.

c. Запрашивает у DC1 список всех групп пользователя Ivan.

Примечание: Перед подключением к базе данных DB2 убедитесь, что запущена служба защиты DB2. Служба защиты устанавливается вместе с продуктом для операционной системы Windows. Во время установки DB2 “регистрируется” служба Windows NT, однако эта служба не запускается автоматически. Чтобы запустить службу защиты DB2, введите команду `NET START DB2NTSECSERVER`.

Понятия, связанные с данным:

- “DB2 for Windows NT и функции защиты Windows NT - Введение” на стр. 393

Сценарий аутентификации типа Client на клиентском компьютере Windows 9x для DB2 for Windows NT

1. Администратор Dale регистрируется на SRV3 и изменяет тип аутентификации для экземпляра базы данных на Client:

   ```
   db2 update dbm cfg using authentication client
   db2stop
   db2start
   ```

2. Пользователь Ivan на компьютере клиента Windows 9x входит в домен DC1 (это значит, что он становится известен в базе данных SAM DC1).
3. Затем Ivan соединяется с некоторой базой данных DB2, зарегистрированной в каталоге как находящаяся на SRV3:

   ```
   db2 connect to remotedb user Ivan using johnpw
   ```

4. Компьютер пользователя Ivan с Windows 9x не может проверить имя пользователя и пароль. Поэтому имя пользователя и пароль признаются допустимыми.
5. Затем SRV3:
   a. Выясняет, известен ли Ivan.
   b. Запрашивает DC1, является ли пользователь Ivan администратором.
   c. Запрашивает у DC1 список всех групп пользователя Ivan.

Примечание: Поскольку клиент Windows 9x не может проверить имя пользователя и пароль, аутентификация типа client в Windows 9x не обеспечивает защиту. Если у компьютера Windows 9x есть доступ к провайдеру защиты Windows NT, то некоторый уровень защиты можно обеспечить, настроив систему Windows 9x для регистрации через проверенный промежуточный сервер. Более подробную информацию по этому вопросу можно найти в документации фирмы Microsoft по Windows 9x.

Понятия, связанные с данным:

- “DB2 for Windows NT и функции защиты Windows NT - Введение” на стр. 393

- “Поддержка глобальных групп (в Windows)” на стр. 397

Поддержка глобальных групп (в Windows)

DB2® также поддерживает глобальные группы. Чтобы использовать глобальные группы, нужно включить глобальные группы в какую-нибудь локальную группу. Когда DB2 получит список всех групп, членом которых является некоторый пользователь, она также создаст списки локальных групп, которым этот пользователь принадлежит косвенно (благодаря тому, что он входит в глобальную группу, являющуюся членом одной или нескольких локальных групп).

Глобальные группы применяются одним из двух способов:

- Включаются в локальную группу. Этой локальной группе могут быть предоставлены права доступа.
- Применяются в системе контроллера домена. В этом случае права доступа могут быть предоставлены глобальной группе.

Понятия, связанные с данным:

- “Аутентификация пользователей и групп в Windows NT” на стр. 399

Применение резервного контроллера домена в DB2

Процедура:

Если используемый вами сервер DB2 также играет роль резервного контроллера домена, вы можете повысить производительность DB2 и уменьшить сетевой трафик, если сконфигурируете DB2 для использования резервного контроллера домена.

Резервный контроллер домена можно задать в DB2 с помощью переменной реестра DB2DMNBCKCTLR.

Если вы знаете имя домена, для которого сервер DB2 является резервным контроллером домена, введите:

```
db2dmnbckctlr=<имя-домена>
```

где `имя-домена` должно быть задано в верхнем регистре.

Чтобы DB2 нашла домен, для которого локальный компьютер является резервным контроллером домена, введите:

```
DB2DMNBCKCTLR=?
```

Примечание: DB2 не использует существующий резервный контроллер домена по умолчанию, поскольку резервный контроллер домена может выйти из синхронизации с первичным контроллером домена, и из-за этого в защите возникнет брешь. Контроллеры доменов могут выходить из синхронизации, если база данных защиты первичного контроллера домена модифицируется, но эти изменения не распространяются на резервный контроллер домена. Это может случиться при задержках в сети или при неработоспособности службы Computer Browser.

Задачи, связанные с данной темой:

- “Установка DB2 на резервном контроллере домена” на стр. 400

Аутентификация типа user при помощи DB2 for Windows NT

Аутентификация типа user может создать проблемы для пользователей Windows NT из-за способа, которым операционная система проверяет права доступа. В этом разделе описаны некоторые особенности аутентификации типа user в DB2 for Windows NT:

- “Ограничения на имена пользователей и групп в DB2 for Windows NT”
- “Служба защиты DB2 for Windows NT” на стр. 400
- “Установка DB2 на резервном контроллере домена” на стр. 400
- “Аутентификация DB2 for Windows NT с применением функций защиты групп и доменов” на стр. 402

Ограничения на имена пользователей и групп в DB2 for Windows NT

В этой среде действуют следующие ограничения:

- Имена пользователей в DB2 ограничены 30 символами. Имена групп ограничены 8 символами.
- Имена пользователей в Windows NT не зависят от регистра (но пароли регистрозависимы).
- В именах пользователей и групп могут сочетаться символы верхнего и нижнего регистров. Однако обычно они преобразуются в верхний регистр при использовании в DB2. Так, если вы соединитесь с базой данных и создадите таблицу schema1.table1, в базе данных эта таблица будет сохранена как SCHEMA1.TABLE1. (Если вы хотите использовать имена объектов с символами нижнего регистра, вводите команды из процессора командной строки, заключая имена объектов в кавычки, или используйте инструменты ввода независимых разработчиков ODBC.)
- Пользователь может входить максимум в 64 группы.
- В DB2 применяется единое пространство имен. Это означает, что при работе в среде надежных доменов имя учетной записи пользователя должно быть

уникальным во всех доменах. Локальная база данных SAM компьютера сервера не должна содержать имен, определенных в других доменах.

Понятия, связанные с данным:

- “Аутентификация пользователей и групп в Windows NT” на стр. 399
- “Доверительные отношения между доменами Windows NT” на стр. 400

Аутентификация пользователей и групп в Windows NT

Пользователи определяются в Windows® NT путем создания учетных записей с помощью инструмента “Диспетчер пользователей”.

Учетная запись, содержащая в качестве элементов другие учетные записи, называется группой. Группы дают администраторам Windows NT возможность предоставлять права доступа всем пользователям некоторой группы одновременно. Учетные записи групп, как и учетные записи пользователей, определяются и обслуживаются с помощью базы данных Диспетчера прав доступа (SAM).

Есть два типа групп:

- Локальные группы. Локальная группа включает учетные записи пользователей, находящиеся в локальной базе данных учетных записей. Если локальная группа находится в системе, входящей в домен, она может также включать учетные записи и группы домена Windows NT. Если локальная группа находится на рабочей станции, она может содержать только учетные записи этой рабочей станции.
- Глобальные группы. Глобальная группа может существовать только на контроллере домена; она содержит учетные записи пользователей из базы данных SAM домена. Глобальная группа может включать только учетные записи пользователей из своего домена; она не может содержать в качестве элементов другие группы. Глобальная группа может применяться на серверах и рабочих станциях того же домена, а также доменов, доверяющих ему.

Понятия, связанные с данным:

- “Доверительные отношения между доменами Windows NT” на стр. 400
- “Поддержка глобальных групп (в Windows)” на стр. 397

Задачи, связанные с данной темой:

- “Аутентификация DB2 for Windows NT с применением функций защиты групп и доменов” на стр. 402

Ссылки, связанные с данной темой:

- “Ограничения на имена пользователей и групп в DB2 for Windows NT” на стр. 398

Доверительные отношения между доменами Windows NT

Доверительные отношения - это связь в управлении двумя доменами. Доверительные отношения между двумя доменами позволяют использовать учетные записи пользователей и групп одного домена в другом. Информация учетных записей совместно используется для проверки прав доступа учетных записей пользователей и групп доверенного домена без аутентификации. Доверительные отношения упрощают управление пользователями путем объединения нескольких доменов в одну административную единицу.

Доверительные отношения устанавливаются между двумя доменами:

- Доверяющий домен. Этот домен доверяет аутентификацию пользователей другому домену.
- Доверенный домен. Домен, аутентифицирующий пользователей для другого домена.

Доверительные отношения несимметричны. Это означает, что их нужно устанавливать в каждом направлении явным образом. Доверяющий домен не обязательно является доверенным.

Понятия, связанные с данным:

- “Аутентификация пользователей и групп в Windows NT” на стр. 399
- “Поддержка глобальных групп (в Windows)” на стр. 397

Ссылки, связанные с данной темой:

- “Ограничения на имена пользователей и групп в DB2 for Windows NT” на стр. 398

Служба защиты DB2 for Windows NT

В DB2® Universal Database служба аутентификации имен пользователей и паролей встроена в системный контроллер DB2. Служба защиты требуется только когда клиента соединен с сервером, настроенном на аутентификацию типа CLIENT.

Понятия, связанные с данным:

- “DB2 for Windows NT и функции защиты Windows NT - Введение” на стр. 393

Установка DB2 на резервном контроллере домена

Процедура:

В среде Windows NT 4.0 аутентификацию пользователей может выполнять главный или резервный контроллер домена. Эта возможность очень важна в больших распределенных локальных сетях с одним центральным главным

контроллером домена и одним или несколькими резервными контроллерами домена (BDC, backup domain controller) на каждом сайте. Тогда пользователи могут проходить аутентификацию на резервном контроллере домена на своем сайте, не вызывая для аутентификации главный контроллер домена (PDC, primary domain controller).

В этом случае преимущество применения резервного контроллера домена состоит в том, что аутентификация пользователей выполняется быстрее, и локальная сеть не перегружается.

Аутентификация может происходить на резервном контроллере домена при следующих условиях:

- Сервер DB2 for Windows NT установлен на резервном контроллере домена.
- Переменная DB2DMNBCKCTLR реестра профилей имеет соответствующее значение.

Если переменная DB2DMNBCKCTLR реестра профилей не задана или равна пустой строке, DB2 for Windows NT производит аутентификацию на главном контроллере домена.

Допустимыми объявленными значениями для DB2DMNBCKCTLR являются только “?” или имя домена.

Если значение переменной DB2DMNBCKCTLR реестра профилей - вопросительный знак (DB2DMNBCKCTLR=?), DB2 for Windows NT будет производить аутентификацию на резервном контроллере домена при следующих условиях:

- Значение реестра cachedPrimaryDomain задано для имени домена, которому принадлежит компьютер. (Это значение можно найти в **HKEY_LOCAL_MACHINE—> Software—> Microsoft—> Windows NT—> Current Version—> WinLogon**.)
- Менеджер сервера покажет, что резервный контроллер домена активен и доступен (то есть значок этого компьютера не затенен).
- Реестр сервера DB2 Windows NT указывает, что система - резервный контроллер домена для заданного домена.

В обычных обстоятельствах значение DB2DMNBCKCTLR=? работает, но так будет не во всех средах. Информация о серверах в домене динамически обновляется, и служба Computer Browser должна работать, чтобы поддерживать точность и соответствие этой информации текущему моменту. В больших LAN служба обозревателя может быть не запущена, поэтому у администратора сервера может быть устаревшая информация. В этом случае есть другой метод заставить DB2 for Windows NT производить аутентификацию на резервном контроллере домена: задать DB2DMNBCKCTLR=*xxx*, где *xxx* - имя домена

Windows NT для сервера DB2. При задании этого значения аутентификация будет производиться на резервном контроллере домена при следующих условиях:

- Значение реестра cachedPrimaryDomain задано для имени домена, которому принадлежит компьютер. (Это значение можно найти в **HKEY_LOCAL_MACHINE—> Software—> Microsoft—> Windows NT—> Current Version—> WinLogon**.)
- Компьютер настроен как резервный контроллер домена для заданного домена. (Если компьютер настроен как резервный контроллер другого домена, это значение приведет к ошибке.)

Задачи, связанные с данной темой:

- “Применение резервного контроллера домена в DB2” на стр. 397

Аутентификация DB2 for Windows NT с применением функций защиты групп и доменов

Процедура:

При предоставлении привилегий и определении уровней полномочий в DB2 UDB можно задать локальную или удаленную группу. Пользователь является членом группы, если его учетная запись явно определена в локальной или глобальной группе, либо если он входит в глобальную группу, которая является членом локальной группу.

DB2 for Windows NT поддерживает следующие типы групп:

- Локальные группы
- Глобальные группы
- Глобальные группы как члены локальных групп.

  DB2 for Windows NT получает список локальных и глобальных групп, членом которых является данный пользователь, используя базу данных защиты, в которой пользователь был найден. DB2 Universal Database позволяет переопределить эту стратегию и создавать список групп на локальном сервере Windows NT, на котором установлена DB2, независимо от того, где была найдена учетная запись пользователя. Такое переопределение стратегии достигается следующими командами:

  – Для глобальных параметров:

    ```
    db2set -g DB2_GRP_LOOKUP=local
    ```

  – Для параметров экземпляра:

    ```
    db2set -i <имя-экземпляра> DB2_GRP_LOOKUP=local
    ```

Для того чтобы изменение вступило в силу, после выполнения команды необходимо перезапустить экземпляр DB2. После этого создайте локальные группы и добавьте в них учетные записи доменов или глобальные группы.

Чтобы просмотреть все заданные переменные реестра профилей DB2, введите

```
db2set -all
```

Если переменной DB2_GRP_LOOKUP реестра профилей задано значение local, DB2 будет искать пользователя только на локальном компьютере. Если пользователь не будет найден на локальном компьютере или не окажется членом локальной или глобальной группы, он не пройдет аутентификацию. DB2 **не** выполняет поиск пользователя на других компьютерах домена и контроллерах домена.

Если переменная DB2_GRP_LOOKUP реестра профилей не задана, то:

1. DB2 вначале ищет пользователя на том же компьютере.
2. Если имя пользователя определено локально, пользователь пройдет аутентификацию локально.
3. Если пользователь не будет найден локально, DB2 начнет искать имя пользователя в его домене, а затем - на доверенных доменах.

Если программа DB2 запущена на компьютере, который является главным или резервным контроллером домена ресурсов, то можно найти любой контроллер домена в любом доверенном домене. Это связано с тем, что имена доменов резервных контроллеров доменов из доверенных доменов известны только контроллеру домена.

Если DB2 запущен не на контроллере домена, вызовите следующую команду:

```
db2set -g DB2_GRP_LOOKUP=DOMAIN
```

Эта команда указывает DB2, что для поиска имени контроллера домена учетной записи следует воспользоваться контроллером локального домена. То есть, если DB2 обнаружит, что учетная запись пользователя определена в домене x, он отправит запрос контроллеру собственного домена, вместо того чтобы пытаться найти контроллер домена x. В результате имя контроллера домена учетной записи будет найдено и возвращено на тот компьютер, на котором запущен DB2. У такого способа есть два преимущества:

1. Если главный контроллер домена недоступен, то будет найден резервный контроллер домена.
2. Если есть резервный контроллер домена, расположенный ближе, чем главный контроллер домена, то будет возвращено его имя.

Поддержка функций защиты доменов в DB2 for Windows NT

В следующем примере показано, как DB2 for Windows NT может работать с защитой домена. В первом примере соединение происходит благодаря тому, что имя пользователя и локальная группа находятся в одном и том же домене. Во втором примере соединение не происходит, поскольку имя пользователя и локальная или глобальная группа находятся в разных доменах.

Пример успешного соединения: В следующем сценарии соединение происходит благодаря тому, что имя пользователя и локальная или глобальная группа находятся в одном и том же домене.

Обратите внимание на то, что имя пользователя и локальная или глобальная группа не обязательно должны быть определены именно в том домене, где работает сервер базы данных; нужно только, чтобы имя и группа находились в одном домене.

Таблица 48. Успешное соединение с использованием контроллера домена

Domain1	**Domain2**
Существует доверенное соединение с доменом Domain2.	• Существует доверенное соединение с доменом Domain1. • Определена локальная или глобальная группа `grp2`. • Определено имя пользователя `id2`. • Имя пользователя `id2` входит в группу `grp2`.

Сервер DB2 запущен в данном домене. Он выполняет следующие команды DB2:
`REVOKE CONNECT ON db FROM public`
`GRANT CONNECT ON db TO GROUP grp2`
`CONNECT TO db USER id2`	

Происходит поиск в локальном или глобальном домене, но `id2` не найден. Проводится поиск в защите домена.	
	Имя пользователя `id2` найдено в этом домене. DB2 получает дополнительную информацию об этом имени пользователя (то есть о том, что оно входит в группу `grp2`).
Соединение происходит благодаря тому, что имя пользователя и локальная или глобальная группа находятся в одном и том же домене.	

Приложение F. Работа с монитором производительности Windows

Монитор производительности Windows - Введение

В DB2® for Windows® предусмотрено два монитора производительности:

- **Монитор производительности DB2**

  Монитор производительности DB2 создает снимки и сохраняет информацию о событиях, связанных с DB2 и DB2 Connect™. (Чтобы получить более подробную информацию, нажмите кнопку **Справка** в Центре управления и посмотрите электронную справку С чего начать.)

- **Монитор производительности Windows**

  Монитор производительности Windows позволяет отслеживать производительность базы данных и системы и получать информацию из любых источников статистических данных, зарегистрированных в системе. Windows предоставляет статистическую информацию обо всех аспектах работы компьютера, включая:

  – Занятость процессора
  – Использование памяти
  – Активность дисков
  – Активность сети

Задачи, связанные с данной темой:

- “Регистрация DB2 в мониторе производительности Windows” на стр. 405
- “Предоставление доступа к удаленной информации о производительности DB2” на стр. 406
- “Просмотр значений производительности DB2 и DB2 Connect” на стр. 407
- “Доступ к удаленной информации о производительности DB2” на стр. 409
- “Сброс значений производительности DB2” на стр. 409

Ссылки, связанные с данной темой:

- “Объекты производительности Windows” на стр. 408

Регистрация DB2 в мониторе производительности Windows

Процедура:

Программа установки автоматически регистрирует DB2 в мониторе производительности Windows.

Для того чтобы информация о производительности DB2 и DB2 Connect стала доступной для монитора производительности Windows, зарегистрируйте DLL для счетчиков производительности DB2 for Windows. Одновременно данные производительности станут доступны для всех прикладных программ Windows, использующих API производительности Win32.

Для того чтобы установить и зарегистрировать DLL счетчиков производительности DB2 for Windows (DB2Perf.DLL) в мониторе производительности Windows, введите:

```
db2perfi -i
```

Регистрация DLL также создает новый ключ в разделе служб реестра. Одна запись задает имя модуля DLL, обеспечивающего поддержку счетчиков. Еще три записи задают имена функций, поддерживаемых внутри DLL. Это следующие функции:

- **Open**

  Вызывается, когда DLL впервые загружается системой в некотором процессе.
- **Collect**

  Вызывается для запроса информации производительности от DLL.
- **Close**

  Вызывается, когда DLL выгружается.

Ссылки, связанные с данной темой:

- “db2perfi - Performance Counters Registration Utility Command” в *Command Reference*

Предоставление доступа к удаленной информации о производительности DB2

Процедура:

Если рабочая станция DB2 for Windows подключена по сети к другим компьютерам Windows, вы можете выполнить описанную ниже процедуру.

Для того чтобы просмотреть объекты производительности Windows, хранящиеся на другом компьютере DB2 for Windows, зарегистрируйте в DB2 имя и пароль администратора. (Имя пользователя монитора производительности Windows по умолчанию равно **SYSTEM**. Это имя зарезервировано в DB2 и его нельзя использовать.) Чтобы зарегистрироваться под нужным именем, введите:

```
db2perfr -r имя_пользователя пароль
```

Примечание: Используемое `имя_пользователя` должно отвечать правилам именования DB2.

Информация об имени пользователя и пароле хранится в ключе в реестре с защитой, которая разрешает доступ только администраторам и учетной записи SYSTEM. Эти данные шифруются, чтобы обезопасить хранение пароля администратора в реестре.

Примечания:

1. После того, как сочетание имени пользователя и пароля зарегистрированы в DB2, даже локальные экземпляры монитора производительности будут явно регистрироваться при помощи этого имени пользователя и пароля. Это значит, что если информация об имени пользователя, зарегистрированного в DB2, не совпадет, локальные сеансы монитора производительности не дадут информации производительности DB2.
2. Указанные имя пользователя и пароль должны совпадать с именем пользователя и паролем, хранящимися в базе данных защиты Windows. При изменении этих значений в базе данных защиты Windows нужно сбросить имя пользователя и пароль, применяемые для удаленного отслеживания производительности.
3. Для отмены регистрации введите:
   ```
   db2perfr -u <имя_пользователя> <пароль>
   ```

Понятия, связанные с данным:

- “Правила именования” на стр. 321

Ссылки, связанные с данной темой:

- “db2perfr - Performance Monitor Registration Tool Command” в *Command Reference*

Просмотр значений производительности DB2 и DB2 Connect

Процедура:

Чтобы вывести значения производительности DB2 и DB2 Connect при помощи монитора производительности, достаточно просто выбрать нужные вам счетчики производительности из подокна **Добавить к**. В этом подокне выводится список объектов производительности, поставляющих данные производительности. Выберите объект, чтобы увидеть список его счетчиков.

Объект производительности может существовать в нескольких экземплярах. Например, объект LogicalDisk поддерживает такие счетчики, как “Процент

времени чтения диска” и “Скорость диска в байтах/с”; у этих счетчиков есть по экземпляру для каждого логического диска компьютера, включая “C:” и “D:”.

Объекты производительности Windows

В Windows предусмотрены следующие объекты производительности::

- **Менеджер баз данных DB2**

  Этот объект предоставляет общую информацию об отдельном экземпляре Windows. Экземпляр DB2, за котором ведется мониторинг, выводится как экземпляр объекта.

  Из соображений удобства и производительности в данный момент времени вы можете получить информацию о производительности только от одного экземпляра DB2. Экземпляр DB2, который показывает монитор производительности, определяется переменной `db2instance` реестра в процессе монитора производительности. Если у вас одновременно работают несколько экземпляров DB2, и вы хотите увидеть информацию о производительности нескольких экземпляров, нужно запустить отдельные сеансы монитора производительности, в которых `db2instance` будет иметь соответствующие значения для каждого наблюдаемого экземпляра DB2.

  Если у вас работает система распределенных баз данных, в данный момент времени вы можете получить информацию о производительности только от одного сервера раздела баз данных (узла). По умолчанию выводится информация о производительности узла по умолчанию (то есть узла с логическим портом 0). Чтобы увидеть информацию о другом узле, нужно запустить отдельные сеансы монитора производительности, в которых переменной среды DB2NODE будут заданы значения номеров наблюдаемых узлов.

- **Базы данных DB2**

  Этот объект обеспечивает информацию о конкретной базе данных. Информация доступна по каждой активной в настоящей момент базе данных.

- **Программы DB2**

  Этот объект обеспечивает информацию о конкретной программе DB2. Информация доступна по каждой активной в настоящей момент программе DB2.

- **Базы данных DB2**

  Этот объект обеспечивает информацию о конкретной базе данных DCS. Информация доступна по каждой активной в настоящей момент базе данных.

- **Программы DB2 DCS**

  Этот объект обеспечивает информацию о конкретной программе DCS DB2. Информация доступна по каждой активной в настоящей момент программе DCS DB2.

Список объектов монитора производительности Windows зависит от того, какое программное обеспечение установлено на компьютере Windows, и какие программы активны в данный момент. Например, если установлена DB2 UDB и запущен менеджер баз данных, в списке будет объект менеджер баз данных DB2. Если на этом компьютере в настоящий момент активны какие-то базы данных и прикладные программы DB2, в списке также будут объекты баз данных DB2 и программ DB2. Если система Windows применяется в качестве шлюза DB2 Connect, и в настоящий момент активны некоторые базы данных и приложения DCS, то будут показаны объекты баз данных DCS DB2 и приложений DCS DB2.

Понятия, связанные с данным:

- "Переменные реестра и среды DB2" в *Руководство администратора: Производительность*

Доступ к удаленной информации о производительности DB2

Процедура:

Предоставление доступа к удаленной информации производительности DB2 описано выше. В подокне **Добавить к** выберите другой компьютер, который нужно подвергнуть мониторингу. Появится список всех доступных объектов производительности на этом компьютере.

Для мониторинга объекта производительности DB2 на удаленном компьютере на нем должны быть установлены программы DB2 UDB или DB2 Connect Версии 6 или более новой.

Сброс значений производительности DB2

Процедура:

Когда прикладная программа вызывает интерфейсы API монитора DB2, возвращаемая информация обычно представляет собой суммарные значения, накопленные с момента запуска сервера DB2. Но нередко требуется:

- Обнулить значения производительности
- Выполнить тест
- Снова обнулить значения производительности
- Повторить тест.

Для сброса значений производительности используйте программу **db2perfc**.Введите:

```
db2perfc
```

По умолчанию эта программа сбросит значения производительности для всех активных баз данных DB2. Однако вы можете задать список баз данных, для которых нужно сбросить значения. Кроме того, с помощью опции -d можно задать сброс значений производительности для баз данных DCS. Например:

```
db2perfc
db2perfc dbalias1 dbalias2 ... dbaliasn

db2perfc  -d
db2perfc  -d  dbalias1 dbalias2 ... dbaliasn
```

В первом примере сбрасываются значения производительности для всех активных баз данных DB2. В следующем примере сбрасываются значения для конкретных баз данных DB2. В третьем примере сбрасываются значения производительности для всех активных баз данных DCS DB2. В последнем примере сбрасываются значения для конкретных баз данных DCS DB2.

Программа **db2perfc** сбрасывает значения для всех программ, которые в настоящее время работают с информацией о производительности базы данных соответствующего экземпляра сервера DB2 (то есть экземпляра, заданного в параметре DB2INSTANCE того сеанса, в котором запущена команда **db2perfc**).

Вызов **db2perfc** также сбросит значения, которые увидит всякий удаленный получатель информации производительности DB2, выполнив команду **db2perfc**.

Примечание: Существует API DB2 sqlmrset, с помощью которого приложение может сбросить значения, видимые на локальном уровне, для отдельных баз данных.

Ссылки, связанные с данной темой:

- “db2ResetMonitor - Reset Monitor” в *Administrative API Reference*
- “db2perfc - Reset Database Performance Values Command” в *Command Reference*

Приложение G. Работа с серверами разделов баз данных Windows

Задачи по изменению характеристик конфигурации в среде Windows решаются при помощи специальных утилит.

Ниже описаны следующие утилиты:

- “Просмотр списка серверов раздела базы данных в экземпляре”
- “Добавление сервера раздела базы данных к экземпляру (Windows)” на стр. 412
- “Изменение раздела базы данных (Windows)” на стр. 414
- “Отбрасывание раздела базы данных из экземпляра (Windows)” на стр. 415

Просмотр списка серверов раздела базы данных в экземпляре

Процедура:

В Windows для получения списка серверов раздела базы данных, относящихся к экземпляру, можно воспользоваться командой **db2nlist**.

Использование команды:

```
db2nlist
```

При этом подразумевается текущий экземпляр (заданный переменной среды DB2INSTANCE). Чтобы задать нужный экземпляр, введите:

```
db2nlist /i:имяЭкз
```

где `имяЭкз` - имя конкретного экземпляра.

По желанию можно запросить состояния всех серверов разделов при помощи команды:

```
db2nlist /s
```

Сервер разделов может иметь одно из следующих состояний: запускается, запущен, останавливается, остановлен.

Задачи, связанные с данной темой:

- “Добавление сервера раздела базы данных к экземпляру (Windows)” на стр. 412
- “Изменение раздела базы данных (Windows)” на стр. 414

• “Отбрасывание раздела базы данных из экземпляра (Windows)” на стр. 415

Добавление сервера раздела базы данных к экземпляру (Windows)

Процедура:

В Windows для добавления сервера раздела базы данных (узла) к экземпляру служит команда **db2ncrt**.

Примечание: Не используйте команду **db2ncrt**, если этот экземпляр уже содержит базы данных. В этом случае следует использовать команду **db2start addnode**. Это обеспечит правильное добавление базы данных к новому серверу разделов баз данных. **НЕ РЕДАКТИРУЙТЕ** файл `db2nodes.cfg` - изменение этого файла может привести к несогласованности в системе распределенных баз данных.

Обязательные параметры этой команды:

```
db2ncrt /n:номер_узла
        /u:имя-пользователя,пароль
        /p:логический_порт
```

- /n:

  Уникальный номер узла для обозначения этого сервера разделов баз данных. Используются числа от 1 до 999 в порядке возрастания.

- /u:

  Имя и пароль учетной записи регистрации службы DB2.

- /p:логический_порт

  Номер логического порта, используемого для сервера разделов баз данных, если логический порт не ноль (0). Если параметр не задан, предполагается логический порт 0.

Параметр логического порта необязателен только при создании первого узла на компьютере. Если вы создаете логический узел, необходимо задать этот параметр, выбрав не используемый сейчас номер логического порта. Есть несколько ограничений:

- На каждом компьютере должен быть сервер разделов баз данных с логическим портом 0.
- Номер порта не должен выходить за пределы диапазона номеров портов, зарезервированного для соединений FCM в файле служб из каталога `%SystemRoot%\system32\drivers\etc`. Например, если для текущего экземпляра отведен диапазон из четырех портов, максимальный номер порта

будет 3 (порты 1, 2 и 3; порт 0 - для логического узла по умолчанию). Диапазон портов определяется командой **db2icrt** с параметром `/r:начальный_порт, конечный_порт`.

Кроме того, есть несколько необязательных параметров:

- /g:имя_сети

  Задает имя сети для сервера разделов баз данных. Если не задать этот параметр, DB2 будет использовать первый IP-адрес, который обнаружит в вашей системе.

  Используйте этот параметр, если на вашем компьютере несколько IP-адресов, и вы хотите задать конкретный IP-адрес для сервера разделов баз данных. Ввести параметр *имя_сети* можно, используя имя сети или IP-адрес.

- /h:имя_хоста

  Имя хоста TCP/IP, используемое FCM для внутренней связи, если имя хоста - не имя локального хоста. Этот параметр обязателен, если вы добавляете сервер разделов баз данных на удаленный компьютер.

- /i:имя_экземпляра

  Имя экземпляра; по умолчанию - текущий экземпляр.

- /m:имя_компьютера

  Имя компьютера рабочей станции Windows, на которой расположен узел. По умолчанию применяется имя локального компьютера.

- /o:компьютер_владелец_экземпляра

  Имя компьютера, владеющего экземпляром; по умолчанию - локальный компьютер. Этот параметр - обязательный, когда команда **db2ncrt** выдается не на компьютере, владеющем экземпляром.

Например, если вы хотите добавить новый сервер разделов баз данных к экземпляру TESTMPP (и, следовательно, у вас работает несколько логических узлов) на компьютере MYMACHIN, владеющем экземпляром, и вы хотите, чтобы этот новый узел был известен как узел 2, использующий логический порт 1, введите:

```
db2ncrt /n:2 /p:1 /u:мой_id,мой_прл /i:TESTMPP
   /M:TEST /o:MYMACHIN
```

Ссылки, связанные с данной темой:

- “db2start - Start DB2 Command” в *Command Reference*
- “db2icrt - Create Instance Command” в *Command Reference*
- “db2ncrt - Add Database Partition Server to an Instance Command” в *Command Reference*

Изменение раздела базы данных (Windows)

Процедура:

В Windows с помощью команды **db2nchg** можно выполнить следующие действия:

- Переместить раздел базы данных с одного компьютера на другой.
- Изменить у компьютера имя хоста TCP/IP.

  Если вы планируете использовать несколько сетевых адаптеров, с помощью этой команды нужно будет задать адрес TCP/IP для поля “netname” в файле *db2nodes.cfg*.
- Использовать другой номер логического порта.
- Использовать другое имя для сервера раздела базы данных (узла).

Обязательные параметры этой команды:

```
db2nchg /n:номер_узла
```

Параметр /n: - номер узла конфигурации сервера разделов баз данных, которую вы хотите изменить. Это обязательный параметр.

Необязательные параметры:

- /i:имя_экземпляра

  Задает экземпляр, в который входит этот сервер разделов баз данных. Если не указать этот параметр, по умолчанию принимается текущий экземпляр.
- /u:имя_пользователя,пароль

  Изменяет имя учетной записи регистрации и пароль для службы DB2. Если не указать этот параметр, учетная запись регистрации и пароль не изменятся.
- /p:логический_порт

  Изменяет логический порт для сервера разделов баз данных. Этот параметр надо задать, если вы перемещаете сервер разделов баз данных на другой компьютер. Если не указать этот параметр, номер логического порта останется прежним.
- /h:имя_хоста

  Изменяет имя хоста TCP/IP, используемое FCM для внутренней связи. Если не указать этот параметр, имя хоста останется прежним.
- /m:имя_компьютера

  Перемещает сервер разделов баз данных на другой компьютер. Сервер разделов баз данных можно перемещать, только если в экземпляре нет ни одной существующей базы данных.
- /g:сетевое_имя

  Изменяет сетевое имя для сервера разделов баз данных.

Используйте этот параметр, если на вашем компьютере несколько IP-адресов, и вы хотите использовать конкретный IP-адрес для сервера разделов баз данных. В качестве `сетевого_имени` можно задать имя или IP-адрес.

Например, чтобы изменить логический порт, назначенный узлу 2, входящему в экземпляр TESTMPP, чтобы он использовал логический порт 3, введите команду:

```
db2nchg /n:2 /i:TESTMPP /p:3
```

DB2 позволяет обращаться к переменным реестра DB2 уровня экземпляра с удаленного компьютера. В настоящее время переменные реестра DB2 хранятся на трех различных уровнях: уровне компьютера или глобальном уровне, уровне экземпляра и уровне узла. Переменные реестра, сохраненные на уровне экземпляра (включая уровень узла) можно перенаправить на другой компьютер, используя переменную DB2REMOTEPREG. Если переменная DB2REMOTEPREG задана, DB2 будет обращаться к переменным реестра DB2 с компьютера, указанного в этой переменной. Команда db2set должна выглядеть следующим образом:

```
db2set DB2REMOTEPREG=<удаленная рабочая станция>
```

где <удаленная рабочая станция> - это имя удаленной рабочей станции.

Примечание: Будьте осторожны, задавая эту опцию, поскольку все профили и списки экземпляров DB2 будут расположены на указанном удаленном компьютере.

Эту переменную можно использовать вместе с переменной DBINSTPROF, задающей удаленный диск локальной сети на том же компьютере, где находится реестр.

Понятия, связанные с данным:

- “Переменные реестра и среды DB2” в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- “db2nchg - Change Database Partition Server Configuration Command” в *Command Reference*

Отбрасывание раздела базы данных из экземпляра (Windows)

Процедура:

В Windows для отбрасывания сервера раздела базы данных (узла) из экземпляра, не содержащего баз данных, служит команда **db2ndrop**. Если вы отбросите сервер разделов баз данных, его номер узла может вновь использоваться для нового сервера разделов баз данных.

Отбрасывая серверы разделов баз данных из экземпляра, соблюдайте осторожность. Если вы удалите из экземпляра узел ноль (0) сервера разделов баз данных, владеющего экземпляром, экземпляр станет недоступен. Если нужно отбросить экземпляр, используйте команду **db2idrop**.

Примечание: Не используйте команду **db2ndrop**, если экземпляр содержит базы данных.Вместо этого используйте команду **db2stop drop nodenum**.Это обеспечит корректное удаление базы данных из разделы базы данных. **НЕ РЕДАКТИРУЙТЕ** файл `db2nodes.cfg` - изменение этого файла может привести к несогласованности в системе распределенных баз данных.

Перед тем как отбросить узел, применяющий логический порт 0, на компьютере с несколькими логическими узлами, необходимо отбросить все остальные узлы, применяющие другие логические порты. На любом сервере раздела базы данных есть узел, применяющий логический порт 0.

Параметры этой команды:

```
db2ndrop /n:номер_узла /i:имя_экземпляра
```

- /n:

  Уникальный номер узла для обозначения этого сервера разделов баз данных. Это обязательный параметр. Используются числа от нуля (0) до 999 в порядке возрастания. Помните, что узел ноль (0) представляет компьютер - владелец экземпляра.

- /i:имя_экземпляра

  Имя экземпляра. Это необязательный параметр. Если он не задан, подразумевается текущий экземпляр (заданный переменной реестра DB2INSTANCE).

Понятия, связанные с данным:

- “Переменные реестра и среды DB2” в *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

- “db2stop - Stop DB2 Command” в *Command Reference*
- “db2idrop - Remove Instance Command” в *Command Reference*
- “db2ndrop - Drop Database Partition Server from an Instance Command” в *Command Reference*

Приложение H. Конфигурирование нескольких логических узлов

Применение нескольких логических узлов

Обычно DB2® UDB Enterprise Server Edition настраивается таким образом, чтобы на каждом компьютере находился один сервер раздела базы данных. Но есть несколько ситуаций, когда выгодно иметь несколько серверов разделов баз данных на одном компьютере. Это означает, что в конфигурации узлов может быть больше, чем компьютеров. Если эти узлы входят *в один и тот же* экземпляр, говорят, что на компьютере работает *несколько логических узлов*. Если же они принадлежат разным экземплярам, этот компьютер *не* называют компьютером с несколькими логическими узлами.

Поддержка нескольких логических узлов позволяет выбирать из трех типов конфигурации:

- Стандартная конфигурация, в которой на каждом компьютере есть только один сервер разделов баз данных.
- Конфигурация с несколькими логическими узлами, в которой на одном компьютере есть несколько серверов разделов баз данных.
- Конфигурация, в которой на каждом компьютере работают несколько логических узлов.

Конфигурации с несколькими логическими узлами полезны, когда система выполняет запросы на компьютере с симметрической многопроцессорной архитектурой (SMP). Возможность сконфигурировать несколько логических узлов на одном компьютере полезна также и в случае отказа компьютера. При отказе компьютера (вызывающем сбой одного или нескольких работающих на нем серверов разделов баз данных) вы можете перезапустить один или несколько серверов разделов баз данных на другом компьютере при помощи команды DB2START NODENUM. Благодаря этому пользовательские данные останутся доступными.

Другим преимуществом является то, что несколько логических узлов могут использовать аппаратную конфигурацию SMP. Кроме того, уменьшаются разделы баз данных, за счет чего можно повысить производительность для таких задач, как резервное копирование и восстановление разделов баз данных и табличных пространств, а также создание индексов.

Задачи, связанные с данной темой:

- “Настройка нескольких логических узлов” на стр. 418

Ссылки, связанные с данной темой:

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

Настройка нескольких логических узлов

Процедура:

Есть два способа сконфигурировать несколько логических узлов:

- Сконфигурируйте логические узлы (разделы баз данных) в файле `db2nodes.cfg`. Тогда вы сможете запускать все логические узлы и удаленные узлы командой DB2START или связанными с ней API.

  Примечание: В Windows NT нужно добавить узел при помощи *db2ncrt*, если в системе нет баз данных, или ввести команду DB2START ADDNODE, если есть одна или несколько баз данных. В среде Windows NT ни в коем случае нельзя редактировать файл *db2nodes.cfg* вручную.

- Перезапустить логический узел на другом процессоре, на котором уже запущены другие логические разделы баз данных (узлы). Это позволит переопределить имя хоста и номер порта, заданные для логического раздела базы данных в `db2nodes.cfg`.

Чтобы сконфигурировать логический раздел базы данных (узел) в `db2nodes.cfg`, в этом файле нужно создать запись, которая назначит номер логического порта для узла. Используется следующий синтаксис:

```
номер_узла имя_хоста логический_порт сетевое_имя
```

Примечание: В Windows NT нужно добавить узел при помощи *db2ncrt*, если в системе нет баз данных, или ввести команду DB2START ADDNODE, если есть одна или несколько баз данных. В среде Windows NT ни в коем случае нельзя редактировать файл *db2nodes.cfg* вручную.

Формат файла *db2nodes.cfg* в Windows NT отличается от формата такого же файла в Unix. В Windows NT используется следующий формат столбца:

```
номер_узла имя_хоста имя_компьютера логический_порт сетевое_имя
```

Нужно убедиться, что в файле `services` в каталоге `etc` для связи FCM задано достаточно портов.

Понятия, связанные с данным:

- “Изменение файла конфигурации узла” на стр. 175
- “Применение нескольких логических узлов” на стр. 417

Задачи, связанные с данной темой:

- “Создание файла конфигурации узлов” на стр. 40

Ссылки, связанные с данной темой:

- “db2start - Start DB2 Command” в *Command Reference*
- “db2ncrt - Add Database Partition Server to an Instance Command” в *Command Reference*

Приложение I. Расширение Центра управления

Встраиваемые модули Центра управления - Введение

Набор функций Центра управления DB2 Universal Database можно расширить с помощью *встраиваемых модулей*.

Архитектура *встраиваемых модулей* позволяет добавлять компоненты во всплывающее меню Центра управления для заданного объекта, объекты в дерево Центра управления и кнопки на панель инструментов. Набор необходимых для этого интерфейсов Java™ поставляется вместе с инструментами. Эти интерфейсы используются, чтобы сообщить Центру управления, какие дополнительные действия надо добавить.

Понятия, связанные с данным:

- “Влияние встраиваемых модулей Центра управления на производительность” на стр. 421
- “Компиляция и запуск примеров встраиваемых модулей” на стр. 422
- “Создание встраиваемых модулей - расширений Центра управления” на стр. 424
- “Рекомендации для разработчиков встраиваемых модулей Центра управления” на стр. 421

Влияние встраиваемых модулей Центра управления на производительность

Встраиваемые модули (db2plug.zip) загружаются во время запуска инструментов Центра управления. Это может увеличить время запуска инструментов в зависимости от размера ZIP-файла, однако можно ожидать, что ZIP-файл для большинства пользователей окажется небольшим, и его влияние на время загрузки будет минимальным.

Рекомендации для разработчиков встраиваемых модулей Центра управления

Поскольку файл db2plug.zip может содержать несколько встраиваемых модулей, разработчики встраиваемых модулей Центра управления должны придерживаться следующих рекомендаций:

- Для того чтобы гарантировать уникальность имен классов воспользуйтесь пакетами Java™. Следуйте соглашениям об именах пакетов Java. Имя пакета должно начинаться с имени домена Internet в обратном порядке (например,

com.companyname). Все имена пакетов (или, по крайней мере, их уникальные префиксы) должны состоять из символов нижнего регистра.

- Файл db2plug.zip должен быть установлен в подкаталоге tools каталога sqllib. До версии V8 файл db2plug.zip должен был быть установлен в подкаталоге cc каталога sqllib.
- При создании встраиваемого модуля Центра управления, если файл db2plug.zip уже существует, следует добавить свои классы в этот существующий файл. Не следует заменять существующий файл db2plug.zip своим файлом. Для добавления встраиваемого модуля в существующий файл db2plug.zip введите следующую команду:

  ```
  zip -r0 db2plug.zip com\companyname\myplugin\*.class
  ```

  где имя пакета встраиваемого модуля - com.companyname.myplugin
- При запуске Центра управления загружаются все классы в файле db2plug.zip. Файл db2plug.zip должен содержать все файлы подкласса CCExtension. Файлы классов, не являющихся подклассом CCExtension, но нужных встраиваемому модулю, не обязательно должны находиться в файле db2plug.zip. Для ускорения запуска Центра управления они могут находиться в отдельном файле jar. При большом числе таких классов рекомендуется поместить их в отдельный файл. Этот файл должен находиться в подкаталоге tools каталога sqllib. Файл jar будет автоматически включен в *classpath* при запуске Центра управления командой **db2cc**.

Понятия, связанные с данным:

- “Компиляция и запуск примеров встраиваемых модулей” на стр. 422
- “Создание встраиваемых модулей - расширений Центра управления” на стр. 424

Ссылки, связанные с данной темой:

- “db2cc - Start Control Center Command” в *Command Reference*

Компиляция и запуск примеров встраиваемых модулей

Функция встраиваемых модулей Центра управления описана в следующих разделах. Возможности этой функции продемонстрированы на примере следующих программ встраиваемых модулей: Example1.java, Example2.java, Example3.java, Example3Folder.java, Example4.java, Example5.java и Example6.java. Файлы java примеров программ устанавливаются вместе с клиентом разработки программ DB2®. В операционной системе Windows® эти примеры программ помещаются в каталог Диск:\sqllib\samples\java\plugin, где Диск: - это диск, на котором установлен DB2. На платформе UNIX® эти примеры помещаются в каталог /u/db2inst1/sqllib/samples/java/plugin, где /u/db2inst1 - это каталог, в котором установлен DB2.

Для запуска примеров встраиваемых модулей необходимо упаковать файлы классов расширения с помощью ZIP в соответствии с правилами создания архивных файлов Java™. Имя файла ZIP (db2plug.zip) необходимо добавить в значение переменной *classpath*. В операционной системе Windows поместите файл db2plug.zip в каталог `Диск:\sqllib\tools`, где `Диск:` - это диск, на котором установлен DB2. На платформе UNIX поместите файл db2plug.zip в каталог `/u/db2inst1/sqllib/tools`, где `/u/db2inst1` - это каталог, в котором установлен DB2.

Примечание: Команда **db2cc** добавляет путь к файлу db2plug.zip из каталога инструментов в переменную *classpath*.

Не следует помещать в один архивный файл db2plug.zip несколько примеров, так как это может привести к конфликту (исключение составляют примеры Example3 и Example3Folder, которые применяются совместно).

Для того чтобы скомпилировать файлы java, содержащие примеры, необходимо добавить следующие имена в переменную *classpath*:

- На платформах Windows:
  - `Диск: \sqllib\java\Common.jar`
  - `Диск: \sqllib\tools\db2cc.jar`

  где `Диск` - диск, на котором установлена система DB2.
- На платформах UNIX:
  - `/u/db2inst1/sqllib/java/Common.jar`
  - `/u/db2inst1/sqllib/tools/db2cc.jar`

  где `/u/db2inst1` - каталог, в котором установлен DB2.

Создайте файл db2plug.zip, содержащий все классы, созданные в результате компиляции файла java примера. Файл не должны быть сжатым. Например:

```
zip -r0 db2plug.zip *.class
```

Эта команда помещает все файлы классов в файл db2plug.zip с сохранением информации об относительных путях.

Понятия, связанные с данным:

- “Создание встраиваемых модулей - расширений Центра управления” на стр. 424
- “Рекомендации для разработчиков встраиваемых модулей Центра управления” на стр. 421

Ссылки, связанные с данной темой:

- “db2cc - Start Control Center Command” в *Command Reference*

Создание встраиваемых модулей - расширений Центра управления

Первый этап при создании встраиваемого модуля - определение класса, реализующего интерфейс CCExtension. Этот класс будет содержать список классов встраиваемых модулей, загружаемых Центром управления. Для добавления элементов меню к стандартным объектам Центра управления, таким как Базы данных и Таблицы, или создания собственных объектов дерева, нужно создать классы, реализующие интерфейс CCObject, и вернуть массив объектов CCObject в методе getObjects. Для добавления кнопки на панель инструментов необходимо реализовать класс CCToolbarAction и вернуть массив объектов CCToolbarAction в методе getToolbarActions.

Каждый из этих интерфейсов документирован в:

- На платформах Windows® - в файле `X:\sqllib\samples\java\plugin\doc`, где `X:` - устройство установки DB2®.
- На платформах UNIX® - в файле `/u/db2inst1/sqllib/samples/java/plugin/doc`, где `/u/db2inst1` - каталог установки DB2.

Задачи, связанные с данной темой:

- “Создание расширения, добавляющего кнопку на панель инструментов” на стр. 425
- “Добавление действий в основное меню” на стр. 426
- “Изменение расположения пункта меню” на стр. 428
- “Создание разделителя в основном меню” на стр. 429
- “Создание подменю” на стр. 430
- “Добавление пункта меню для объекта с указанным именем” на стр. 431
- “Добавление папки для хранения нескольких объектов в дерево” на стр. 431
- “Добавление примера объекта в папку” на стр. 434
- “Настройка атрибутов нового объекта дерева” на стр. 435
- “Добавление команды создания” на стр. 437
- “Добавление команды удаления с возможностью выбора нескольких объектов” на стр. 439
- “Добавление команды изменения” на стр. 440
- “Отключение функции настройки с помощью метода isConfigurable()” на стр. 442
- “Отключение возможности изменять объекты с помощью метода isEditable()” на стр. 442
- “Отключение кнопок значений по умолчанию в окнах настройки с помощью метода hasConfigurationDefaults()” на стр. 443

Задачи подключения модулей

В разделе описаны следующие задачи, связанные с подключаемыми модулями:

1. Создание модуля, добавляющего кнопку панели инструментов
2. Создание модуля, добавляющего пункты меню в объект базы данных
3. Создание модуля, который добавляет объекты модулей в дерево базы данных
4. Выключение возможности настройки с помощью isConfigurable()
5. Выключение возможности изменения объектов с помощью isEditable()
6. Выключение кнопок по умолчанию в диалоговых окнах настройки с помощью hasConfigurationDefaults()

Создание расширения, добавляющего кнопку на панель инструментов

Процедура:

Ниже приведен пример добавления кнопки на панель инструментов; getObjects возвращает пустой массив:

```
import com.ibm.db2.tools.cc.navigator.*;
import java.awt.event.*;
import javax.swing.*;

public class Example1 implements CCExtension {

  public CCObject[] getObjects () {
    return null;
  }

}
```

Обратите внимание, что импортируется пакет com.ibm.db2.tools.cc.navigator. Этот класс реализует интерфейс CCToolbarAction, в котором предусмотрено три метода: getHoverHelpText, getIcon и actionPerformed. Центр управления получает с помощью метода getHoverHelpText текст, который выводится в небольшом прямоугольнике при помещении указателя мыши на кнопку панели инструментов. Метод getIcon задает значок кнопки. Метод actionPerformed вызывается Центром управления при нажатии кнопки. В приведенном ниже примере добавляется кнопка X, при нажатии которой выводится сообщение на консоль. Для этой кнопки применяется значок Refresh из класса хранилища изображений Центра управления.

```
class Example1ToolbarAction implements CCToolbarAction {

  public String getHoverHelpText() { return "X"; }

  public ImageIcon getIcon() {
   return CommonImageRepository.getCommonIcon(CommonImageRepository.WC_NV_
REFRESH);
  }
```

```
  public void actionPerformed(ActionEvent e) {
   System.out.println("I've been clicked");
  }

}
```

Наконец, в классе Example1 необходимо реализовать метод getToolbarActions, возвращающий экземпляр нового класса:

```
  public CCToolbarAction[] getToolbarActions () {
   return new CCToolbarAction[] { new Example1ToolbarAction() };
  }
```

Понятия, связанные с данным:

- “Компиляция и запуск примеров встраиваемых модулей” на стр. 422

Создание модуля, добавляющего пункты меню в объект базы данных

Ниже описана процедура создания модуля, добавляющего пункты меню в объект базы данных:

1. Создание действия основного меню
2. Размещение пункта меню
3. Создание разделителя в основном меню
4. Создание подменю
5. Создание пункта меню только для объекта с определенным именем

Добавление действий в основное меню

Процедура:

В этом более сложном примере добавляются новые команды в выпадающее меню объекта Database.

Как и в примере Example 1, вначале нужно создать класс, расширяющий CCExtension.

```
import com.ibm.db2.tools.cc.navigator.*;
import java.awt.event.*;
import javax.swing.*;

public class Example2 implements CCExtension {

  public CCToolbarAction[] getToolbarActions () {
    return null;
  }

}
```

Далее необходимо создать объект CCObject для объекта Database в дереве:

```
class CCDatabase implements CCObject {

  public String getName () { return null; }
  public boolean isEditable () { return true; }
  public boolean isConfigurable () { return true; }

  public int getType () { return UDB_DATABASE; }
}
```

Поскольку в этом примере просто добавляются пункты в меню встроенного объекта Центра управления (объекта Database), большинство функций возвращают значение null или true. Тип UDB_DATABASE (константа в CCObject) указывает, что данный объект представляет объект Database DB2 UDB. В примере класс назван CCDatabase, однако вы можете присвоить классу любое уникальное имя. Следует учесть, что вместе с вашим расширением в файле zip могут находиться расширения других вендоров, и имена классов в этих расширениях не должны совпадать. Для того чтобы обеспечить уникальность имени класса, рекомендуется применять пакеты Java.

Метод getObjects класса CCExtension должен возвращать объект, содержащий экземпляр CCDatabase:

```
public CCObject[] getObjects () {
  return new CCObject[] { new CCDatabase() };
}
```

Можно создать несколько подклассов CCObject типа UDB_DATABASE, однако если значения, возвращаемые методом isEditable или isConfigurable этих подклассов будут совпадать, то объекты, возвращающие значение false, переопределят те объекты, которые возвращают значение true.

В данном примере осталось реализовать метод getMenuActions. Этот метод возвращает массив CCMenuAction, поэтому вначале нужно создать класс, реализующий этот интерфейс.

Необходимо реализовать два метода: getMenuText и actionPerformed. Метод getMenuText задает текст, отображаемый в меню. При выборе пункта меню генерируется событие и вызывается метод actionPerformed.

Приведенный ниже пример класса показывает пункт меню ″Example2a Action″, если выбран один объект базы данных. При выборе этого пункта меню на консоль выводится сообщение ″Example2a menu item actionPerformed″.

```
class Example2AAction implements CCMenuAction {

  public String getMenuText () { return "Example2a Action"; }

  public void actionPerformed (ActionEvent e) {
```

```
    System.out.println("Example2a menu item actionPerformed");
  }

}
```

Этот пункт меню необходимо подключить к объекту CCObject, представляющему объект Database, добавив в объект CCObject следующее:

```
public CCMenuAction[] getMenuActions () {
   return new CCMenuAction[] { new Example2AAction() };
}
```

Понятия, связанные с данным:

- “Компиляция и запуск примеров встраиваемых модулей” на стр. 422

Задачи, связанные с данной темой:

- “Изменение расположения пункта меню” на стр. 428
- “Создание разделителя в основном меню” на стр. 429
- “Создание подменю” на стр. 430
- “Добавление пункта меню для объекта с указанным именем” на стр. 431

Изменение расположения пункта меню

Процедура:

В части A расположение пункта меню указано не было. По умолчанию новые пункты меню добавляются в конец меню, либо перед пунктами Обновить и Фильтр, если они есть.

Вы можете явно задать позицию пункта меню, указав значение от 0 до максимального числа пунктов меню без учета пунктов Обновить и Фильтр. Укажите в определении подкласса CCMenuAction ключевое слово Positionable и реализуйте метод getPosition:

```
class Example2BAction implements CCMenuAction, Positionable {

  public String getMenuText () { return "Example2B Action"; }

  public void actionPerformed (ActionEvent e) {
    System.out.println("Example2B menu item actionPerformed");
  }

  public int getPosition() {
     return 0;
  }
}
```

Пункт меню с нулевым номером помещается в начало списка, а пункт меню, номер которого совпадает с числом пунктов меню без учета нового пункта

меню, помещается в конец списка, но перед пунктами Обновить и Фильтр. Кроме того, можно вернуть значение Positionable.POSITION_BOTTOM, чтобы была выбрана позиция пункта меню по умолчанию, то есть чтобы пункт меню был помещен в конец списка, но перед пунктами меню Обновить и Фильтр. Если существует несколько объектов CCObject типа UDB_DATABASE, в которых для пункта меню выбрано расположение POSITION_BOTTOM, эти пункты меню располагаются в том порядке, в котором объекты CCObject типа UDB_DATABASE возвращаются методом getObjects в классе CCExtension.

Добавьте Example2BAction в CCDatabase:

```
public CCMenuAction[] getMenuActions () {
  return new CCMenuAction[] { new Example2AAction(),
                              new Example2BAction() };
}
```

Задачи, связанные с данной темой:

- “Добавление действий в основное меню” на стр. 426
- “Создание разделителя в основном меню” на стр. 429
- “Создание подменю” на стр. 430
- “Добавление пункта меню для объекта с указанным именем” на стр. 431

Создание разделителя в основном меню

Процедура:

Для добавления разделителя создайте класс CCMenuAction, реализующий интерфейс Separator. Все остальные методы (за исключением getPosition, если создается класс Positionable) игнорируются.

```
class Example2CSeparator implements CCMenuAction, Separator, Positionable {

  public String getMenuText () { return null; }

  public void actionPerformed (ActionEvent e) {}

  public int getPosition() {
     return 1;
  }
}

public CCMenuAction[] getMenuActions () {
  return new CCMenuAction[] { new Example2AAction(),
                              new Example2BAction(),
                              new Example2CSeparator() };
}
```

Задачи, связанные с данной темой:

- “Добавление действий в основное меню” на стр. 426

- “Изменение расположения пункта меню” на стр. 428
- “Создание подменю” на стр. 430
- “Добавление пункта меню для объекта с указанным именем” на стр. 431

Создание подменю

Процедура:

Подменю представляет собой массив объектов CCMenuAction. Для того чтобы создать пункт меню, представляющий подменю, необходимо реализовать интерфейс SubMenuParent. Затем необходимо реализовать метод CCMenuAction для каждого пункта подменю и вернуть их в виде массива при помощи метода getSubMenuActions интерфейса SubMenuParent. Пункты меню нельзя добавить в подменю, не относящиеся к расширению. Кроме того, обратите внимание, что SubMenuParent не получает ActionEvent от Центра управления. Например:

```
class Example2DAction implements CCMenuAction, SubMenuParent {

  public String getMenuText () { return "Example2D Action"; }

  public void actionPerformed (ActionEvent e) {}

  public CCMenuAction[] getSubMenuActions() {
     return new CCMenuAction[] { new Example2DSubMenuAction() };
  }

}

class Example2DSubMenuAction implements CCMenuAction {

  public String getMenuText () { return "Example2D Sub-Menu Action"; }

  public void actionPerformed (ActionEvent e) {
    System.out.println("Example2D sub-menu menu item actionPerformed");
  }

}
```

Добавим новый пункт меню в CCDatabase.

```
  public CCMenuAction[] getMenuActions () }
    return new CCMenuAction[] { new Example2AAction(),
                                new Example2BAction(),
                                new Example2CSeparator(),
                                new Example2DAction() };
  }
```

Задачи, связанные с данной темой:

- “Добавление действий в основное меню” на стр. 426
- “Изменение расположения пункта меню” на стр. 428
- “Создание разделителя в основном меню” на стр. 429

• “Добавление пункта меню для объекта с указанным именем” на стр. 431

Добавление пункта меню для объекта с указанным именем

Процедура:

В настоящий момент созданные пункты меню будут показаны в меню всех баз данных в дереве объектов Центра управления. Для того чтобы эти пункты были добавлены в меню только одной базы данных, имя этой базы данных должно возвращаться методом getName объекта CCDatabase. Имя должно быть задано полностью. Другими словами, в значении, возвращаемом методом getName, должно быть задано имя системы, имя экземпляра и имя базы данных. Эти имена должны быть отделены друг от друга символом ″ – ″. В приведенном ниже примере рассматривается система MYSYSTEM с экземпляром DB2 и базой данных SAMPLE.

```
class CCDatabase implements CCObject {
  ...
  public String getName () { return "MYSYSTEM - DB2 - SAMPLE"; }
  ...
}
```

Задачи, связанные с данной темой:

• “Добавление действий в основное меню” на стр. 426
• “Изменение расположения пункта меню” на стр. 428
• “Создание разделителя в основном меню” на стр. 429
• “Создание подменю” на стр. 430

Создание модуля, который добавляет объекты модулей в дерево базы данных

Ниже описана процедура создания модуля, который добавляет объекты модулей в дерево базы данных:

1. Добавление папки для нескольких объектов в дерево
2. Добавление в папку примера объекта
3. Задание атрибутов объекта дерева модулей
4. Добавление действия создания
5. Добавление действия удаления
6. Добавление действия изменения

Добавление папки для хранения нескольких объектов в дерево

Процедура:

В данном примере вместо класса CCObject будет реализован класс CCTreeObject, для того чтобы добавить новый объект в категорию База данных дерева

объектов Центра управления. Вначале необходимо создать реализацию CCTreeObject для этого объекта. Если в список База данных необходимо добавить несколько объектов, для этих объектов можно создать папку. Ниже приведено начальное определение папки:

```
  public class Example3Folder implements CCTreeObject {
    public boolean isEditable () { return false; }
    public boolean isConfigurable () { return false; }
    public CCTableObject getChildren () { return null; }
    public CCColumn[] getColumns () { return null; }
    public boolean isLeaf () { return false; }
    public CCMenuAction[] getMenuActions () { return null; }

    public String getName () { return "Example3 Folder"; }

    public void getData (Object[] data) {
       data[0] = this;
    }

    public int getType () { return CCTypeFactory.getTypeNumber(this); }

    public Icon getIcon (int iconState) {
      switch (iconState) {
        case CLOSED_FOLDER:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_CLOSED_
FOLDER);

        case OPEN_FOLDER:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_OPEN_
FOLDER);

        default:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_CLOSED_
FOLDER);
      }
    }

  }
```

Обратите внимание, что метод getType применяет класс CCTypeFactory. CCTypeFactory гарантирует, что номера типов объектов не будут совпадать, поэтому Центр управления сможет идентифицировать расширения по типу. Новая папка относится не к одному из встроенных типов объектов CC, а к новому типу, поэтому ее номер типа не должен совпадать с номерами встроенных типов и номерами тех типов, которые будут созданы в будущем.

В методе getIcon предусмотрен параметр iconState, позволяющий узнать, открыта ли папка. Вы можете изменить значок в соответствии с текущим состоянием папки, как показано выше.

Для того чтобы папка выводилась не только в дереве объектов, но и в окне сведений при выборе базы данных, метод getData должен возвращать столбец,

значением которого является сам объект расширения. Метод getData устанавливает указатель *this* на первый элемент массива данных. Это позволяет вывести в одном столбце окна сведений и значок, и имя. Если возвращается подкласс CCTableObject, то Центр управления считает, что можно вызвать метод getIcon и getName для папки Example3Folder.

Далее необходимо создать класс CCDatabase для реализации CCTreeObject и вернуть в методе getChildren массив CCTableObject, содержащий экземпляр Example3Folder:

```
import java.util.*;

  class CCDatabase implements CCTreeObject {
    private Vector childVector;

    public CCDatabase() {
      childVector = new Vector();
      childVector.addElement(new Example3Folder());
    }

    public CCTableObject[] getChildren() {
      CCTableObject[] children = new CCTableObject[childVector.size()];
      childVector.copyInto(children);
      return children;
    }

    public String getName () { return null; }
    public boolean isEditable () { return false; }
    public boolean isConfigurable () { return false; }
    public void getData (Object[] data) { }
    public CCColumn[] getColumns () { return null; }
    public boolean isLeaf () { return false; }
    public int getType () { return UDB_DATABASE; }
    public Icon getIcon (int iconState) { return null; }
    public CCMenuAction[] getMenuActions () { return null; }
  }
```

Понятия, связанные с данным:

- “Компиляция и запуск примеров встраиваемых модулей” на стр. 422

Задачи, связанные с данной темой:

- “Добавление примера объекта в папку” на стр. 434
- “Настройка атрибутов нового объекта дерева” на стр. 435
- “Добавление команды создания” на стр. 437
- “Добавление команды удаления с возможностью выбора нескольких объектов” на стр. 439
- “Добавление команды изменения” на стр. 440

Добавление примера объекта в папку

Процедура:

Вначале необходимо создать реализацию CCObject для дочернего объекта:

```
class Example3Child implements CCTableObject {
  public String getName () { return null; }
  public boolean isEditable () { return false; }
  public boolean isConfigurable () { return false; }
  public CCTableObject[] getChildren () { return null; }
  public void getData (Object[] data) { }
  public CCColumn[] getColumns () { return null; }
  public boolean isLeaf () { return false; }
  public Icon getIcon (int iconState) { return null; }
  public CCMenuAction[] getMenuActions () { return null; }
  public int getType () { return CCTypeFactory.getTypeNumber(this); }
}
```

Затем нужно изменить класс Example3Folder, сохранив вектор объектов Exercise3Child:

```
public class Example3Folder implements CCObject {
  private Vector childVector;

  public Example3Folder() {
    childVector = new Vector();
  }

  ...
  public CCTableObject[] getChildren () {
    CCTableObject[] children = new CCTableObject[childVector.size()];
    childVector.copyInto(children);
    return children;
  }
}
```

Понятия, связанные с данным:

- “Компиляция и запуск примеров встраиваемых модулей” на стр. 422

Задачи, связанные с данной темой:

- “Добавление папки для хранения нескольких объектов в дерево” на стр. 431
- “Настройка атрибутов нового объекта дерева” на стр. 435
- “Добавление команды создания” на стр. 437
- “Добавление команды удаления с возможностью выбора нескольких объектов” на стр. 439
- “Добавление команды изменения” на стр. 440

Настройка атрибутов нового объекта дерева

Процедура:

Если в дерево объектов была добавлена папка, то при выборе этой папки никакой информации на панели сведений показано не будет. Это связано с тем, что реализация метода getColumns в объекте Example3Child возвращает значение null. Для того чтобы задать непустое значение, необходимо создать несколько реализаций класса CCColumn. В данном примере будет создано два столбца. Один из них - это обязательный неизменяемый столбец объекта, а второй - столбец, который будет применяться в последующих примерах для демонстрации того, каким образом можно изменить значение столбца. Неизменяемый столбец будет называться “Name”, а изменяемый столбец - “State”.

```
class NameColumn implements CCColumn {
   getName() { return "Name"; }
   getColumnClass { return CCTableObject.class; }
}

class StateColumn implements CCColumn {
   getName() { return "State"; }
   getColumnClass { return String.class; }
}
```

В число поддерживаемых типов классов входят классы, эквивалентные простейшим типам Java (например, java.lang.Intger), класс java.util.Date и класс CCTableObject.

Измените метод getColumns класса Example3Child таким образом, чтобы он возвращал указанные два столбца.

```
class Example3Child implements CCTableObject {
  ...
  public CCColumn[] getColumns () {
     return new CCColumn[] { new NameColumn(),
                             new StateColumn() };
  }
  ...
}
```

Кроме того, необходимо добавить эти столбцы в родительский класс.

```
class Example3Folder implements CCTableObject {
  ...
  public CCColumn[] getColumns () {
     return new CCColumn[] { new NameColumn(),
                             new StateColumn() };
  }
  ...
}
```

Теперь необходимо задать значения, которые будут указаны в строках в окне сведений. Для этого необходимо задать элементы массива Object, передаваемого методу getData. Класс каждого столбца должен совпадать с классом соответствующего столбца, возвращаемого методом getColumnClass.

```
class Example3Child implements CCTableObject {
  ...
  private String name;
  private String state;

  public Exampe3Child(String name, String state) {
    this.name = name;
    this.state = state;
  }
  ...
  public void getData (Object[] data) {
    data[0] = this;
    data[1] = state;
  }
  ...
}
```

В данном случае первый столбец, относящийся к классу CCTableObject, будет содержать значение this. Это позволит Центру управления вывести как текст, возвращаемый функцией getName, так и значок, возвращаемый методом getIcon. Это реализуется на следующем шаге. Будет применяться тот же значок обновления, который был задан в первом примере для кнопки панели инструментов.

```
class Example3Child implements CCTableObject {
  ...
  public String getName () {
    return name;
  }
  public Icon getIcon () {
 return CommonImageRepository.getScaledIcon(CommonImageRepository.WC_NV_
REFRESH);
  }
  ...
}
```

Для того чтобы просмотреть результаты выполненных действий, можно создать пример дочернего объекта, который будет удален в следующем примере. Добавьте экземпляр объекта Example3Child в Example3Folder при создании childVector.

```
public class Example3Folder implements CCTreeObject {
  ...
  public Example3Folder() {
    childVector = new Vector();
    childVector.addElement(new Example3Child("Plugin1", "State1"));
  }
  ...
}
```

Понятия, связанные с данным:

- “Компиляция и запуск примеров встраиваемых модулей” на стр. 422

Задачи, связанные с данной темой:

- “Добавление папки для хранения нескольких объектов в дерево” на стр. 431
- “Добавление примера объекта в папку” на стр. 434
- “Добавление команды создания” на стр. 437
- “Добавление команды изменения” на стр. 440

Добавление команды создания

Процедура:

Для того чтобы пользователи могли динамически создавать объекты в вашей папке, необходимо обновить вектор, пометить класс как Observable и указать, что при генерации события пользователем должен вызываться метод notifyObservers. Центр управления автоматически регистрируется как наблюдать для всех объектов CCTableObject, помеченных как Observable.

Вначале задайте в классе Example3Folder метод, добавляющий дочерний объект в вектор.

```
public class Example3Folder implements CCTreeObject, Observable {
  ...
  public void addChild(Example3Child child) {
    childVector.addElement(child);
    setChanged();
    notifyObservers(new CCObjectCollectionEvent(this,
                    CCObjectCollectionEvent.OBJECT_ADDED,
                    child));
  }
  ...
}
```

В этом фрагменте кода в качестве аргумента метода notifyObservers задан новый класс CCObjectCollectionEvent. Класс CCObjectCollectionEvent представляет событие, генерируемое при изменении набора объектов CCObject, например, папки в дереве объектов Центра управления. Центр управления наблюдает за всеми объектами CCObjects, расширяющими класс Observable, и обновляет дерево объектов и панель сведений при возникновении события CCObjectCollectionEvents. Существует три типа событий: добавление, удаление и изменение.

В конструкторе класса CCObjectCollectionEvent предусмотрено три аргумента. Первый аргумент представляет объект, который генерирует событие. Второй

аргумент представляет тип события. Допустимы значения OBJECT_ADDED, OBJECT_ALTERED и OBJECT_REMOVED. Последний аргумент представляет создаваемый объект.

Теперь необходимо добавить пункт в меню папки, при выборе которого будет вызываться новый метод addChild.

```
class CreateAction implements CCMenuAction {
  private int pluginNumber = 0;
  public String getMenuText () { return "Create"; }

  public void actionPerformed (ActionEvent e) {
Example3Folder folder = (Example3Folder)((Vector)e.getSource()).elementAt(0);
    folder.addChild(new Example3Child("Plugin " + ++pluginNumber, "State1"));
  }
}
```

ActionEvent всегда содержит вектор всех объектов, для которых выполняется действие. Поскольку действие будет выполняться только для папки Example3Folder, и эта папка единственная в списке, то метод addChild вызывается для первого объекта вектора.

Теперь можно добавить действие в меню папки и удалить пример объекта, созданный ранее.

```
public class Example3Folder extends Observable implements CCTreeObject {
  private CCMenuAction[] menuActions =
     new CCMenuAction[] { new CreateChildAction(); }
  ...
  public Example3Folder() {
    childVector = new Vector();
  }
  ...
  public CCMenuAction[] getMenuActions () {
    retun menuActions;
  }
  ...
}
```

Понятия, связанные с данным:

- “Компиляция и запуск примеров встраиваемых модулей” на стр. 422

Задачи, связанные с данной темой:

- “Добавление папки для хранения нескольких объектов в дерево” на стр. 431
- “Добавление примера объекта в папку” на стр. 434
- “Настройка атрибутов нового объекта дерева” на стр. 435
- “Добавление команды удаления с возможностью выбора нескольких объектов” на стр. 439
- “Добавление команды изменения” на стр. 440

Добавление команды удаления с возможностью выбора нескольких объектов

Процедура:

Теперь пользователи могут создать любое число экземпляров расширения, и им необходимо предоставить возможность удалять такие экземпляры. Вначале необходимо добавить метод в класс Example3Folder для удаления дочернего объекта и уведомления Центра управления.

```
public class Example3Folder extends Observable implements CCTreeObject {

  public void removeChild(Example3Child child) {
    childVector.removeElement(child);
    setChanged();
    notifyObservers(new CCObjectCollectionEvent(this,
                    CCObjectCollectionEvent.OBJECT_REMOVED,
                    child));
  }
}
```

Далее необходимо добавить действие в меню объекта Example3Child. Соответствующий класс CCMenuAction будет реализовывать класс MultiSelectable, для того чтобы пользователи могли удалить сразу несколько объектов. Поскольку действие будет выполняться над объектами Example3Child из вектора Vector, а не над объектом Example3Folder, объект Example3Folder необходимо передать каким-то другим способом, например, с помощью конструктора.

```
class RemoveAction implements CCMenuAction, MultiSelectable {
  private Example3Folder folder;

  public RemoveAction(Example3Folder folder) {
     this.folder = folder;
  }

  public String getMenuText () { return "Remove"; }

  public int getSelectionMode () { return MultiSelectable.MULTI_HANDLE_ONE; }

  public void actionPerformed (ActionEvent e) {
    Vector childrenVector = (Vector)e.getSource();
    for (int i = 0; i < childrenVector.size(); i++) {
      folder.removeChild((Example3Child)childrenVector.elementAt(i));
    }
  }
}
```

Для реализации класса MultiSelectable необходимо реализовать метод getSelectionMode. В данном случае он возвращает значение MULTI_HANDLE_ONE, указывающее, что этот пункт меню будет показан даже

в том случае, если выбрано несколько объектов, причем для всех выбранных объектов метод actionPerformed будет вызван только один раз.

Теперь необходимо добавить новую команду в меню объекта Example3Child. Для этого потребуется добавить новый параметр в конструктор Example3Child, в котором будет передаваться объект папки.

```
class Example3Child implements CCTableObject {
  ...
  private CCMenuAction[] menuActions;

  public Example3Child(Example3Folder folder, String name, String state) {
    ...
    menuActions = new CCMenuAction[] { new RemoveAction(folder) };
  }
  ...
  public CCMenuAction[] getMenuActions () {
    return menuActions;
  }
}
```

Не забудьте изменить конструктор в классе CreateAction.

```
class CreateAction implements CCMenuAction {
  ...
  public void actionPerformed (ActionEvent e) {
    ...
folder.addChild(new Example3Child(folder, "Plugin " + ++pluginNumber,
"State 1"));
  }
}
```

Понятия, связанные с данным:

- “Компиляция и запуск примеров встраиваемых модулей” на стр. 422

Задачи, связанные с данной темой:

- “Добавление папки для хранения нескольких объектов в дерево” на стр. 431
- “Добавление примера объекта в папку” на стр. 434
- “Настройка атрибутов нового объекта дерева” на стр. 435
- “Добавление команды создания” на стр. 437
- “Добавление команды изменения” на стр. 440

Добавление команды изменения

Процедура:

Последний тип события, отслеживаемого Центром управления для расширений - это событие OBJECT_ALTERED. В предыдущем примере был создан столбец

“State”, на котором теперь будет продемонстрирована операция изменения. При выполнении действия Alter будет увеличиваться значение состояния.

Вначале необходимо создать метод, изменяющий состояние. Этот метод нужно добавить в класс Example3Child, а не в класс папки. Первый и третий аргументы этого метода представляют собой Example3Child. Не забудьте указать, что класс расширяет класс Observable.

```
class Example3Child extends Observable implements CCTableObject {
  ...
  public void setState(String state) {
    this.state = state;
    setChanged();
    notifyObservers(new CCObjectCollectionEvent(this,
                        CCObjectCollectionEvent.OBJECT_ALTERED, this));
  }
  ...
}
```

Затем создайте пункт меню Alter и добавьте его в массив CCMenuAction в классе Example3Child.

```
class AlterAction implements CCMenuAction {
  private int stateNumber = 1;
  public String getMenuText () { return "Alter"; }

  public void actionPerformed (ActionEvent e) {
    ((Example3Child)((Vector)e.getSource()).elementAt(0)).setState("State "
        + ++stateNumber);
  }
}

class Example3Child implements CCTableObject {
  ...
  public Example3Child(Example3Folder folder, String name, String state) {
    ...
    menuActions = new CCMenuAction[] { new AlterAction(),
                                       new RemoveAction(folder) };
  }
  ...
}
```

Понятия, связанные с данным:

- “Компиляция и запуск примеров встраиваемых модулей” на стр. 422

Задачи, связанные с данной темой:

- “Добавление папки для хранения нескольких объектов в дерево” на стр. 431
- “Добавление примера объекта в папку” на стр. 434
- “Настройка атрибутов нового объекта дерева” на стр. 435
- “Добавление команды создания” на стр. 437

- “Добавление команды удаления с возможностью выбора нескольких объектов” на стр. 439

Отключение функции настройки с помощью метода isConfigurable()

Процедура:

Если метод isConfigurable объекта CCObject типа UDB_DATABASE или UDB_INSTANCE возвращает значение false, то в выпадающем меню базы данных или экземпляра не будет пункта меню Настроить.

Пример программы, удаляющей пункт Настроить из меню всех баз данных или экземпляров, приведен в файле Example4.java. Кроме того, пункт меню можно удалить только для тех баз данных или экземпляров, имя которых совпадает со значением, возвращаемым методом getName().

Понятия, связанные с данным:

- “Компиляция и запуск примеров встраиваемых модулей” на стр. 422

Отключение возможности изменять объекты с помощью метода isEditable()

Процедура:

Для того чтобы запретить изменение любого объекта Центра управления, поддерживающего действие Изменить, можно создать расширение объекта, в котором метод isEditable возвращает значение false. Пример программы, удаляющей действие Изменить из меню таблиц, производных таблиц и индексов UDB, приведен в файле Example5.java. Аналогичным образом можно запретить изменение любого объекта, в меню которого есть действие Изменить. Кроме того, действие Изменить можно удалить из меню только того объекта, полное имя которого совпадает со значением, возвращаемым методом getName.

При необходимости в меню можно добавить действие Показать. Такое действие поддерживается только таблицами, производными таблицами и индексами UDB. Для добавления действия Показать необходимо добавить файл BrowseTable, BrowseView или BrowseIndex в архив db2plug.zip. Файл может не содержать никаких данных, необходимо только то, чтобы он назывался BrowseTable, BrowseView или BrowseIndex. Кнопку Показать нельзя добавить только для объекта с указанным именем.

Понятия, связанные с данным:

- “Компиляция и запуск примеров встраиваемых модулей” на стр. 422

Отключение кнопок значений по умолчанию в окнах настройки с помощью метода hasConfigurationDefaults()

Процедура:

Окна настройки баз данных и экземпляров UDB содержат кнопки, позволяющие восстановить для параметров значения по умолчанию, применяемые в DB2. Если у вас определен собственный набор значений по умолчанию, и вы не хотите, чтобы пользователь мог случайно установить значения по умолчанию DB2, нажав соответствующие кнопки, отключите эти кнопки с помощью расширения. Создайте объект типа UDB_DATABASE или UDB_INSTANCE, реализующий класс CCObject. Реализуйте в нем интерфейс Defaultable. Этот интерфейс содержит метод hasConfigurationDefaults. Если этот метод будет возвращать значение false, то в окнах настройки будут отключены все кнопки значений по умолчанию. Ознакомьтесь с примером, отключающим кнопки значений по умолчанию в окнах настройки базы данных UDB.

Понятия, связанные с данным:

- “Компиляция и запуск примеров встраиваемых модулей” на стр. 422

Приложение J. 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 49. 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 50. 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 50. 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 51. 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 51. 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 52. 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 53. 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 54. 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 54. 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 55. 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 56. 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 57. 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. 454
- “Заказ печатных копий книг DB2” на стр. 455
- “Обращение к электронной справке” на стр. 456
- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 460
- “Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML” на стр. 461

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. 455
- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 460
- “Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML” на стр. 461

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 445

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. 454
- “Поиск тем при обращении к Информационному центру DB2 из браузера” на стр. 457
- “Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML” на стр. 461

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 445

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. 457
- “Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML” на стр. 461

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. 467
- “Информационный центр DB2 при обращении из браузера” на стр. 470

Задачи, связанные с данной темой:

- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 460
- “Обновление документации HTML, установленной на вашем компьютере” на стр. 462
- “Устранение ошибок при поиске в документации DB2 с помощью Netscape 4.x” на стр. 464
- “Поиск в документации DB2” на стр. 465

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 445

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. 467
- “Информационный центр DB2 при обращении из браузера” на стр. 470

Задачи, связанные с данной темой:

- “Поиск тем при обращении к Информационному центру DB2 из браузера” на стр. 457
- “Поиск в документации DB2” на стр. 465

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. 457
- “Копирование файлов с компакт-диска Документация по DB2 в формате HTML на Web-сервер” на стр. 463

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 445

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. 463

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 445

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. 465

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. 445

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. 465

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. 464

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. 470

Задачи, связанные с данной темой:

- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 460

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. 468.
- DB2 поддерживает возможности доступности в программах, которые используют API доступности Java™. Смотрите раздел “Совместимость с технологиями для людей с физическими недостатками” на стр. 468.
- DB2 поставляется с документацией в формате, обеспечивающем доступность. Смотрите раздел “Удобный формат документации” на стр. 468.

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.

На этом языке нужная вам тема недоступна

На этом языке нужная вам тема недоступна.

Поэтому в отдельном окне браузера выведена английская версия этой темы.

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. 457
- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 460
- “Обновление документации HTML, установленной на вашем компьютере” на стр. 462

Приложение K. Замечания

IBM может предлагать описанные продукты, услуги и возможности не во всех странах. Сведения о продуктах и услугах, доступных в настоящее время в вашей стране, можно получить в местном представительстве IBM. Любые ссылки на продукты, программы или услуги IBM не означают явным или неявным образом, что можно использовать только продукты, программы или услуги IBM. Разрешается использовать любые функционально эквивалентные продукты, программы или услуги, если при этом не нарушаются права IBM на интеллектуальную собственность. Однако ответственность за оценку и проверку работы любых продуктов, программ и услуг других фирм лежит на пользователе.

Фирма IBM может располагать патентами или рассматриваемыми заявками на патенты, относящимися к предмету данного документа. Получение этого документа не означает предоставления каких-либо лицензий на эти патенты. Запросы по поводу лицензий следует направлять в письменной форме по адресу:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

По поводу лицензий, связанных с использованием наборов двухбайтных символов (DBCS), обращайтесь в отдел интеллектуальной собственности IBM в вашей стране/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

E

F

G

I

K

L

M

N

P

Q

R

S

T

U

W

A

Б

В

Г

Д

Ж

З

И

К

Л

М

Н

О

П

P

C

Т

У

Ф

Ц

Ш

Э

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.

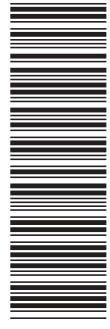
Если вы находитесь в США, позвоните по одному из следующих номеров:

- 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

Код изделия: CT182RU

Напечатано в Дании



(1P) P/N: CT182RU

GH43-0202-00

Spine information:

IBM® DB2 Universal Database™

Руководство администратора: Реализация *Версия 8*