IBM® DB2® Universal Database

Руководство администратора: Реализация

Версия 7

SH43-0144-01

IBM® DB2® Universal Database

Руководство администратора: Реализация

Версия 7

SH43-0144-01

Перед тем, как использовать данный документ и продукт, описанный в нем, прочтите общие сведения под заголовком “Приложение M. Замечания” на стр. 489.

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

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

Содержание

Об этой книге

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

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

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

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

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

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

Для кого предназначена эта книга

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

Структура этой книги

Эта книга содержит сведения на следующие основные темы:

Управление при помощи Центра управления

- Раздел Глава 1. Управление DB2 с помощью инструментов GUI знакомит с инструментами графического пользовательского интерфейса (GUI) для управления базой данных.

Реализация вашего проекта

- В разделе Глава 2. Перед созданием базы данных описываются предварительные требования для создания базы данных и объектов в базе данных.
- В разделе Глава 3. Создание базы данных описываются задачи, связанные с созданием базы данных и объектов в базе данных.
- В разделе Глава 4. 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 5. Управление доступом к базам данных описывается, как управлять доступом к ресурсам вашей базы данных.

- В разделе Глава 6. Аудит действий DB2 описывается, как обнаруживать и отслеживать нежелательные или непредвиденные попытки обращения к данным.

Перемещение данных

- В разделе Глава 7. Утилиты для перемещения данных описываются утилиты Load, AutoLoader, Import и Export.

Восстановление

- В разделе Глава 8. Восстановление базы данных описываются факторы, которые следует принимать во внимание при выборе методов восстановления базы данных и табличных пространств. В задачи восстановления входят резервное копирование и восстановление базы данных или табличного пространства, а также использование восстановления с повтором транзакций.

Приложения

- В разделе Приложение A. Правила именования приводятся правила именования баз данных и объектов.
- В разделе Приложение B. Использование служб каталогов среды DCE приводится информация о том, как пользоваться службами каталога DCE.
- В разделе Приложение C. Обработчик пользователя для восстановления баз данных обсуждается, каким можно использовать обработчики пользователя с файлами журнала базы данных, и описываются некоторые примеры обработчиков пользователя.
- В разделе Приложение D. 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.
- В разделе Приложение E. Как DB2 for Windows NT работает с защитой Windows NT описывается, как DB2 работает с защитой Windows NT.
- В разделе Приложение F. Использование монитора производительности Windows NT описываются два монитора производительности, доступные пользователям DB2 for Windows NT: монитор производительности DB2 и монитор производительности Windows NT.
- В разделе Приложение G. Работа с серверами разделов баз данных Windows NT или Windows 2000 описываются утилиты, используемые Windows NT и Windows 2000 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. Высокоскоростная межузловая связь описывается, как разрешить использование Virtual Interface Architecture с DB2 Enterprise - Extended Edition в среде Windows NT.

- В разделе Приложение J. Службы каталогов протокола LDAP приводится информация о том, как пользоваться службами каталога LDAP.
- В разделе Приложение K. Расширение Центра управления приводится информация о том, как расширить Центр управления, добавив новые кнопки на панель инструментов, новые действия, новые определения объектов и действий.
- В разделе Приложение L. Использование библиотеки DB2 приводится информация о структуре библиотеки DB2, включая мастера, электронную справку, сообщения и книги.

Краткий обзор других томов Руководства администратора

Руководство администратора: Планирование

Книга *Administration Guide: Planning* посвящена разработке базы данных. В ней освещаются вопросы, связанные с логическими и физическими устройствами, распределенными транзакциями и высокой доступностью. Отдельные главы и приложения из этого тома кратко описаны здесь:

В мире DB2 Universal Database

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

Концепции баз данных

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

Проектирование баз данных

- В разделе ″Логическая структура базы данных″ обсуждаются концепции логической структуры базы данных и даются указания по разработке баз данных.

- В разделе ″Физическая структура базы данных″ содержатся указания по разработке физической структуры базы данных, в том числе по вопросам хранения данных.

Распределенная обработка транзакций

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

Системы высокой доступности

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

Приложения

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

Руководство администратора: Производительность

Книга *Administration Guide: Performance* 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 ″Элементы производительности″ знакомит с идеями и особенностями управления производительностью DB2 UDB и ее повышения.
- В разделе ″Обзор архитектуры и процессов обработки″ излагаются основы архитектуры и процессов DB2 Universal Database.

Настройка производительности прикладных программ

- В разделе ″Особенности прикладных программ″ описываются некоторые методы повышения производительности базы данных при разработке ваших прикладных программ.

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

Настройка и конфигурирование вашей системы

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

Приложения

- В приложении ″Реестр и переменные среды DB2″ 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. Управление с использованием Центра управления

Глава 1. Управление DB2 с помощью инструментов GUI

DB2 Universal Database содержит инструменты графического интерфейса пользователя (GUI), позволяющие легко управлять локальными и удаленными базами данных из одной центральной программы, которая называется Центр управления.

Эта глава содержит обзор доступных инструментов управления DB2 Universal Database и объясняет, как их использовать для облегчения вашей работы и повышения ее эффективности. Приводится также сводка сведений по Центру управления Java и по настройке Центра управления для включения собственных инструментов, поддерживающих Java.

В этой главе рассматриваются:

Инструменты управления

Инструменты управления DB2 входят в клиент управления, отдельно устанавливаемый по выбору компонент большинства продуктов DB2 Universal Database. Клиент управления имеется также на наборе компакт-дисков, где записаны клиенты управления для всех операционных систем, в которых работает DB2. Они позволяют установить и использовать клиент управления на любой рабочей станции, независимо от того, используете вы локальные или удаленные серверы баз данных, и в какой операционной системе они работают. Эти инструменты позволяют выполнять из графического интерфейса пользователя те же действия, что и из процессора командной строки. К ним относятся ввод команд DB2, операторов SQL и команд операционной системы. При работе с инструментами не требуется запоминать сложные операторы или команды, и вы получаете дополнительную помощь.

Примечание: Установку клиента управления можно выбрать при установке системы.

На панели инструментов Центра управления есть следующие инструменты:

- Центр управления. Это основной графический инструмент DB2 для управления базами данных. При помощи Центра управления можно наглядно видеть все системы и объекты баз данных, имеющиеся в локальном каталоге.
- Центр управления сателлитами. Центр управления сателлитами позволяет управлять серверами сателлитов DB2.
- Командный центр. Командный центр позволяет выдавать команды баз данных DB2, операторы SQL и команды операционной системы, а также повторять предыдущие команды и просматривать планы доступа для запросов SQL.
- Центр сценариев. Центр сценариев позволяет создавать, выполнять и включать в расписание команды операционной системы, командные сценарии DB2 и сценарии операторов SQL.
- Центр оповещений. Центр оповещений выдает сообщения о превышении установленных порогов и о прекращении работы узла в многоузловой среде.
- Журнал. Журнал позволяет просматривать состояние заданий, изменять расписания заданий, а также просматривать журнал истории восстановлений и журнал сообщений.
- Информационный центр. Информационный центр дает быстрый доступ к информации руководств по продуктам DB2 и к примерам программ, а также предоставляет доступ к другим источникам информации о DB2 в Web.
- Центр лицензий. Центр лицензий выводит состояние вашей лицензии и позволяет конфигурировать систему для правильного мониторинга лицензий.

Для некоторых функций, выполняемых при помощи инструментов GUI, можно использовать мастера. Мастера вызываются из всплывающих меню в Центре

управления. Они дают более подробные справки, шаг за шагом предлагают вам ввести необходимую для выполнения задачи информацию и даже делают вычисления и рекомендации на основе введенной вами информации. Мастера чрезвычайно полезны для начинающих администраторов баз данных и людей, управляющих базой данных только изредка.

В DB2 Universal Database есть следующие мастера:

- Мастер по резервному копированию баз данных. Он предлагает ввести простую информацию о данных в базе, доступности базы данных и требованиях к возможностям восстановления. Затем он предлагает план резервного копирования, создает сценарий задания и заносит его в расписание. Чтобы вызвать мастер резервного копирования баз данных, выберите значок базы данных для резервного копирования, щелкните правой кнопкой мыши и выберите **Резервное копирование —> Базы данных при помощи мастера**.
- Мастер по созданию баз данных. Этот мастер позволяет создать базу данных, назначить параметры хранения и выбрать основные опции производительности. Чтобы вызвать мастер создания баз данных, выберите значок Базы данных, щелкните правой кнопкой мыши и выберите **Создать —> Базу данных при помощи мастера**.
- Мастер по созданию таблиц. Этот мастер помогает создать столбцы с использованием готовых шаблонов столбцов, создать первичный ключ для таблицы и назначить для таблицы одно или несколько табличных пространств. Чтобы вызвать мастер, выберите значок Таблицы, щелкните правой кнопкой мыши и выберите **Создать —> Таблицу при помощи мастера**.
- Мастер по созданию табличных пространств. Этот мастер позволяет создать новое табличное пространство и задать основные опции хранения и производительности. Чтобы вызвать его, выберите значок Табличное пространство, щелкните правой кнопкой мыши и выберите **Создать —> Табличное пространство при помощи мастера**.
- Мастер по созданию индексов. Используйте этот мастер, чтобы решить, какие индексы создать или отбросить для данного набора операторов SQL. Рекомендуемые значения даются с учетом заданной вами рабочей нагрузки. Чтобы вызвать мастер по созданию индексов, выберите папку Индексы, щелкните правой кнопкой мыши и выберите **Создать —> Индекс при помощи мастера**.
- Мастер по настройке производительности. Этот мастер помогает настроить базы данных на основе информации о базе данных, ее данных и назначении системы.Затем он предлагает новые параметры конфигурации для базы данных и экземпляра и, если вы согласны, автоматически применяет их. Чтобы вызвать этот мастер, выберите значок базы данных, щелкните правой кнопкой мыши и выберите **Сконфигурировать при помощи мастера**.
- Мастер по восстановлению баз данных. Этот мастер проводит вас по процессу восстановления базы данных. Чтобы вызвать мастер, выберите

значок базы данных, щелкните правой кнопкой мыши и выберите **Восстановить —> Базу данных при помощи мастера**.

- Мастер по конфигурирование многоузлового изменения. Этот мастер позволяет так сконфигурировать базы данных, чтобы программы могли изменять данные сразу на нескольких узлах. Это важно, когда данные на всех узлах должны быть совместимы. Чтобы вызвать этот мастер, выберите экземпляр, щелкните правой кнопкой мыши и выберите **Многоузловое изменение —> Сконфигурировать при помощи мастера**.

Примечание: Для подсистемы DB2 for OS/390 мастеров нет.

Помимо графических инструментов, которые можно вызвать из панели инструментов Центра управления, есть несколько дополнительных инструментов GUI, не вызываемых прямо из панели инструментов Центра управления.

- Монитор производительности. Монитор производительности - это инструмент для наблюдения за такими объектами DB2, как экземпляры, базы данных, таблицы, табличные пространства и соединения. Этот инструмент используется для поиска проблем производительности и настройки баз данных для наилучшей производительности. Монитор производительности вызывается из всплывающих меню в Центре управления.
- Монитор событий. Монитор событий - это инструмент, позволяющий анализировать использование ресурсов с помощью записи состояния базы данных в моменты, когда происходят определенные события. Чтобы создать монитор событий, введите в командной строке DB2 `db2emcrt`.
- Анализатор событий. Анализатор событий - это инструмент, позволяющий анализировать данные, собранные Монитором событий. Чтобы вызвать анализатор событий, введите в командной строке DB2 `db2evmon`.
- Наглядное объяснение. Функция наглядного объяснения позволяет просматривать план доступа для операторов SQL в виде диаграммы, чтобы можно было настроить запросы SQL для наилучшей производительности. До Версии 6 наглядное объяснение работало как инструмент просмотра планов доступа. Теперь эта функция уже не является отдельным инструментом; она доступна во всплывающих меню различных объектов баз данных в Центре управления, а также в Командном центре.

Кроме этих инструментов, есть еще один полезный инструмент для управления базами данных, не входящий в Центр управления - Ассистент конфигурирования клиента. Ассистент конфигурирования клиента - это инструмент, содержащий мастера для помощи при конфигурировании связи клиентов с удаленными серверами.

Все эти инструменты более подробно описываются в дальнейшем. В следующем разделе приводится обзор возможностей этих инструментов.

Общие возможности инструментов

Следующие возможности доступны сразу в нескольких инструментах:

- Показать SQL и Показать команду
- Показать связанные
- Генерировать DDL
- Фильтр
- Справка

Показать SQL и Показать команду

Если инструмент генерирует операторы SQL, в его интерфейсе будет доступна кнопка **Показать SQL**. Аналогично, в инструменте, генерирующем команды DB2, будет доступна кнопка **Показать команду**. Эти кнопки позволяют:

- Увидеть операторы SQL или команды DB2, генерируемые инструментом на основе вашего выбора в графическом интерфейсе. Эта информация поможет вам понять, как работает интерфейс.
- Сохранить операторы или команды в виде сценария для последующего повторного использования. Эта возможность позволяет избежать повторного ввода операторов SQL или команд DB2, если вы захотите еще раз выполнить те же операторы или команды. Когда операторы SQL или команды DB2 записаны в сценарий, его можно включить в расписание, или внести в него изменения, или создавать подобные сценарии без повторного ввода операторов или команд.

Чтобы вывести операторы SQL или команды DB2:

1. Из Центра управления перейдите в окно или записную книжку для работы с инструментом, генерирующим операторы SQL или команды DB2. Будет доступна кнопка **Показать SQL** или **Показать команду**.
2. Нажмите кнопку **Показать SQL** или **Показать команду**. Откроется соответствующее окно.

Сохранение операторов SQL и команд DB2 в особенности полезно для сложных операторов SQL или команд DB2.

Используя кнопку **Показать команду** или **Показать SQL**, можно либо создать новые сценарии, которые потом можно отредактировать, либо закрыть диалоговое окно, вернуться в исходное окно и внести в нем изменения. Если вы нажмете кнопку Создать сценарий, появится окно Новый командный сценарий. В нем можно отредактировать операторы SQL или команды DB2 перед сохранением сценария.

Показать связанные

Показать связанные показывает прямые связи между таблицами, индексами, производными таблицами, алиасами, триггерами, табличными пространствами, пользовательскими функциями и пользовательскими типами. Например, если

вы выбрали таблицу и задали вывод связанных производных таблиц, вы увидите только производные таблицы, основанные непосредственно на этой таблице. Производные таблицы, основанные на связанных производных таблицах, не будут выведены, так как эти производные таблицы не были созданы прямо из этой таблицы.

Вывод связанных объектов помогает:

- Понять структуру базы данных.
- Выяснить, какие индексы для таблицы уже существуют.
- Выяснить, какие объекты хранятся в табличном пространстве.
- Выяснить, какие объекты связаны с данным объектом и будут затронуты вашими возможными действиями. Например, если вы хотите отбросить таблицу с зависящими от нее производными таблицами, **Показать связанные** покажет, какие производные таблицы станут недействительными.

Чтобы использовать возможность Показать связанные:

- В Центре управления выберите объект на панели содержимого и щелкните правой кнопкой мыши.
- Выберите **Показать связанные**
- Щелкните по закладке, чтобы открыть страницу с нужными связанными объектами. В зависимости от выбранной закладки будет выведен список различных связанных объектов. Выводятся только объекты, непосредственно связанные с выбранным вами объектом.

  Можно щелкнуть по связанному объекту правой кнопкой мыши на выбранной странице и выбрать во всплывающем меню Показать связанные. На выбранной странице будут выведены объекты, связанные с последним из выбранных. Можно также щелкнуть по стрелке вниз рядом с выбранным объектом, чтобы вывести список объектов, выбиравшихся для вывода связей ранее.
- Чтобы закрыть записную книжку Показать связанные и вернуться в Центр управления, нажмите кнопку **Закрыть**.

Генерировать DDL

Функция **Генерировать DDL** позволяет повторно создать и сохранить в файл сценария DDL, операторы SQL и статистику по следующим объектам:

- Объекты базы данных
- Операторы авторизации
- Табличные пространства, группы узлов и пулы буферов
- Статистика баз данных

Это позволяет:

- Сохранять DDL, чтобы создавать идентично определенные таблицы, базы данных и индексы в другой базе данных, например, для программы хранилища баз данных
- Использовать DDL для копирования базы данных из среды тестирования в среду производства или из одной системы в другую
- Редактировать DDL для создания похожих объектов

Если нажать кнопку **Генерировать DDL**, откроется окно Показать команду с операторами, сгенерированными утилитой **db2look**. В окне Показать команду можно нажать кнопку **Сохранить сценарий**, чтобы сохранить операторы. Операторы будут помещены в сценарий. Если вы нажмете кнопку **Генерировать**, откроется окно Запустить сценарий.

Примечание: При работе с Центром управления для System 390 генерация операторов DDL выполняется иначе. Подробно эти отличия описаны в справке.

Вы можете выбрать, генерировать ли операторы DDL для выбранных схем или же для всех схем в базе данных. Затем можно отредактировать сценарий, если вы хотите внести изменения до его использования в среде производства. Для создания идентичных баз данных при помощи сгенерированных операторов DDL нужно просто взять сгенерированный сценарий и запустить его в новой среде.

Чтобы сгенерировать операторы DDL:

1. Выделите объект, для которого нужно сгенерировать операторы DDL, и щелкните правой кнопкой мыши.
2. Выберите **Генерировать DDL**. Появится окно Запустить сценарий.
3. Введите ID пользователя и пароль и нажмите кнопку **OK**. Будет создано задание с содержанием команды **db2look**. Появится окно сообщения DB2 с ID задания нового задания.
4. Нажмите кнопку **OK**, чтобы закрыть окно сообщения.
5. Для просмотра результатов задания и для просмотра содержимого сохраненного сценария, соответствующего заданию, используйте страницу Хронология заданий записной книжки Журнал.
6. Выберите задание и щелкните правой кнопкой мыши. Во всплывающем меню выберите **Показать результаты**. Откроется окно Результаты задания. Вывод команды **db2look** будет показан на панели Результат задания.
7. Чтобы создать сценарий результатов, выберите **Создать сценарий**. Появится окно Новый командный сценарий.
8. Сохраните новый сценарий, если вы собираетесь использовать его снова.

Фильтр

В Центре управления можно фильтровать информацию, выводимую на панели содержимого, или фильтровать информацию, получаемую из таблицы как реальный набор результатов. Количество выводимых или возвращаемых объектов можно ограничить, создав фильтры для одного или нескольких объектов. Если вы задали фильтр и хотите, чтобы снова выводились все объекты в дереве, нужно его очистить или удалить.

Фильтрация вывода

Чтобы уменьшить количество объектов, выводимых на панели содержимого, для более удобного управления:

1. Выберите значок Фильтр из панели инструментов панели содержимого в нижней части Центра управления, или же выберите пункт Фильтр из меню Вид.
2. Выберите критерии отбора объектов.
3. Включите переключатель Включить фильтр, чтобы активировать фильтр.

После этого при выборе объекта для просмотра его содержимого фильтр, который вы связали с объектом, ограничивает вывод на экран в соответствии с заданными ранее критериями.

Фильтрация получаемых данных

Чтобы уменьшить количество строк, возвращаемых запросом, и сократить время ответа, вы можете определить вывод, или набор результатов, выводящийся на панели содержимого при выборе объекта.

1. Выберите в дереве объект папки и щелкните правой кнопкой мыши.
2. Из всплывающего меню выберите **Фильтр**. Откроется окно Фильтр.
3. При помощи функции Фильтр определите набор критериев для получения строк объекта.

Определение фильтра для получения специального набора данных

Чтобы определить фильтр для получения специального набора данных:

1. В Центре управление раскройте папку Базы данных или папку Подсистемы в зависимости от вашей платформы.
2. Выберите объект, для которого вы хотите определить фильтр. Щелкните по этому объекту правой кнопкой мыши.
3. Из всплывающего меню выберите **Фильтр**. Откроется записная книжка Фильтр.
4. На странице Поиск задайте имя или другие описательные критерии фильтрации выбранного объекта. Результат фильтра - это набор результатов, соответствующий выбранному объекту и выводимый на панели содержимого Центра управления.

5. На странице Поиск выберите радиокнопку, чтобы задать либо одновременное выполнение всех условий, выбранных в полях на странице Поиск, либо выполнение хотя бы одного из условий.
6. На странице Дополнительно записной книжки Фильтр можно указать дополнительные критерии, отредактировав выведенный текст, чтобы еще сильнее ограничить количество возвращаемых строк.
7. Нажмите кнопку **OK**, чтобы использовать заданные вами критерии фильтрации.

Чтобы автоматически вызывать записную книжку Фильтр в зависимости от количества строк, выберите в меню Инструменты пункт Параметры инструментов. Переключатель **Установить фильтрацию при количестве строк, превышающем** позволяет заранее определить пороговое количество возвращаемых строк из любого выбора. При достижении порога появляется записная книжка Фильтр, чтобы вы могли ограничить текущий вывод на основе определенных критериев. В особенности это полезно, если таблица ранее не фильтровалась и неожиданно выросла. В зависимости от платформы и от типа данных запрос может возвратить миллионы строк, в то время как вам требуется только их малая часть.

Справка

Инструменты управления содержат подробную справку. Кнопка справки есть во всех окнах и записных книжках, а также на полосе инструментов меню. Можно получить как общую справку, так и справку о том, как заполнять поля и выполнять задачи. В меню справки вы можете также открыть указатель терминов, справочную информацию и информацию, содержащуюся в руководствах к программе.

Центр управления

Используйте Центр управления в качестве основного инструмента управления системами, экземплярами DB2, базами данных, объектами баз данных, такими как таблицы, производные таблицы и группы пользователей. Центр управления можно использовать также для доступа к подсистемам DB2 for OS/390. Чтобы все базы данных DB2 появились в Центре управления, их нужно внести в каталог. На рис. 1 на стр. 12 показаны основные возможности Центра управления. Из-за различий операционных систем Центр управления может выглядеть на вашей системе иначе, чем на рисунке.

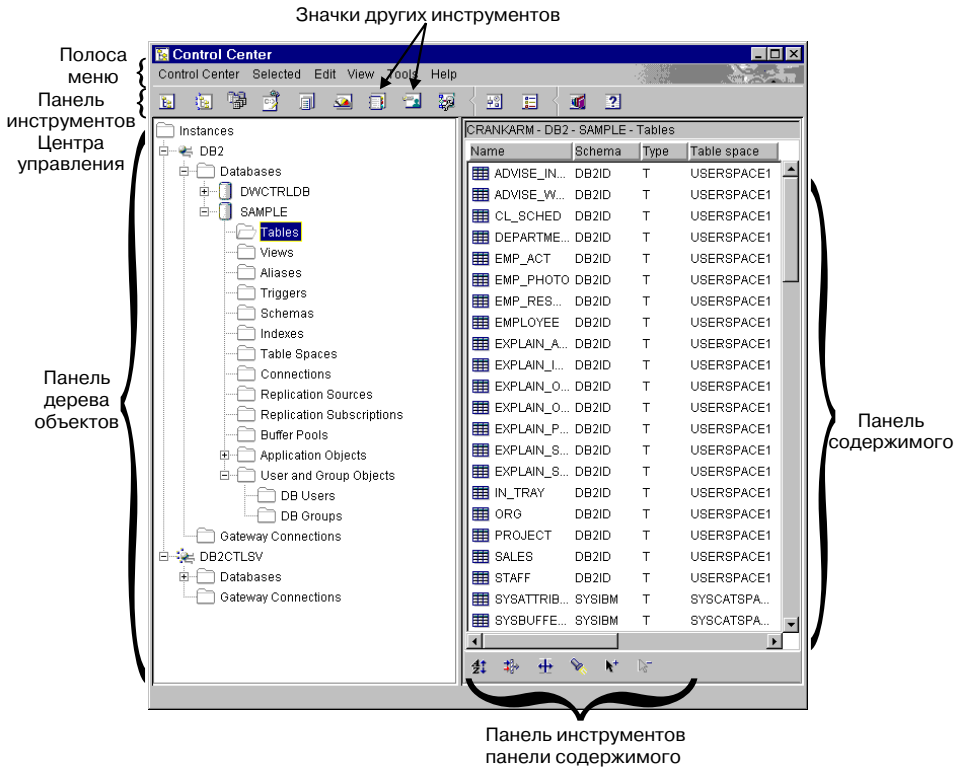


Рисунок 1. Возможности Центра управления

Основные элементы Центра управления

Центра управления содержит следующие основные элементы:

- Полоса меню. Полоса меню находится в верхней части экрана. Выбор меню из полосы меню позволяет выполнять множество функций, например, выключение инструментов DB2, обращение к графическим инструментам, к электронной справке и к информации о продукте. Вы можете ближе ознакомиться с этими функциями, щелкая по соответствующим элементам в полосе меню.
- Панель инструментов Центра управления. На панели инструментов Центра управления находятся значки Центра управления и других инструментов. Если поставить курсор на значок, появится всплывающая справка об этом значке.

  Параметры этих инструментов можно изменить, выбрав значок Параметры инструментов из панели инструментов Центра управления.
- Дерево объектов. Дерево объектов находится на левой панели экрана. Оно содержит значки всех серверов и объектов баз данных, которыми можно

управлять из Центра управления. Чтобы удаленный сервер баз данных появился на панели дерева объектов, его нужно предварительно внести в каталог. Некоторые объекты на панели дерева объектов содержат другие объекты. Значок плюс (+) слева от объекта означает, что объект свернут. Его можно раскрыть, щелкнув по этому значку. Значок минуса (–) слева от объекта означает, что объект раскрыт. Чтобы свернуть объект, щелкните по этому значку.

- Панель содержимого. Панель содержимого находится в правой части экрана. На этой панели показаны все объекты, содержащиеся в выбранном объекте на панели дерева объектов; например, если на панели дерева объектов выбрать папку таблиц, на панели содержимого будут показаны все имеющиеся таблицы. Если выбрать папку баз данных, на панели содержимого будут показаны все базы данных. Столбцы на панели содержимого можно отфильтровать, щелкнув по значку Фильтр на панели инструментов панели содержимого и задав требуемую информацию, или же выбрав на панели инструментов **Инструменты**, а затем **Параметры инструментов**. Необходимо, чтобы в диалоговом окне фильтра панели содержимого был включен переключатель **Включить фильтр**.
- Панель инструментов панели содержимого. Эта панель инструментов находится в нижней части панели содержимого. Она позволяет настроить вывод информации на панели содержимого. Эта панель инструментов - общий элемент управления, выводящийся в нижней части или же в боковой части окон подробного вида в программе.

При работе в Центре управления вы можете заметить жирную красную рамку вокруг некоторых полей. Эта рамка означает, что в поле необходимо ввести информацию. Как только вы выберете или введете значение, красная рамка исчезнет.

Использование настроенного центра управления в DB2 for OS/390

Настроенный центр управления на платформе DB2 for OS/390 используется в качестве собственного настраиваемого инструмента управления подсистемами, базами данных или объектами баз данных, такими как таблицы, производные таблицы и пользователи баз данных. Настроенный центр управления можно использовать для доступа к любым определенным вами объектам DB2 for OS/390.

Основные элементы Настроенного центра управления совпадают с перечисленными выше для Центра управления по умолчанию. Настроенный центр управления позволяет задавать объекты, которые вы хотите включить в свой Центр управления. Настроенное дерево можно сохранить и вызывать для управления объектами в DB2. Оно не заменяет дерево Центра управления, которое используется по умолчанию всеми пользователями, но полезно, если вам нужно обращаться к одному и тому же набору объектов при каждом вызове Центра управления. Можно создать любое необходимое количество

пользовательских деревьев, и в каждом будет собственный набор объектов, который можно упорядочить любым желаемым способом.

Использование настроенного дерева упрощает перемещение по заданной иерархической структуре объектов DB2 и позволяет сгруппировать связанные объекты. Например, можно создать дерево, в котором будут только таблицы с информацией о заработной плате.

Системы, которыми можно управлять

В Центре управления можно управлять объектами баз данных семейства продуктов DB2 Universal Database для платформ OS/2, Windows и UNIX. Информацию об установке и конфигурировании смотрите в книгах *Быстрый старт* для вашей платформы.

Вы можете также реплицировать данные из систем DB2 for AS/400, DB2 for VSE, DB2 for VM и DB2 for OS/390 в семейство продуктов DB2 Universal Database. Информацию о репликации между продуктами смотрите в руководстве *Replication Guide and Reference*.

Объекты, которыми можно управлять

Если вы хотите управлять объектами из Центра управления, их нужно добавить в дерево объектов. Если вы удалили базу данных или исключили ее из каталога вне Центра управления, но хотите выполнять над ней действия в Центре управления, ее нужно добавить в дерево объектов.

Ниже перечислены объекты DB2 Universal Database, которыми можно управлять из Центра управления:

- Системы
- Экземпляры
- Таблицы
- Производные таблицы
- Индексы
- Триггеры
- Пользовательские типы
- Пользовательские функции
- Пакеты
- Алиасы
- Объекты репликации
- Пользователи и группы

Ниже перечислены объекты DB2 for OS/390 Версии 5, которыми можно управлять из Центра управления:

- Пулы буферов

- Производные таблицы
- Таблицы каталога
- Группы хранения
- Алиасы
- Синонимы
- Пользователи DB2
- Положения
- Объекты программ (собрания, пакеты, планы, процедуры)
- Базы данных
- Таблицы
- Табличные пространства
- Индексы
- Источник репликации
- Регистрации репликаций

В DB2 for OS/390 Версии 6 в Центре управления можно управлять всеми объектами, перечисленными для Версии 5, а также:

- Схемами
- Триггерами
- Пользовательскими функциями
- Особыми типами

Чтобы увидеть, какие функции можно выполнять над каждым из этих объектов, выберите объект на панели объектов и щелкните правой кнопкой мыши. Появится всплывающее окно со списком функций.

Вывод систем в Центре управления

Чтобы вывести все установленные системы DB2, занесенные в каталог на вашей системе:

1. Раскройте дерево объектов, щелкнув по значку плюс (+) рядом с папкой **Системы**. Будут выведены значки, обозначающие локальный компьютер и все удаленные компьютеры. Локальная система представлена значком Локальный. Он появляется, только если локальный компьютер является сервером DB2. Если вы щелкнете по значку Локальный правой кнопкой мыши, во всплывающем меню будет опция под названием **Подключиться к серверу администратора**. Сервер администратора позволяет воспользоваться такими функциями, как мониторинг производительности и планирование. Он используется инструментами управления DB2 для выполнения служебных требований DB2; создается и запускается он автоматически. Имя сервера администратора DB2 по умолчанию зависит от платформы. Например, на платформах Windows и OS/2 используется имя “DB2DAS00”; в AIX - “db2as”.

2. Раскройте значок Локальный. В структуре дерева будет выведен экземпляр DB2 на локальном компьютере.

   В OS/2, Windows и поддерживаемых DB2 системах на основе UNIX каждую копию кода менеджера баз данных можно считать отдельным *экземпляром*, хранящимся в каталоге на вашем компьютере. В DB2 for OS/390 экземпляр называется подсистемой. При установке DB2 создается локальный экземпляр по умолчанию. На одном компьютере может быть несколько экземпляров. Эти экземпляры можно использовать для отделения среды разработки от среды производства, или для предоставления доступа к важной информации ограниченной группе пользователей. Можно также настроить каждый экземпляр базы данных для своей среды.

3. Раскройте значок **Экземпляры**. Для всех существующих баз данных будет выведены значки и имена.

Управление объектами DB2 for OS/390

При помощи Центра управления можно выполнять множество функций существующих продуктов DB2 for OS/390 Версии 5 и DB2 UDB for OS/390 Версии 6, например, создавать, изменять и отбрасывать объекты, а также запускать утилиты, реорганизующие или загружающие ваши данные. Но до того, как вы сможете управлять подсистемой DB2 for OS/390 в Центре управления, необходимо сначала добавить ее в дерево объектов, сконфигурировав соединение с ней.

Добавление подсистем DB2 for OS/390

Если у вас установлен Ассистент конфигурирования клиента, с его помощью можно легко конфигурировать соединения с системой DB2 for OS/390. Если у вас не установлен Ассистент конфигурирования клиента, надо конфигурировать соединение с системой DB2 for OS/390 вручную, при помощи процессора командной строки (CLP).

Ассистент конфигурирования клиента может найти все системы DB2 for OS/390, доступные в вашему клиенту в локальной сети. Если вы хотите добавить одну из систем DB2 for OS/390, можно использовать для этого мастер **по добавлению баз данных**, импортировать соединение при помощи профиля, или же добавить соединение вручную.

Если вы выбрали поиск в сети, нужно, чтобы в сети была установлена DB2 Connect и определено соединение с этой системой. Если вы выбрали использование профиля доступа, нужно выбрать соединение с сервером DB2 Connect, обозначающее систему из этого профиля. Если вы выбрали конфигурирование соединения вручную, вам нужно знать имя системы, протокол связи и параметры протокола связи, такие как имя хоста и номер порта для TCP/IP, или символическое имя назначения для SNA. Когда вы добавите систему DB2 for OS/390, на локальной системе Центра управления появятся объекты соединений с серверами DB2 Connect.

При добавлении системы DB2 for OS/390 Версии 5 или более новой она появится в своем собственном разделе Дерева объектов Центра управления. Чтобы увидеть объекты баз данных DB2 for OS/390 и другие объекты баз данных, находящиеся на какой-либо системе, раскройте дерево объектов из значка системы DB2 for OS/390, означающего эту систему DB2 for OS/390.

Чтобы увидеть список действий, которые можно выполнить над каким-либо объектом, выберите объект в дереве объектов и щелкните правой кнопкой мыши. Появится всплывающее меню с действиями, которые можно выполнить с этим объектом. Например, можно создать, изменить или отбросить производную таблицу, посмотреть ее содержание, изменить привилегии для нее и посмотреть список других объектов, связанных с ней. Подробную информацию о возможных действиях смотрите в электронной справке по объектам DB2 for OS/390.

Управление соединениями шлюза

Если сервер DB2 Connect внесен в каталог, в дереве объектов Центра управления под объектом экземпляра локальной системы выводится папка Соединения шлюза. В папке Соединения шлюза содержится иерархия объектов, используемая для управления соединениями с базами данных хоста и AS/400, записанными в локальный каталог. Действия, связанные с этими объектами управления соединениями, можно использовать для получения списка соединений с базами данных хоста и AS/400, для их принудительного отключения и для наблюдения за ними.

Дерево объектов под папкой Соединения шлюза используется для управления соединениями с базами данных хоста и AS/400, но не для задач управления базами данных. Если вам нужно добавить, изменить или удалить на локальной системе базу данных хоста или AS/400, можно использовать Ассистент конфигурирования клиента.

Функции, которые можно выполнять из Центра управления

Из Центра управления можно:

- Управлять объектами баз данных. Можно создавать, изменять и отбрасывать базы данных, табличные пространства, таблицы, производные таблицы, индексы, триггеры и схемы. Можно также управлять пользователями.
- Управлять данными. Можно загружать, импортировать, экспортировать, реорганизовывать данные и собирать статистику.
- Составлять расписания заданий. Задания - могут быть отложенными, выполняемыми или завершенными сценариями. Вы можете вносить в расписание запуск заданий в указанное время.
- Выполнять профилактическое обслуживание баз данных - создание резервных копий баз данных или табличных пространств и их восстановление.
- Наблюдать за производительностью и устранять неисправности.
- Реплицировать данные.

- Конфигурировать и настраивать экземпляры и базы данных.
- Управлять соединениями с базами данных, например, серверами и подсистемами DB2 Connect.
- Управлять программами.
- Анализировать запросы, просматривая планы доступа при помощи функции Объяснить SQL.
- Менять шрифт, используемый для вывода меню и текста в Центре управления. Можно выбрать другой имеющийся шрифт, а также изменить размер и цвет шрифта. Чтобы изменения вступили в силу, нужно перезапустить Центр управления.
- Запускать другие инструменты. Например, можно запустить Центр управления сателлитами или Командный центр.

Чтобы увидеть все действия, которые можно выполнить над объектом, выберите этот объект в дереве объектов или на панели содержимого и щелкните правой кнопкой мыши. Появится всплывающее меню, в котором показаны все функции, которые можно выполнять над этим объектами данного типа; например, если выбрать папку таблиц, можно будет создать новую таблицу при помощи мастера или самостоятельно, наблюдать за производительностью таблиц, фильтровать таблицы для вывода в панели содержимого и так далее. Функции, которые можно выполнять, зависят от выбранного объекта.

Чтобы выполнить над каким-либо объектом дополнительные функции, щелкните по нему правой кнопкой мыши на панели содержимого. Например, если выбрать на панели содержимого таблицу и щелкнуть правой кнопкой мыши, будет выведено всплывающее меню со списком функций, которые можно применить к этой таблице.

Создание новых объектов

Чтобы создавать новые объекты:

1. Раскройте папку баз данных. Типы объектов будут выведены в виде значков папок.
2. Щелкните по значку папки нужного объекта правой кнопкой мыши, например, щелкните по значку **Таблицы**. Будет выведено всплывающее меню. Для некоторых объектов предлагается два способа выполнения функции. Один из способов - использовать мастер. Не для всех доступных функций имеются мастера.
3. Выберите **Создать**. Поскольку для создания таблицы есть мастер, появляется две опции, одна из которых предлагает создать таблицу при помощи мастера. Если выбрать эту опцию, мастер попросит вас ввести необходимую информацию и посоветует, какие параметры выбирать. Мастер в особенности полезен для начинающих и для тех, кто редко создает объекты баз данных.

Работа с существующими объектами

Если щелкнуть по объекту на панели дерева объектов, например, по папке таблиц, на панели содержимого появятся все существующие таблицы. После этого можно выбрать таблицу, с которой вам нужно работать, и щелкнуть правой кнопкой мыши, чтобы вызвать какие-нибудь функции, которые нужно выполнить над данной таблицей.

Более подробную информацию об использовании Центра управления смотрите в его электронной справке, вызываемой через меню **Справка** или нажатием клавиши F1 в любом месте Центра управления.

Поиск объектов (только DB2 for OS/390)

Объект базы данных или экземпляра можно легко найти при помощи записной книжки Найти. Это позволяет:

- Найти объект, не просматривая всю структуру дерева Центра управления. Объект может находиться в базе данных или подсистеме, в табличном пространстве или среди баз данных, таблиц и поддерживающих объектов.
- Искать объекты (табличные пространства, таблицы и индексы) по нескольким базам данных внутри подсистемы.

Страница Найти записной книжки Найти служит для задания критериев поиска. Страница Дополнительно записной книжки Найти служит для дальнейшей настройки поиска. Отредактируйте текст на странице Дополнительно и добавьте или измените критерии поиска.

Чтобы найти объект, определенный в базе данных или в подсистеме DB2 for OS/390:

1. В Центре управления щелкните по какому-нибудь объекту правой кнопкой мыши. Во всплывающем меню выберите **Найти**. Откроется записная книжка Найти.
2. В поле Тип объекта выберите тип искомого объекта базы данных. Список доступных объектов назначения зависит от объекта, с которого вы начали поиск.
3. На странице Найти введите критерии поиска. Необходимо ввести хотя бы один критерий поиска; при поиске можно использовать символы подстановки.Символы переводятся в верхний регистр, если вы не использовали ограничители для символов нижнего регистра или расширенного набора символов.
4. Выберите радиокнопку на странице Найти, чтобы задать либо поиск при одновременном выполнении всех условий, выбранных в полях на странице Найти, либо поиск при выполнении хотя бы одного из условий.
5. Нажмите кнопку **OK**, чтобы использовать заданные критерии поиска. Результаты поиска выводятся в окне Результат поиска. Формат выходной таблицы зависит от типа искомого объекта.

6. Чтобы повторить поиск с теми же самыми или новыми критериями, нажмите кнопку **Применить**.
7. Вы можете выбрать строку в окне Результат поиска и щелкнуть по ней правой кнопкой, чтобы вывести всплывающее меню с дополнительными действиями, которые можно выполнить.

Центр управления сателлитами

Центр управления сателлитами - это набор инструментов, доступный из Центра управления DB2. Они позволяют конфигурировать собрания серверов DB2 и управлять ими из одной точки. Каждый сервер DB2, принадлежащий к группе, называется сателлитом. Управление сателлитами из одной точки означает, что DB2 можно скрыть ото всех пользователей сателлита DB2; таким образом, им не нужно будет ничего знать об управлении базами данных.

Организуйте в группы серверы DB2 с одинаковыми характеристиками, например, выполняемыми программами или конфигурацией баз данных, поддерживающей нужную программу. Подобие серверов DB2 определяется конфигурацией их баз данных, использованием и назначением.

Сгруппировав вместе серверы DB2, можно управлять группами серверов DB2 вместо того, чтобы по отдельности управлять каждым сервером DB2. Если нужно, чтобы дополнительные серверы DB2 выполняли ту же функцию, что и серверы DB2 существующей группы, можно добавить их в эту группу при помощи Центра управления сателлитами.

В Центре управления сателлитами можно создавать группы, сателлиты, версии программ, пакеты и параметры аутентификации. Можно также определять наборы кодов успешного завершения и выполнять другие функции, связанные с управлением средой сателлитов. Информация о среде сателлитов хранится в центральной базе данных, которая называется базой данных управления сателлитами. В этой базе данных, помимо прочего, записывается, какие сателлиты есть в среде, группа, к которой принадлежит каждый сателлит, и версии программ конечного пользователя, установленные на сателлите. Эта база данных находится на сервере DB2, который называется сервером управления DB2.

До того, как можно будет работать с Центром управления сателлитами, необходимо в Центре управления внести в каталог базу данных управления сателлитами (SATCTLDB). Когда это будет сделано, можно будет использовать Центр управления сателлитами для установки и обслуживания сателлитов, групп и пакетов, выполняемых сателлитами при синхронизации для их версии программы.

Для установки и обслуживания своей конфигурации баз данных каждый сателлит соединяется с базой данных управления сателлитами и загружает

пакеты, соответствующие его версии программы конечного пользователя. Сателлит локально выполняет эти пакеты, а затем возвращает результаты в базу данных управления сателлитами. Этот процесс загрузки пакетов, их выполнения, а затем возвращения результатов выполнения пакетов называется синхронизацией. Сателлит синхронизируется для согласованности с другими сателлитами, принадлежащими к его группе и выполняющими ту же версию программы конечного пользователя.

Командный центр

Чтобы запустить Командный центр, щелкните по значку Командный центр на панели инструментов.

Командный центр позволяет:

- Просматривать результат выполнения одного или нескольких операторов SQL и команд DB2 в окне результатов. Можно просмотреть результаты и сгенерировать по ним отчет.
- Создавать командные сценарии и сохранять их в Центре сценариев.Командный сценарий можно отредактировать и создать на его основе новый. Затем в Центре сценариев можно командный сценарий можно внести в расписание для запуска в любое заданное время.
- Выполнять операторы SQL, команды DB2 и команды операционной системы. При запуске команд DB2 из Командного центра перед командой не надо ставить префикс **DB2**. Для выполнения команды операционной системы на любом поддерживаемом языке сценариев операционной системы (например, REXX) перед ними надо ставить восклицательный знак (!). Использование Командного центра для запуска команд и операторов позволяет сразу выполнить много команд вместо того, чтобы вводить и запускать команды по одной.
- Быстро запускать инструменты управления DB2, например, Центр управления, из главной панели инструментов.
- Просматривать план выполнения и статистику, соответствующие оператору SQL, перед выполнением.

Центр сценариев

Центр сценариев можно запустить, выбрав его значок из панели инструментов Центра управления. Центр сценариев - это инструмент, позволяющий создавать сценарии, вводя набор команд и операторов; запуск сценария затем можно запланировать на нужное время. Можно импортировать созданные ранее сценарии или сценарии, сохраненные в Командном центре. Можно выбирать сценарии из набора сохраненных сценариев, а затем редактировать их для создания новых сценариев, копировать и удалять сценарии.

Сценарии можно редактировать в Центре сценариев или же вне Центра сценариев при помощи собственного редактора. При запуске сценария из Центра сценариев поддерживается запись результатов в Журнал.

Чтобы выполнять из сценария в Центре сценариев команды операционной системы:

1. Выберите **Сценарий —> Новый**. Откроется окно Командный сценарий.
2. В поле **Тип сценария** выберите радиокнопку **Команда операционной системы**.
3. Введите имя, описание и рабочий каталог сценария.
4. Введите нужные команды.
5. Нажмите кнопку **OK**.

В Центре сценариев можно просматривать информацию обо всех известных системе командных сценариях, такую, как описание и тип сценария, а также выполнять следующие действия:

- Создание командного сценария, содержащего команды DB2 и операционной системы
- Немедленный запуск командного сценария
- Составление расписания для запуска сценария позднее или регулярно с некоторым интервалом; например, можно создать сценарий, собирающий статистику по нескольким таблицам. Затем можно запланировать запуск этого задания на ночь. Можно запланировать автоматический запуск заданий через заданные интервалы, указав, когда запускать задание - часы, дни, недели, месяцы, несколько раз в неделю или несколько раз в месяц. Задание создается каждый раз, когда вы планируете сценарий или запускаете сценарий немедленно.
- Обращение к Журналу из панели инструментов для просмотра заданий, использующих какой-нибудь сценарий и для просмотра состояния всех запланированных заданий
- Редактирование сохраненного командного сценария

Использование существующего сценария в Центре сценариев

Чтобы использовать Центр сценариев с существующими сценариями, созданными не в Центре сценариев:

1. На панели инструментов **Центра управления** щелкните по значку **Центр сценариев**. Откроется Центр сценариев.
2. Выберите **Сценарий —> Импорт**. Откроется окно Просмотр файлов.
3. Выберите существующий файл сценария и нажмите кнопку **OK**. Откроется окно Командный сценарий. Сценарий будет выведен в нижней части окна - в редакторе сценариев. Заполните поля **Экземпляр**, **Имя сценария**, **Описание сценария** и **Рабочий каталог**, а также выберите **Тип сценария**.
4. Нажмите кнопку **OK**. Сценарий будет создан в Центре сценариев.

Внесение в расписание запуска сохраненного командного сценария

Чтобы внести в расписание сценарий:

1. Щелкните по значку **Центр сценариев** на панели Центра управления. Откроется Центр сценариев.
2. Щелкните правой кнопкой мыши по сценарию, который надо внести в расписание, и выберите **Расписание** во всплывающем меню. Откроется окно Планировщик.
3. Выберите частоту выполнения задания и действие по завершении, например, сообщение о завершении или запуск другого командного сценария.
4. Нажмите кнопку **OK**. Будет сформировано отложенное задание, за которым можно следить через Журнал.

Журнал

Журнал можно запустить, выбрав его значок на панели инструментов Центра управления. Журнал позволяет наблюдать за заданиями и просматривать их результаты. В Журнале можно также вывести хронологию восстановлений и сообщения DB2. Журнал позволяет:

- Следить за отложенными и выполняемыми заданиями, а также за хронологией выполнения заданий
- Просматривать результаты
- Смотреть хронологию восстановлений и оповещения
- Смотреть журнал сообщений DB2

Работа с заданиями

Журнал служит для работы с заданиями. Чтобы открыть Журнал:

1. Щелкните по значку **Журнал** в панели инструментов Центра сценариев. Откроется Журнал.
2. Чтобы увидеть задания, запуск которых запланирован на будущее, нажмите кнопку **Отложенные задания**. В списке заданий вы увидите ваше задание. Вы увидите также всю информацию о заданиях. Можно выполнять действия с отложенным заданием, например, изменить расписание для него, показать связанные с ним сценарии или немедленно запустить его. Если сохраненный сценарий будет изменен, все задания, зависящие от него, унаследуют новое поведение.

В Журнале можно также просматривать задания, выполняющиеся в данный момент, и хронологию заданий.

В окне Журнал есть также следующие страницы:

- Страница Восстановление. На этой странице выводится хронология восстановлений (подробности об операциях резервного копирования, восстановления и загрузки); здесь можно посмотреть журнал восстановлений.

- Страница Оповещения. На этой странице показаны все оповещения.
- Страница Сообщения. На этой странице показаны все сообщения инструментов управления DB2.

В электронной справке Журнала приводятся подробные указания по работе с заданиями и журналами.

Центр лицензий

Центр лицензий используется для вывода на экран состояния лицензии и информации об использовании лицензий для продуктов DB2, установленных на вашей системе. Центр лицензий можно также использовать для конфигурирования системы для правильного мониторинга лицензий. Центр лицензий позволяет:

- Добавить новую лицензию.
- Заменить пробную лицензию для продукта на постоянную лицензию.
- Просмотреть подробную информацию о лицензии.

При просмотре подробной информации о лицензии выводятся:

- Имя продукта
- Информация о версии
- Дата истечения срока
- Зарегистрированные пользователи
- Одновременно работающие пользователи
- Число разрешенных пользователей
- Число одновременно работающих пользователей
- Правила ограничения
- Количество процессоров (для DB2 Universal Database Enterprise Edition и Enterprise – Extended Edition).

Центр оповещений

Центр оповещений - это инструмент слежения за работой системы и предупреждения о возможных проблемах. В Центре оповещений можно задать автоматическое открытие окон для вывода наблюдаемых объектов, для которых превышен порог и зафиксировано состояние тревоги или предупреждения. Пороги задаются в Мониторе производительности, вызываемом из Центра управления. Цвет значка соответствует серьезности предупреждения. Красный значок означает тревогу. Желтый значок означает предупреждение.

Ассистент конфигурирования клиента

Ассистент конфигурирования клиента (CCA) - в первую очередь инструмент, содержащий мастера для помощи при конфигурировании связи клиентов с локальными или удаленными серверами. Но этот инструмент можно использовать и для того, чтобы легко конфигурировать серверы DB2 Connect.

Ассистент конфигурирования клиента позволяет работать со списком баз данных, с которыми могут соединяться ваши программы. Он вносит узлы и базы данных в каталог, скрывая от пользователя внутреннюю сложность этих задач.

В Ассистенте конфигурирования клиента можно выполнять следующие задачи:

- Добавлять, изменять и удалять записи соединений с базами данных.
- Проверять соединение с выбранной базой данных.
- Менять параметры конфигурации менеджера баз данных.
- Конфигурировать параметры CLI/ODBC.
- Связывать утилиты DB2 и другие программы с выбранной базой данных.
- Импортировать и экспортировать информацию о конфигурации. Это позволяет использовать конфигурацию уже сконфигурированного компьютера для конфигурирования новых компьютеров.
- Изменять пароль для ID пользователя, используемого для соединения с выбранной базой данных.

Ассистент конфигурирования клиента поддерживает следующие способы для добавления новых записей соединений с базами данных:

- Использовать профиль. Можно экспортировать профиль с ранее сконфигурированного компьютера и использовать его для конфигурирования новых компьютеров. Из Центра управления можно экспортировать профили серверов, а из Ассистента конфигурирования клиента - профили клиентов и серверов.
- Провести поиск в сети. Ассистент конфигурирования клиента может искать в сети системы DB2, на которых выполняется сервер администратора. Есть два режима поиска - Search и Known. Режим поиска Search зависит от ограничений конфигурации сети. (Обычно сетевые маршрутизаторы не передают требования поиска.) Для нахождения нужной системы сервера при поиске Known требуется лишь небольшой объем информации. Можно найти также системы хоста или AS/400, предварительно определенные на шлюзе.
- Конфигурировать соединение с базой данных вручную. Необходимо ввести всю информацию, но мастер упрощает эту задачу.

Монитор производительности

Монитор производительности выдает информацию о состоянии DB2 Universal Database и об управляемых ею данных. Это графическая утилита, которую можно настроить в соответствии со средой баз данных. Можно определить пороги или зоны, включающие предупреждения или тревогу, когда значения, полученные Монитором производительности, выходят за допустимые пределы.

Можно наблюдать за такими объектами DB2, как экземпляры, базы данных, таблицы, табличные пространства и соединения, выбрав нужный объект на панели дерева объектов или на панели содержимого и щелкнув правой кнопкой мыши. Затем надо выбрать запуск Монитора производительности.

Цвет значка наблюдаемого объекта меняется на зеленый, желтый или красный в зависимости от состояния монитора. Цвета обозначают серьезность проблем, которая определяется заданными порогами. Зеленый означает, что монитор запущен и все в порядке. Желтый - предупреждение; он означает, что наблюдаемый объект приближается к порогу. Красный обозначает тревогу - наблюдаемый объект достиг порогового значения. Можно использовать предопределенные мониторы, входящие в DB2, или создать свои собственные мониторы.

Чтобы увидеть, какую информацию собирает Монитор производительности, щелкните по объекту правой кнопкой мыши и выберите из всплывающего меню **Показать активность монитора**.

Используйте информацию из Монитора производительности, чтобы:

- Обнаруживать проблемы производительности
- Настраивать базы данных для наилучшей производительности
- Анализировать тенденции производительности
- Анализировать производительность программ баз данных
- Предупреждать возникновение проблем

Монитор производительности позволяет анализировать тенденции, создавая через определенные промежутки времени графическое отображение такой информации о базе данных, как активность дисков, использование пулов буферов, размеры предварительной выборки, использование блокировок и блокирование записей.

Этот инструмент используется, когда вы хотите проследить за существующей проблемой или за производительностью системы. Он позволяет делать моментальные снимки активности баз данных и данных производительности. Эти снимки можно использовать для сравнения по времени. Каждая точка на диаграмме обозначает значение некоторых данных. Указания по снятию снимков смотрите в разделе “Наблюдение за производительностью в отдельный

момент времени" на стр. 30. Эта информация может помочь определению и анализу возможных проблем и исключительных ситуаций на основании заданных порогов. Используйте этот инструмент, если вам необходимо знать производительность менеджера баз данных и его программ баз данных в отдельный момент времени и наблюдать временные тенденции. Используйте его также для получения наглядной справки о том, какие элементы находятся в состоянии тревоги. Это помогает определить, какие параметры требуют настройки. Затем можно посмотреть параметры, заданными для этого элемента, и изменить их, чтобы улучшить производительность.

Монитор событий

Монитор событий не делает моментальные снимки, а собирает информацию об активности баз данных за некоторый период времени. Собранная им информация дает хорошую сводку активности для отдельных событий баз данных, например, для соединения с базой данных или для оператора SQL. Мониторинг событий записывает состояние базы данных в момент определенного события. Он позволяет получить трассировку активности базы данных. Записи монитора событий сохраняются и анализируются после сбора нужных данных. Монитор событий используется, когда нужно знать, сколько времени заняла транзакция, или, например, сколько процессорного времени использовал оператор SQL. Для чтения данных, записанных монитором событий, можно затем использовать Анализатор событий.

Для каждого соединения с базой данных сохраняется одна запись события соединения. Для каждого оператора, выполненного в этом соединении, сохраняется запись оператора. Каждая запись события соединения указывает на строку в окне соединений Анализатора событий. В этом окне указана информация о всех программах, соединявшихся за наблюдаемый период времени, в том числе:

- Имя программы
- ID выполнения
- Время соединения
- Общее процессорное время
- Время ожидания блокировок
- Общее время сортировки
- Тупиковые ситуации
- Время отсоединения
- ID прикладной программы

Каждая запись события оператора указывает на строку в окне операторов Анализатора событий.

Использование инструментов наблюдения

Монитор производительности и Анализатор событий обеспечивают:

- Содержательное и гибкое собрание данных. Поддерживается более 200 переменных производительности, включая информацию о пулах буферов и вводе/выводе, блокировках и тупиковых ситуациях, сортировках, связи, агентах и регистрациях. Показываются данные о менеджерах баз данных, базах данных, табличных пространствах, таблицах, пулах буферов, соединениях, транзакциях и операторах SQL.
- Простое в использовании, интуитивное представление данных. Данные можно просматривать в реальном времени в виде простых диаграмм или текстов, удобно организованных в логические группы. Есть как подробные, так и сводные режимы просмотра с возможностью доступа к более подробной информации.
- Надежные возможности оповещений. Для любого показателя производительности можно определить исключительные ситуации, задав пороговое значение. Пороговые значения используются для наглядного обозначения; когда переменная производительности достигает порогового значения или превышает его, значение рисуется в соответствующей зоне диаграммы производительности. По достижении порогового значения можно задать выполнение любого из следующих действий или всех из них:
  - Центр оповещений отправляет вам извещение.
  - Звучит сигнал тревоги.
  - Запускается программа.
  - Выводится сообщение.

  Вы можете выбрать отсутствие оповещения.

На рис. 2 на стр. 29 показана совместная работа мониторов.

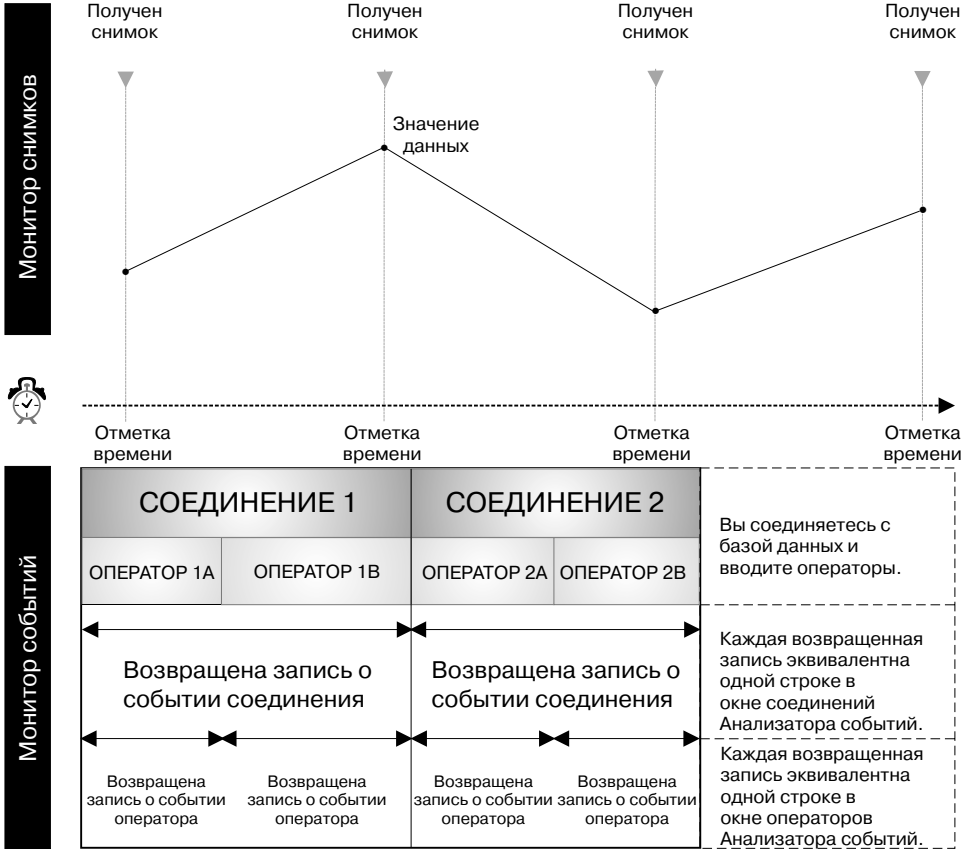


Рисунок 2. Сравнение: получение снимков и мониторинг событий. (Монитор событий, Анализатор событий)

Особенности наблюдения за базой данных и ее настройки

Прежде чем начать наблюдение за базой данных и ее настройку, необходимо:

- Определить свои цели. Например, вы хотите понять, как программы используют ресурсы на уровне экземпляра в заданный момент времени, чтобы, например, проверить, уменьшаются ли показатели одновременности базы данных при запуске какой-то программы. Или вы хотите понять, какие происходят события на уровне экземпляра при выполнении какой-то программы, например, падает ли общая производительность при выполнении определенной программы.
- Определить, какую информацию следует анализировать. Например, чтобы понять, связаны ли ″узкие места″ с аппаратным обеспечением, можно сделать снимки активности баз данных или активности табличных пространств, пулов

буферов и ввода/вывода. Чтобы понять, связаны ли ″узкие места″ со средой, нужно при помощи Анализатора событий выяснить, не имеет ли места следующее:

  – Слишком много заданий баз данных должно по расписанию работать во время высокой загруженности
  – Слишком много подключений пользователей
  – Разделение баз данных (для баланса загрузки аппаратного обеспечения) плохо оптимизировано
  – Сервер используется не только в качестве сервера баз данных

  Примеры некоторых наблюдаемых эффектов:

  – Медленная обработка запросов/ответов
  – Запланированные задания не заканчиваются вовремя
  – Истечение сроков ожидания программ

- Решить, будете ли вы использовать предопределенные в DB2 мониторы или создавать свои собственные.

В следующем разделе описано, как делать снимки и как при помощи Центра оповещений отслеживать проблемы, связанные с производительностью.

Наблюдение за производительностью в отдельный момент времени

Если вам нужно собирать сложные данные и анализировать их для выявления возможных проблем, используйте Монитор производительности, чтобы делать снимки системы и следить за изменением показателей производительности за некоторое время.

Этот инструмент позволяет:

- Строить диаграммы информации о производительности
- Задавать частоту снимков производительности
- Просматривать результаты вычислений производительности
- Определять пороговые значения и пороговые действия
- Генерировать и сохранять оповещения
- Просматривать сводную информацию (например, по всем базам данных)

Захватываются следующие типы информации:

- Информация о длительных действиях (например, активность базы данных, если программа выполняется слишком долго).
- Счетчики, отслеживающие информацию о текущем уровне активности (например, количество открытых указателей для базы данных).

- Совокупная информация об активности базы данных (например, максимальное количество соединений за время активности экземпляра базы данных или общее количество операторов SQL, выполненных для отдельной базы данных).

Снимки с указанными интервалами дают график текущего состояния активности менеджера баз данных и его программ. Эта информация используется, чтобы:

- Обнаруживать проблемы производительности
- Анализировать тенденции производительности
- Настраивать параметры конфигурации менеджера баз данных и параметры конфигурации баз данных
- Анализировать производительность программ баз данных

Доступна информация о производительности следующих объектов баз данных:

- Экземпляры
- Базы данных
- Таблицы
- Табличные пространства
- Соединения с базами данных

Для каждого пункта можно наблюдать за множеством переменных производительности. Описание всех переменных производительности смотрите в справочник по переменным производительности в электронной справке, доступной из меню **Справка** любого окна Монитора снимков. Эти переменные собраны по категориям. Существуют следующие категории:

- Экземпляр: Агенты, Соединения, Сортировка
- База данных: Блокировка и тупиковая ситуация, Пул буферов и ввод/вывод, Соединения, Сортировка, Активность операторов SQL
- Таблица: Таблица
- Табличное пространство: Пул буферов и ввод/вывод
- Соединения с базами данных: Пул буферов и ввод/вывод, Блокировка и тупиковая ситуация, Сортировка, Активность операторов SQL

Более подробную информацию о генерации снимков смотрите в электронной справке.

Предопределенные мониторы

Монитор производительности DB2 содержит набор предопределенных мониторов, которые можно использовать в готовом виде или же скопировать и изменить в соответствии с вашими потребностями. Они обеспечивают полный набор вычислений производительности. Имя, уравнение и текстовое описание монитора производительности, поставляемого IBM, изменить нельзя, однако

можно изменить пороговые значения и действия оповещений. Используйте предопределенные мониторы, чтобы научиться наблюдать за производительностью и создавать собственные мониторы, скопировав предопределенный монитор и добавив или удалив переменные производительности из этой копии.

С DB2 поставляются следующие предопределенные мониторы:

- Монитор объема. Этот монитор используется для получения информации о емкости системы. Можно регулярно проверять его, наблюдая за общим использованием системы за какое-то время.
- Монитор сортировок. Этот монитор используется для проверки, правильно ли заданы параметры кучи сортировки и порога кучи сортировки. Его следует запускать при первом запуске системы, в периоды пиковой активности или при изменении программ.
- Монитор блокировок. Этот монитор используется для определения количества блокировок, происходящих в системе, а также контроля правильности задания параметров списка блокировок.
- Монитор настройки кэша. Этот монитор используется для оптимизации использования кэша. Наблюдая за этими значениями в пиковые периоды, можно определить, нужно ли увеличить размер кэша.
- Монитор пула буферов. Этот монитор используется с маленькими таблицами для определения, требуются ли им собственные пулы буферов.
- Монитор тупиковых ситуаций. Этот монитор используется для выяснения, не попадают ли программы в тупиковые ситуации.
- Монитор настройки FCM (Fast Communication Manager - менеджер быстрой связи). Этот монитор используется для определения, сколько процентов памяти используется для передачи информации между узлами.
- Монитор настройки предварительных выборок. Этот монитор используется для контроля достаточности параметров предварительных выборок в системе.
- Монитор производительности дисков. Этот монитор используется для наблюдения за вводом и выводом. Он содержит переменные производительности, отражающие производительность дисков на уровнях таблиц и табличных пространств.
- Монитор глобальной памяти. Этот монитор используется для наблюдения за использованием памяти прикладными программами.
- Монитор долго выполняемых запросов. Этот монитор используется для определения, почему запрос выполняется слишком долго.
- Монитор соединений шлюза. Этот монитор используется для наблюдения за соединениями с серверами DB2 Connect.

Примеры использования предопределенных мониторов смотрите в электронной справке по наблюдению за производительностью.

Чтобы посмотреть список доступных мониторов, в Центре управления щелкните правой кнопкой мыши по папке **Системы** и выберите **Список мониторов** из всплывающего меню. Откроется окно Список мониторов. В нем перечислены мониторы, доступные на сервере JDBC, с которым вы сейчас соединены. Для каждого монитора указаны имя монитора, описание, состояние, создатель и является ли он монитором по умолчанию. “Состояние мониторов” означает состояние мониторов на локальной системе, а не на сервере JDBC. Пометка “По умолчанию для” означает монитор по умолчанию на уровне экземпляра, базы данных, таблицы, табличного пространства или соединений. Для предварительно определенных мониторов столбец “Создан” содержит **NULLID**. В правой части окна находятся кнопки, позволяющие выполнять над мониторами различные действия. Дополнительную информацию о сервере JDBC смотрите в разделе “Выполнение Центра управления в виде апплета Java” на стр. 49.

Можно выбрать, какой монитор запускать в качестве монитора по умолчанию для объекта.

Если запущен монитор производительности, можно щелкнуть по значку Центр оповещений на панели инструментов, чтобы увидеть состояние всех наблюдаемых объектов, находящихся в состоянии тревоги в связи с превышением каких-либо своих пороговых значений. Они появляются только на то время, в течение которого превышен порог.

Если вы хотите постоянно видеть наблюдаемые объекты, можно оставить открытым окно Центра оповещений или же оставить открытым на странице сводки окно **Показать монитор** и следить за красными или желтыми элементами в нем. Можно также задать в параметрах Центра управления, чтобы Центр оповещений автоматически открывался при получения нового предупреждения или тревоги. В Центре оповещений можно также временно приостановить оповещения, не прерывая работу мониторов.

Действия в случае появления объекта в Центре оповещений

В Центре оповещений можно задать автоматическое открытие для вывода наблюдаемых объектов, находящихся в состоянии тревоги или предупреждения (то есть с превышенными порогами). Этот параметр можно изменить в окне Параметры инструментов.

Если вы видите объект в Центре оповещений, щелкните по объекту правой кнопкой мыши и выберите **Монитор производительности —> Показать монитор**, чтобы просмотреть подробную информацию о производительности для этого объекта базы данных.

Анализ событий в течение периода времени

Анализатор событий - это еще один инструмент производительности DB2. Он используется, если вам нужна диагностическая информация о произошедшем

событии. Анализатор событий используется вместе с монитором событий. Например, можно при помощи монитора событий отслеживать активность базы данных при открытой базе данных, например, соединения, транзакции, операторы и тупиковые ситуации. Монитор событий может записывать совокупные данные по производительности, которые заносятся в журнал при отключении программы от базы данных.Когда монитор событий создает файл монитора событий, информацию о производительности можно смотреть при помощи Анализатора событий.

Инструменты монитора событий позволяют:

- Создавать мониторы событий для наблюдения за теми типами событий баз данных, которые представляют для вас интерес.
- Активировать монитор событий, чтобы начать собирать данные о событиях. Данные записываются в файл.
- Останавливать сбор монитором событий данных о событиях.
- Просматривать сводную информацию типа трассировки, выводимую монитором событий.
- Удалять монитор событий, если он больше не нужен. Можно также задать опцию очистки его файлов трассировки.
- Выводить список мониторов событий, связанных с базой данных.
- Просматривать определение монитора событий.

Анализатор событий позволяет просматривать данные, генерируемые монитором событий для следующих типов событий:

- Активность соединения с базой данных (отрезок времени между соединением и отсоединением)
- Транзакции (единицы работы)
- Выполнения операторов SQL
- Выявление тупиковых ситуаций

Анализатор событий

Можно создать монитор событий для следующих типов событий и затем просматривать собранную информацию в Анализаторе событий, однако лучше для просмотра этих данных использовать программу `db2evmon` (описанную в книгах *Command Reference* и *System Monitor Guide and Reference*):

- Тупиковые ситуации
- Активность баз данных
- Активность табличных пространств
- Активность таблиц
- Активность операторов

Чтобы анализировать данные событий при помощи монитора событий и Анализатора событий, выполните следующие действия. Это только один из возможных примеров создания монитора событий для событий соединений и операторов. Чтобы создать монитор событий:

1. В командной строке Командного центра введите **db2emcrt**. Откроется окно Монитор событий.
2. Щелкните по **Монитору событий** и выберите из меню Создать. Откроется окно Создать монитор событий.
3. Введите в соответствующем поле имя создаваемого монитора событий. Имя нового монитора событий не должно совпадать с именем уже существующего монитора событий. Пробелы в имени не допускаются.
4. Только в продукте DB2 Universal Database Enterprise - Extended Edition выберите из выпадающего списка “На узле ” узел, на котором будут находиться файлы монитора событий.
5. Только в продукте DB2 Universal Database Enterprise - Extended Edition выберите область действия монитора событий. По умолчанию предполагается Глобальная область действия.
6. Включите один или несколько переключателей, чтобы задать тип наблюдаемых событий. Обратите внимание на то, что тип событий Тупиковые ситуации выбран по умолчанию.
7. Укажите, когда нужно запустить монитор. По умолчанию предполагается “Запустить сейчас”.
8. Определите одно или несколько условий для соединений, операторов или транзакций, которые будут управлять наблюдением на этих уровнях.
9. Укажите путь (имя каталога), куда монитор будет записывать файлы с данными событий.
10. Нажмите кнопку **Опции**, чтобы открыть окно **Задание опций файлов монитора событий**. Эти опции определяют, как обрабатывать вывод монитора; они могут повлиять на производительность монитора событий.
11. Нажмите кнопку OK, чтобы создать монитор, или Отмена, чтобы выйти без создания монитора.
12. Выключите наблюдение за событиями, щелкнув правой кнопкой мыши по монитору событий и выбрав из всплывающего меню **Остановить наблюдение за событиями**.

    Это заставит монитор событий записать файл трассировки. Если монитор не выключен, информация записывается на диск, только когда заполнен буфер или завершены все соединения. В окне Мониторы событий можно просматривать данные результатов событий, щелкнув правой кнопкой по созданному монитору событий и выбрав из всплывающего меню **Просмотр файлов монитора событий**. Откроется окно Просмотр периодов наблюдений.

Для доступа к данным событий из Анализатора событий:

1. Чтобы запустить Анализатор событий, в командной строке Командного центра введите **db2eva**. Откроется окно Анализатор событий.
2. В поле Путь введите каталог, где хранятся файлы данных. Если эти файлы не перемещались, это будет путь, заданный при создании монитора событий. Если файлы перемещали, задайте этот каталог. Для вывода списка существующих каталогов нажмите кнопку ....

   Примечание: Если файлы данных хранятся на удаленном компьютере, для просмотра их надо переслать на ваш локальный компьютер. В зависимости от размера файлов передача может занять некоторое время. Файлы можно переписать в любой локальный каталог. Не обязательно выбирать тот же путь, который использовался при их создании.
3. Нажмите кнопку **OK**, чтобы просмотреть находящиеся в этом каталоге файлы данных, или **Отмена**, чтобы выйти. Откроется окно Просмотр периодов наблюдений.
4. Щелкните правой кнопкой мыши по наблюдаемому периоду и выберите из всплывающего меню **Открыть как —> Соединения**. Откроется окно Просмотр соединений. В нем выводится список соединений за сеанс наблюдения за событиями. (В списке может быть несколько соединений. Нужное соединение не обязательно будет первым в списке.)
5. Щелкните правой кнопкой мыши по соединению и выберите из всплывающего меню **Открыть как —> Операторы**. Откроется окно Просмотр операторов SQL. В нем выводятся все операторы для выбранного соединения. Для каждого оператора приводится информация в следующих столбцах:
   - **Операция**
   - **Имя пакета**
   - **Создатель**
   - **Время запуска**
   - **Истекшее время**
   - **Общее процессорное время**
   - **Текст**

Подробные инструкции по созданию мониторов событий и просмотру данных результатов событий смотрите в электронной справке для монитора событий и Анализатора событий.

Анализ операторов SQL

Можно просматривать план доступа для объясненных операторов SQL в виде диаграммы и использовать эту информацию при настройке запросов SQL для наилучшей производительности.

В диаграмме плана доступа выводится подробная информация о:

- Таблицах (со связанными с ними столбцами) и индексах
- Операторах (например, просмотра, сортировки и объединения таблиц)
- Табличных пространствах и функциях

До Версии 6 для просмотра планов доступа использовался инструмент Наглядное объяснение. Теперь Наглядное объяснение нельзя запускать из командной строки в качестве отдельного инструмента, но можно вызывать *функцию* наглядного объяснения из различных объектов баз данных в Центре управления и из Командного центра. В этом разделе для обозначения такой возможности используется термин *функция наглядного объяснения*.

Функция наглядного объяснения используется, чтобы:

- Просматривать статистику, использовавшуюся во время оптимизации. Можно затем сравнить эту статистику с текущей статистикой каталога, чтобы понять, можно ли улучшить производительность, повторно связав пакет.
- Выяснить, использовался ли для доступа к таблице индекс. Если индекс не использовался, функция наглядного объяснения поможет выяснить, какие столбцы можно проиндексировать для выигрыша в производительности.
- Просматривать эффект различных вариантов настройки, сравнивая версии диаграммы плана доступа для указанного запроса до и после настройки.
- Получить информацию о каждой операции в плане доступа, включая общую примерную стоимость и количество полученных строк.

Повышение производительности запроса

Функцию наглядного объяснения можно использовать для анализа и помощи при настройке операторов SQL. Она позволяет графически просматривать план доступа для объясненных операторов SQL. Таблицы, индексы и все операции на них представлены в виде узлов, а поток данных представлен связями между узлами. Полученную из этой диаграммы информацию можно использовать для поиска вариантов настройки производительности запросов SQL.

Функция наглядного объяснения собирает информацию о том, как скомпилированы операторы SQL. Эта информация позволяет понять план и возможную производительность выполнения операторов SQL.

Эта информация может помочь:

- Писать прикладные программы.

- Создавать базы данных.
- Выяснять, как объединены две таблицы: использованный метод объединения, порядок, в котором объединены таблицы, выполнялись ли сортировки, и если да, то какие.
- Определять способы повышения производительности операторов SQL (например, создание нового индекса).
- Просматривать статистику, использовавшуюся во время оптимизации. Можно затем сравнить эту статистику с текущей статистикой каталога, чтобы понять, можно ли улучшить производительность, повторно связав пакет. Это также помогает понять, улучшится ли производительность от сбора статистики.
- Выяснить, использовался ли для доступа к таблице индекс. Если индекс не использовался, функция наглядного объяснения поможет выяснить, какие столбцы можно проиндексировать для выигрыша в производительности запросов.
- Просматривать эффект от использования различных способов настройки для наилучшей производительности, сравнивая версии диаграммы плана доступа для указанного запроса до и после настройки.
- Получать информацию о каждой операции в плане доступа, включая общую примерную стоимость и количество полученных строк.

После использования функции наглядного объяснения при просмотре плана доступа для объясненного оператора SQL вы можете определить, что производительность данного запроса может улучшиться с использованием индекса. Для получения рекомендуемых индексов для этого запроса воспользуйтесь мастером по индексам или режимом объяснения RECOMMENDED_INDEXES. Для получения дополнительной информации о мастере по индексам войдите в Центр управления и откройте Информационный центр.

Дополнительную информацию о режиме RECOMMENDED_INDEXES EXPLAIN смотрите в книге *Administration Guide: Performance*.

Анализ простого динамического оператора SQL

В этом разделе приводится простой пример анализа динамического запроса SQL.

1. В Центре управления щелкните правой кнопкой мыши по базе данных SAMPLE и выберите **Объяснить SQL** из всплывающего меню. Откроется окно Объяснить оператор SQL.
2. В поле **Текст SQL** введите следующий оператор SQL:

   ```
   select * from staff order by name
   ```

3. Нажмите кнопку **OK**. Откроется окно Диаграмма плана доступа. На диаграмме изображен путь, выбранный оптимизатором в качестве наиболее эффективного для получения результатов запроса.

4. Необязательно: Дважды щелкните по любому узлу (например, по узлу оператора RETURN). Откроется окно Подробности оператора, в котором будут выведены подробности для этого оператора.

Объясненный оператор SQL автоматически сохраняется. Чтобы позже просмотреть его:

1. В Центре управления щелкните правой кнопкой мыши по базе данных SAMPLE и выберите из всплывающего меню **Показать хронологию объясненных операторов**. Откроется окно Хронология объясненных операторов.
2. Найдите нужную запись. Чтобы увидеть ранее объясненный оператор SQL, посмотрите на столбец **Текст SQL**.
3. Щелкните по записи правой кнопкой мыши и выберите из всплывающего меню **Показать план доступа**. Откроется окно Диаграмма плана доступа.

В электронной справке для Наглядного объяснения (доступного из меню **Справка**) приводится подробная информация о том, как улучшить производительность операторов SQL при помощи окна Диаграмма плана доступа. В электронной справке также приведены подробные примеры, помогающие научиться работе с этой функцией.

Управление удаленными базами данных

В этом разделе описывается, как:

- Добавить удаленную систему
- Добавить для этой системы экземпляр, с которым вы хотите работать
- Добавить базу данных в этом экземпляре, с которой вы хотите работать

Сначала DB2 проверяет по каталогу узла (содержащий записи для всех серверов, с которыми может соединиться клиент базы данных, и использовавшийся в соединении протокол связи), известна ли уже удаленная система. Если удаленная система неизвестна, а система, экземпляр или база данных находится на удаленном компьютере, необходимо сконфигурировать ваш компьютер в качестве клиента удаленной системы.

После установки DB2 при помощи Ассистента конфигурирования клиента можно искать в сети системы, экземпляры и базы данных и конфигурировать связь с ними. Затем удаленную систему нужно внести в каталог. При этом в каталоге узла создается запись для этой системы, и ее экземпляры и базы данных можно будет сделать известными. Затем нужно внести в каталог экземпляры и базы данных для этой системы и таким образом создать для них записи в каталоге узла и каталоге баз данных, соответственно. Это создаст для них записи в каталоге узла и каталоге баз данных, соответственно. Когда

конфигурирование будет завершено, удаленные системы будут выводиться в Центре управления, и с ними можно будет работать.

Чтобы добавить удаленную систему:

1. В Центре управления щелкните правой кнопкой мыши по объекту **Системы** и выберите во всплывающем меню **Добавить**. Откроется окно Добавить систему.
2. Введите имя системы в поле **Имя системы**.

   Если параметр конфигурации **Discover** имеет значение **search**, и параметр конфигурации **discover comm** не пуст, можно нажать кнопку **Обновить**, чтобы получить список удаленных систем. Затем выберите систему из списка под полем **Имя системы**.
3. Введите имя удаленного экземпляра в поле **Имя удаленного экземпляра**.
4. Из списка **Операционная система** выберите тип операционной системы для удаленной системы.
5. Выберите протокол, который нужно использовать для связи с удаленными положениями. Для локальной системы автоматически выбирается **Локальный** - единственный доступный протокол. Для удаленных систем доступны следующие возможные протоколы:
   - APPC
   - IPX/SPX
   - NetBIOS
   - TCP/IP
   - Именованный конвейер (только для операционных систем Windows NT и Windows 9x)

   В окне списка появятся только установленные в данный момент на компьютере протоколы.
6. Введите необходимые параметры протокола.
7. Введите комментарий к этой системе.
8. Нажмите кнопку **Применить**, чтобы добавить систему в каталог узла.

Затем добавьте экземпляр, с которым вы хотите работать на этой системе:

1. В Центре управления щелкните правой кнопкой мыши по объекту **Экземпляры**, принадлежащему только что добавленной системе.
2. Во всплывающем меню выберите **Изменить**. Откроется окно Добавить экземпляр.
3. Введите в полях необходимые значения.
4. Чтобы вывести список доступных экземпляров, нажмите кнопку **Обновить**.
5. Выберите экземпляр, с которым хотите работать.
6. Нажмите кнопку **Применить**, а затем кнопку **Закрыть**.

Наконец, добавьте базу данных, с которой вы хотите работать в этом экземпляре:

1. В Центре управления щелкните правой кнопкой мыши по объекту **Базы данных**.
2. Во всплывающем меню выберите **Изменить**. Откроется окно Добавить базу данных.
3. Введите имя базы данных, тип протокола связи, и, если хотите, алиас. В данном случае алиас - это альтернативное имя для базы данных.
4. Чтобы вывести список доступных баз данных для этого экземпляра, нажмите кнопку **Обновить**.
5. Выбрать базу данных.
6. Нажмите кнопку **Применить**, а затем кнопку **Закрыть**.

Управление пользователями

Вам, как администратору базы данных, может потребоваться изменять тип доступа пользователей к данным или ограничивать видимую ими область данных.Ниже описано, как при помощи инструментов управления управлять полномочиями и привилегиями баз данных для объектов баз данных.

Полномочия к базам данных предполагают действия надо всей базой данных. При создании базы данных некоторые полномочия автоматически предоставляются всем имеющим доступ к базе данных. Например, всем пользователям предоставляются полномочия CONNECT, CREATETAB, BINDADD и IMPLICIT_SCHEMA. *Привилегии* баз данных предполагают действия над различными объектами в базе данных. При создании базы данных некоторые привилегии автоматически предоставляются всем имеющим доступ к базе данных. Например, всем пользователям предоставляется привилегия SELECT на производные таблицы каталога и привилегии EXECUTE и BIND на все успешно связанные утилиты.

Привилегии и полномочия вместе позволяют управлять доступом к экземпляру и его объектам баз данных. Пользователям доступны только объекты, для которых у них есть соответствующие права - то есть требуемые привилегии или полномочия.

Предоставление и отзыв полномочий и привилегий

При помощи инструментов управления DB2 можно предоставлять пользователям привилегии на базы данных, табличные пространства, таблицы, производные таблицы и схемы и отзывать их.

1. В Центре управления щелкните правой кнопкой мыши по базе данных, таблице, производной таблице, схеме или индексу, для которых вы хотите

предоставить или отозвать привилегии. Выберите во всплывающем меню **Полномочия** или **Привилегии**. Откроется окно Полномочия или окно Привилегии.

2. Выберите страницу **Пользователь**, чтобы работать с полномочиями или привилегиями пользователей, или страницу **Группа**, чтобы работать с полномочиями или привилегиями групп.
3. Выберите одного или нескольких пользователей или групп. Чтобы добавить в список пользователя или группу, нажмите кнопку **Добавить пользователя** или **Добавить группу**.
4. В нижней части окна выберите для каждого отдельного полномочия или привилегии **Да**, **Нет** или **Предоставлять права**. **Предоставлять права** выводится только для объектов, для которых эта опция доступна.
5. Когда закончите, нажмите кнопку **Применить**.

Если вы хотите просмотреть или изменить список объектов, для которых у конкретного пользователя есть авторизация, можно выбрать пользователя, щелкнуть правой кнопкой мыши, а затем добавить, изменить или удалить авторизацию на объект.

Перемещение данных

DB2 включает в себя утилиты Import, Export и Load, помогающие перемещать в таблицу данные из готовых источников. В этом разделе приводится краткий обзор перемещения данных. Более подробную информацию о перемещении данных смотрите в справочном руководстве *Data Movement Utilities Guide and Reference*.

Утилита Import берет данные из входного файла и вставляет их в таблицу или производную таблицу. В этом случае входной файл содержит данные, взятые из существующего источника данных, например, файла Lotus 1–2–3 или файла ASCII. Утилиту импорта можно использовать также для повторного создания таблицы или производной таблицы, ранее сохраненной при помощи утилиты Export. Ниже описано, как импортировать данные.

Если у вас есть входной файл в поддерживаемом формате, используйте записную книжку Импорт, чтобы вставить данные из этого файла в существующую таблицу. Если в таблице уже есть данные, можно заменить их данными из файла или добавить данные из файла к уже существующим.

Записную книжку Импорт можно также использовать для создания новой таблицы, заполненной данными из входного файла, или для удаления существующих строк из выбранной таблицы и заполнения ее данными из входного файла.

Чтобы импортировать файл в существующую таблицу:

1. Откройте страницу Файл записной книжки Импорт.
2. Необязательно. Укажите имя импортируемого файла.
3. Необязательно. Получите большие объекты.
4. Необязательно. Укажите опции импорта столбцов.
5. Нажмите кнопку **OK**.

Чтобы открыть страницу Файл записной книжки Импорт:

1. В Центре управления раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните по папке **Таблицы**. Все существующие таблицы появятся на панели содержимого.
3. Щелкните по таблице на панели содержимого правой кнопкой мыши и выберите из всплывающего меню **Импорт**. Появится записная книжка Импорт с открытой страницей Файл.

Чтобы задать опции файла:

1. В поле **Импортировать файл** страницы Файл введите имя файла с данными, которые нужно импортировать.
2. Укажите тип импортируемого файла, выбрав одну из следующих опций:
   - Формат ASCII без ограничителей (ASC)

     Данные ASCII без ограничителей - это данные, выравненные по столбцам.
   - Формат ASCII с ограничителями (DEL)

     Данные ASCII с ограничителями - часто используемый способ хранения данных, в котором значения столбцов разделены пользовательским символом-разделителем, например, запятой.
   - Формат WSF
   - Интегрированный формат обмена (IXF)

     PC/IXF - это структурированное описание таблицы или производной таблицы базы данных. Данные, экспортированные в формате PC/IXF, можно импортировать или загрузить в базу данных другого продукта DB2 Universal Database.

   Поддерживаемые продукты и версии перечислены в электронной справке.
3. Необязательно: Укажите модификаторы типа файла, нажав соответствующую кнопку **Опции**. Откроется окно Опции для этого формата.
4. Выберите **Режим импорта**. Возможные режимы импорта зависят от выбранного типа файла.
5. Необязательно: В поле **Принимать записи** введите количество записей, которое можно импортировать до принятия изменений.
6. Необязательно: В поле **Перезапуск** введите количество записей в файле, которые нужно пропустить перед началом импорта.
7. Необязательно: В поле **Составной** введите число, означающее, сколько будет выполнено операторов SQL (в выполняемом блоке).

8. Необязательно: Включите переключатель **Вставлять подразумеваемый разделитель целой и дробной части в данные десятичного формата (IMPLIEDDECIMALPOINT)**.
9. В поле **Файл сообщений** введите имя файла для предупреждений и сообщений об ошибках процесса импорта.

Чтобы получить большие объекты из отдельных файлов, используйте страницу Большие объекты записной книжки Импорт, где можно задать получение больших объектов из пути или путей, где хранятся их файлы:

1. Включите переключатель **Получить большие объекты (LOB) в отдельных файлах (LOBSINFILE)**, чтобы включить опции на странице Большие объекты.
2. Укажите положение отдельных файлов больших объектов в списке Пути большого объекта, нажав кнопку **Добавить**. Поиск файлов больших объектов, заданные в столбце большого объекта во входном файле, будут производиться по этим путям (в том порядке, в котором они заданы в списке **Пути больших объектов**).
3. Нажмите кнопку **OK**, чтобы принять значения по умолчанию на прочих страницах записной книжки и начать процесс импорта.

Укажите опции импорта столбцов. Опции импорта столбцов задаются на странице **Столбцы** записной книжки **Импорт**:

1. Выберите одну из радиокнопок в окне **Включать столбцы по**, чтобы указать метод импорта в таблицу столбцов файла данных. В зависимости от типа файла и режима, выбранных на странице Файл, доступны разные методы.
2. Необязательно: Укажите или измените атрибуты столбцов файла импорта, нажав кнопку **Изменить**.

   Эта опция недоступна, если вы выбрали радиокнопку **По умолчанию (метод D)**.

Управление хранением

Вам, как администратору базы данных, надо оценивать размеры таблиц и индексов, проверять наличие свободного места в табличном пространстве и добавлять к существующему табличному пространству по мере его заполнения дополнительное пространство.

В этом разделе описывается, как:

- Оценивать размер таблиц и индексов
- Определять объем свободного места в табличном пространстве
- Добавлять место в существующее табличное пространство по мере его заполнения

Оценка размера таблиц и индексов

Оценить размер пространства хранения, требуемый для новых или существующих таблиц или индексов можно, вызвав диалоговое окно Оценить размер. Это окно открывается, если выбрать нужные таблицы и индексы и щелкнуть по ним правой кнопкой мыши, или же выбрать **Оценить размер** из окон Создать таблицу или Создать индекс. Размер оценивается на основе определения данной таблицы и зависящих от нее индексов. Оценка - это предполагаемый объемом пространства хранения, который будет занят, если в таблице будет заданное количество строк. На основе наименьшего и наибольшего размеров полей переменной длины вычисляется также минимальное и максимальное пространство. При вызове для таблицы или индекса диалоговое окно **Оценить размер** заранее заполняется спецификациями таблицы и содержит показатели таблицы и всех зависящих от нее индексов. Если нажать кнопку **Обновить**, оценка размера, минимальный размер и максимальный размер будут обновлены на основе чисел, введенных в полях **Новое общее число строк** и **Новая средняя длина строки**.

Оценка размера таблицы или индекса полезна, когда нужно:

- При создании новой таблицы нужно знать, какого размера делать табличное пространство.
- Создать новую таблицу на основе оценки размера существующей таблицы.
- При нехватке памяти узнать, сколько места в табличном пространстве занимают различные объекты таблиц и индексов.
- Оценить предполагаемый размер таблицы перед загрузкой данных.

Примечание: При использовании Оценить размер в продукте DB2 Universal Database Enterprise-Extended Edition убедитесь, что оценки размера основаны на логическом размере данных в таблице.

Если статистика для таблицы в течение некоторого времени не обновлялась, нажмите кнопку **Запустить статистику**, чтобы обновить статистику для выбранной таблицы. Если выбрать индекс, а затем нажать кнопку **Запустить статистику**, будет запущен сбор статистики для соответствующей таблицы.

Чтобы оценить размер таблицы:

- Откройте окно Оценить размер.
- Выберите в поле **Новое общее число строк** новое значение или оставьте значение по умолчанию.
- Нажмите кнопку **Обновить**, чтобы обновить оценку размера для нового значения.
- Выберите в поле **Новая средняя длина строки** новое значение или оставьте значение по умолчанию.

- Нажмите кнопку **Обновить**, чтобы обновить оценку размера для нового значения.

Определение свободного места в табличном пространстве

Чтобы определить объем свободного места в табличном пространстве DMS:

1. В Центре управления дважды щелкните по значку **Табличные пространства**. На панели содержимого появится список всех табличных пространств.
2. Подробности об объеме свободного места в табличном пространстве находятся в столбцах, озаглавленных **Allocated size** (Распределенный размер), **Size used** (Используемый размер) и **Percentage used** (Процент использованного). Пространство измеряется в страницах по 4 Кбайта.

Порядок вывода столбцов, а также выводимые столбцы можно задать при помощи значка Настроить столбцы в нижней части панели содержимого.

Чтобы выяснить объем свободного пространства в табличном пространстве SMS, используйте возможности, предоставляемые операционной системой для наблюдения за использованием пространства, чтобы убедиться, что в каталоге табличного пространства осталось место.

Добавление места в табличное пространство

Объем табличного пространства DMS - это общий размер контейнеров, выделенных табличному пространству. Если заполнение табличного пространства DMS приближается к его объему (в зависимости от использования табличного пространства; возможно значение порога - 90%), к нему надо добавить дополнительное место. Менеджер баз данных автоматически перераспределит таблицы табличного пространства DMS по всем доступным контейнерам. Во время перебалансировки данные в табличном пространстве остаются доступными.

К заполненному табличному пространству DMS можно добавить новый контейнер:

1. В Центре управления щелкните правой кнопкой мыши по табличному пространству на панели содержимого, для которого нужно добавить контейнер, и выберите **Изменить** во всплывающем меню. Откроется окно Изменить табличное пространство.
2. Нажмите кнопку **Добавить**. Откроется окно Добавить контейнер.
3. Выберите радиокнопку **Файл** или **Непосредственное устройство** и заполните нужные поля. Дополнительную информацию смотрите в электронной справке.
4. Нажмите кнопку **OK**.

Обычно размер табличного пространства SMS увеличить довольно сложно, так как емкость SMS зависит от свободного пространства в файловой системе и максимального размера файла, поддерживаемого операционной системой.Но, в

зависимости от операционной системы, может быть возможным увеличить размер файловой системы средствами операционной системы. Для табличного пространства SMS в системе на основе UNIX можно увеличивать размеры табличных пространств при помощи соответствующих команд операционной системы. Смотрите документацию по вашей системе на основе UNIX. Если в файловой системе, содержащей табличное пространство SMS, содержатся также файлы, не относящиеся к DB2, можно переместить их в другую файловую систему, освободив таким образом в файловой системе дополнительное пространство для DB2. Можно также выполнить восстановление с перераспределением, то есть восстановление табличного пространства в большее количество контейнеров, чем было при создании резервной копии. Восстановление с перераспределением можно выполнить из записной книжки Восстановить базу данных: для базы данных, которую нужно восстановить, выберите из всплывающего меню **Восстановить —> Базу данных**.

Устранение неисправностей

В DB2 входит руководство по устранению неисправностей, предназначенное для представителей службы технической поддержки серверов и клиентов DB2. Это поможет:

- Быстро диагностировать проблемы или ошибки
- Разрешать проблемы на основе их симптомов
- Использовать доступные средства диагностики
- Разрабатывать стратегию устранения неисправностей для ежедневной работы с DB2.

Руководство *Troubleshooting Guide* содержит следующие основные разделы об устранении неисправностей:

- Рекомендуемые способы устранения неисправностей
- Устранение неисправностей на сервере
- Устранение неисправностей на клиенте
- Устранение неисправностей связи с хостом
- Устранение неисправностей программ
- Устранение неисправностей и диагностика проблем.

В руководстве *Troubleshooting Guide* предлагаются следующие дополнительные темы по устранении неисправностей:

- Модель процесса DB2
- Использование информации журналов
- Выполнение трассировки
- Средства диагностики для операционных систем на основе UNIX, OS/2 и Microsoft Windows.

В WWW свежие новости и техническая документация доступны по адресу http://www.software.ibm.com/data/db2/library/.

Подробная информация о том, как связаться с IBM, приводится в последнем разделе этой книги.

Репликация данных

Репликация - это процесс применения изменений, сохраненных в журнале базы данных на сервере источника, к серверу назначения. Репликацию можно использовать для определения, синхронизации и управления операциями копирования данных внутри вашего предприятия. Можно автоматически передавать данные с системы хоста на узлы назначения. Например, можно копировать данные и программы для филиалов, розничных продавцов и даже на портативные компьютеры торговых представителей.

В репликации используются два компонента: Capture и Apply. Компонент Capture, читая журнал базы данных, захватывает изменения данных в таблицах источника, определенных для репликации. Компонент Apply читает измененные данные (ранее захваченные и сохраненные в таблице изменений данных) и применяет их к таблицам назначения.

Для конфигурирования репликации при помощи Центра управления используйте действия **Определить как источник репликации** и **Определить регистрацию**. Компоненты репликации Capture и Apply выполняются вне инструментов управления DB2.

Из Центра управления администраторы репликации могут выполнять следующие действия:

- Определять источники репликаций
- Определять регистрации репликаций
- Задавать SQL для изменения данных в процессе применения

Ниже приведена последовательность основных действий для репликации данных. Дополнительную информацию смотрите в руководстве *Replication Guide and Reference*.

1. Создайте сценарий репликации (задайте отображение таблиц источника на таблицы назначения).
2. Определите источник репликации (это связано с действием Capture).

Чтобы определить источник репликации:

1. Задайте захватываемые исходные столбцы.
2. Выберите опции репликации.
3. Определите регистрацию репликации (это связано с действием Apply).

4. Измените исходную таблицу для захвата изменений данных.
5. Запустите программу Capture, чтобы прочесть и сохранить изменения данных.
6. Запустите программу Apply, чтобы реплицировать изменения в таблицы назначения.

Чтобы определить регистрацию репликации:

1. Задайте имя набора регистрации.
2. Задайте базу данных и таблицу назначения.
3. Задайте столбцы назначения.
4. Задайте выбор строк.
5. Задайте оператор SQL для обработки во время выполнения.
6. Задайте временные характеристики регистрации.

Использование протокола LDAP (Lightweight Directory Access Protocol)

Для добавления и удаления записей на сервере LDAP можно использовать Ассистент конфигурирования клиента. Все экземпляры баз данных, зарегистрированные на сервере LDAP, будут автоматически внесены в каталог на клиенте. Они будут видны в Центре управления в виде обычных узлов дерева. Этими базами данных можно управлять так же, как и другими базами данных, внесенными в каталог на вашем компьютере (исключение: в данной версии не реализованы опции добавления базы данных и отбрасывания базы данных).

Для управления базой данных LDAP выберите эту базу данных и щелкните правой кнопкой мыши. Появится всплывающее меню с функциями, которые можно выполнить. Дополнительную информацию о LDAP смотрите в разделе “Приложение J. Службы каталогов протокола LDAP” на стр. 421.

Использование Центра управления Java

Центр управления может выполняться или как программа Java, или как апплет Java через сервер Web. В обоих случаях для работы с Центром управления на компьютере должна быть установлена виртуальная Java-машина (JVM). Для выполнения Центра управления в виде программы Java должна быть установлена также правильная среда Java Runtime Environment (JRE). Это может быть среда Java Runtime Environment (JRE) для запуска программ или браузер с поддержкой Java для запуска апплетов.

Если на компьютере установлена правильная среда JRE, **прикладные программы** Java выполняются так же, как и другие прикладные программы.

Выполнение Центра управления в виде апплета Java

Апплеты Java - это программы, которые выполняются внутри браузера Web, поддерживающего Java. Код апплета Центра управления может располагаться

на удаленном компьютере; в этом случае сервер Web передает код апплета браузеру клиента. При выполнении Центра управления как апплета Java необходимо использовать поддерживаемый браузер с поддержкой Java, работающий в 32-битной операционной системе Windows или в операционной системе OS/2. На данный момент браузеры для систем на основе UNIX не поддерживаются.

Сервер апплетов JDBC Центра управления должен быть запущен под ID пользователя с полномочиями администратора на компьютере, находится Сервер апплетов. Можно задать автоматический запуск Сервера апплетов JDBC Центра управления при загрузке.

Для выполнения Центра управления в виде апплета Java на компьютере, на котором расположены код апплета Центра управления и сервер апплетов JDBC Центра управления, должен быть установлен сервер Web. Сервер Web должен разрешать доступ к каталогу `sqllib`.Если нужно использовать виртуальный каталог, вместо этого каталога задавайте начальный каталог. Например, если виртуальный каталог называется `temp`, нужно использовать `sqllib/temp`. DB2 не поддерживает установку Центра управления на диске FAT системы OS/2, поскольку на таких дисках не поддерживаются длинные имена файлов, необходимые для Java. Дополнительную информацию об установке и конфигурировании Центра управления как программы Java или апплета Java смотрите в руководстве *Быстрый старт* для вашей платформы.

Использование инструментов Java для управления

DB2 включает набор интерфейсов Java, позволяющих расширить возможности Центра управления. Интерфейсы Java позволяют:

- Добавлять в список меню при работе с объектами дополнительные пункты.
- Добавлять кнопки в панель инструментов Центра управления.

Для использования этой возможности у вас должно быть установлено программное обеспечение Java нужного уровня. Дополнительную информацию об использовании этой функции смотрите в справочнике “Приложение K. Расширение Центра управления” на стр. 453.

Часть 2. Реализация вашего проекта

Глава 2. Перед созданием базы данных

После определения структуры базы данных необходимо создать эту базу данных и объекты в ней. Эти объекты - схемы, групп узлов, табличные пространства, таблицы, производные таблицы, оболочки, службы, псевдонимы, отображения типов, отображения функций, алиасы, пользовательские типы (UDT), пользовательские функции (UDF), триггеры, ограничения, индексы и пакеты. Эти объекты можно создать при помощи операторов SQL в процессоре командной строки, в Центре управления DB2 или через API в прикладных программах.

Информацию об операторах SQL смотрите в руководстве *SQL Reference*. Информацию о командах процессора командной строки смотрите в руководстве *Command Reference*. Информацию об API смотрите в руководстве *Administrative API Reference*.

Примечание: Для вашей платформы может существовать пользовательский интерфейс, позволяющий создавать объекты баз данных. Этот интерфейс можно использовать вместо операторов SQL, команд процессора командной строки или API. Чтобы узнать, можете ли вы воспользоваться таким интерфейсом, посмотрите руководство *Быстрый старт* для вашей платформы.

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. Описание конкретных отличий смотрите в соответствующем руководстве *Быстрый старт* и в документации на операционную систему.

Например, Windows NT** поддерживает тип прикладных программ, называемых “службами”. В DB2 for Windows NT можно использовать экземпляр DB2, определенный как служба. Служба может автоматически запускаться при загрузке системы (этот режим может быть задан пользователем с помощью апплета панели управления Службы или прикладной программой Win32, использующей функции служб из интерфейса прикладного программирования (API) Microsoft** Win32**). Службы могут работать, даже когда в системе не зарегистрирован ни один пользователь.

Что требуется перед созданием базы данных

Перед созданием базы данных нужно рассмотреть следующие темы:

Запуск DB2

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.

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. В операционных системах UNIX выполните следующий сценарий запуска:

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

где INSTHOME - начальный каталог нужного экземпляра.

3. Для запуска экземпляра используйте один из следующих двух методов:
   a. Чтобы запустить экземпляр с помощью Центра управления:

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

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

```
db2start
```

Примечание: Команда **db2start** запускает экземпляр в соответствии с правилами, изложенными в разделе “Задание текущего экземпляра” на стр. 71.

Запуск DB2 UDB в Windows NT

Команда db2start запускает DB2 в качестве службы NT. Чтобы запустить DB2 в Windows NT как процесс, нужно задать в команде DB2START ключ ″/D″. Запустить DB2 как службу можно также при помощи Панели управления или команды ″NET START″.

Чтобы успешно запустить DB2 как службу командой DB2START, пользователь должен обладать в операционной системе Windows NT соответствующей привилегией для запуска служб NT. Этот пользователь может быть членом группы администраторов, операторов сервера или привилегированных пользователей.

При работе в среде многораздельных баз данных каждый сервер раздела базы данных запускается как служба NT.

Использование нескольких 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.

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* 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.

Информацию о типе соединения, необходимом для выполнения каждой команды, смотрите в руководстве *Command Reference*.

Поддержка системой DB2 нескольких экземпляров зависит от операционной системы. Информацию об определении нескольких экземпляров DB2 на одном компьютере смотрите в руководстве *Быстрый старт* для вашей операционной системы.

Для подключения к другому экземпляру (возможно, удаленному) используйте команду ATTACH, как описано в руководстве *Command Reference*.

При помощи 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* от этого экземпляра, введя следующую команду:

```
db2 detach
```

Организация и группировка объектов в схемы

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. Дополнительную информацию смотрите в справочнике *SQL Reference*.

Как описано в разделе “Определение таблиц системного каталога” на стр. 108, при создании базы данных создаются некоторые объекты в определенных схемах.

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. Дополнительную информацию о явном создании схем смотрите в разделе “Создание схемы” на стр. 121.

Разрешение параллелизма

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).

Разрешение внутриразделього параллелизма

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 внутриразделный параллелизм влияют параметры конфигурации менеджера баз данных *max_querydegree* и *intra_parallel* и параметр конфигурации базы данных *dft_degree*. Дополнительную информацию о параметрах конфигурации смотрите в руководстве *Administration Guide: Performance*.

Разрешение внутриразделього параллелизма запросов

Чтобы разрешить внутриразделный параллелизм запросов, необходимо изменить параметры конфигурации базы данных и параметры конфигурации менеджера баз данных.

INTRA_PARALLEL
: Параметр конфигурации менеджера баз данных. Дополнительную информацию об этом параметре смотрите в руководстве *Administration Guide: Performance*.

DFT_DEGREE
: Параметр конфигурации базы данных. Задает значения по умолчанию для опции связывания DEGREE и специального регистра CURRENT DEGREE. Дополнительную информацию об этом параметре смотрите в руководстве *Administration Guide: Performance*.

DEGREE
: Опция прекомпиляции или связывания для статического SQL. Дополнительную информацию смотрите в руководстве *Command Reference*.

CURRENT DEGREE
: Специальный регистр для динамического SQL. Дополнительную информацию смотрите в руководстве *SQL Reference*.

Дополнительную информацию о параметрах конфигурации и о том, как разрешить выполнение прикладных программ в параллельном режиме, смотрите в главе "Конфигурирование DB2" руководства *Administration Guide: Performance*.

Разрешение межраздельного параллелизма запросов

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

Разрешение параллелизма утилит

В этом разделе описано, как разрешить внутираздельный 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.

Загрузка: Утилита загрузки использует параллелизм автоматически; можно также задать следующие параметры команды LOAD:

- CPU_PARALLELISM
- DISK_PARALLELISM

Информацию о команде LOAD смотрите в руководстве *Data Movement Utilities Guide and Reference*.

Автозагрузка: Можно разрешить несколько процессов разбиения для автозагрузки, задав параметр MODIFIED BY ANYORDER для команды LOAD в файле `autoloader.cfg`. Дополнительную информацию смотрите в руководстве *Data Movement Utilities Guide and Reference*.

Создание индекса: 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.

Информацию об операторе CREATE INDEX смотрите в справочнике *SQL Reference*.

Разрешение разделения данных

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). Информацию об этом смотрите в руководствах *Command Reference* и *Administrative API Reference*.

Перед созданием многораздельной базы данных нужно определить, будете вы подключаться как локальный или удаленный клиент для экземпляра, в котором нужно создать эту базу данных. Затем необходимо подключиться к этому экземпляру. Нужно также выбрать раздел базы данных, который будет узлом каталога для этой базы данных. 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. 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.

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. Значения отдельных записей в файле конфигурации конкретной базы данных или менеджера баз данных можно узнать при помощи команд 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*.

Дополнительную информацию о параметрах конфигурации смотрите в руководстве *Administration Guide: Performance*.

Резервное копирование базы данных или табличного пространства

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

- Используйте несколько носителей назначения.
- Сконфигурируйте табличные пространства для параллельных операций ввода-вывода.
- При помощи параметра PARALLELISM команды BACKUP задайте степень параллелизма.
- При помощи 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 доступного 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.

Информацию о команде BACKUP DATABASE смотрите в руководстве *Command Reference*.

Восстановление 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.
- При помощи параметра PARALLELISM команды RESTORE задайте степень параллелизма.

- При помощи 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 доступного 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.

Информацию о команде RESTORE DATABASE смотрите в руководстве *Command Reference*.

Остановка DB2

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.

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.
4. В операционных системах UNIX выполните следующий сценарий запуска:

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

где INSTHOME - начальный каталог нужного экземпляра.

5. Для остановки экземпляра используйте один из следующих методов:

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**.

Чтобы остановить экземпляр из командной строки, введите команду:

```
db2stop
```

Подробности создания базы данных

Перед созданием базы данных нужно рассмотреть или выполнить следующие задачи:

- Разработка логических или физических характеристик базы данных
- Создание экземпляра
- Задание переменных среды и реестра профиля
- Создание сервера администратора DB2 (DAS)
- Создание файла конфигурации узлов
- Создание файла конфигурации базы данных
- Использование файлов ответов для копирования конфигурации
- Включение 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 *Administration Guide: Planning*.

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 диагностики DB2 (`db2diag.log`)
- 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 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.

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.

Автоматическое задание параметров среды DB2

Примечание: Эта информация относится только к операционным системам UNIX.

По умолчанию сценарии 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`.

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.

Задание параметров среды DB2 вручную

Примечание: Эта информация относится только к операционным системам UNIX.

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.

Несколько экземпляров в одной системе

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

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.

Добавление экземпляра

Если у вас есть полномочия администратора в OS/2 или полномочия root в системах 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). Экземпляры надо добавлять на компьютере, на котором находится сервер администратора.

Чтобы добавить еще один экземпляр, выполните следующие шаги:

1. Зарегистрируйте 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**.

   Чтобы добавить экземпляр из командной строки, введите команду:

   ```
   db2icrt <имя_экземпляра>
   ```

3. Создайте сервер администратора.

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. Дополнительную информацию об аутентификации смотрите в разделе “Глава 5. Управление доступом к базам данных” на стр. 235.

Примечание: Для изменения конфигурации экземпляра можно использовать команду **db2iupdt**.

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**.

Подробности добавления экземпляров DB2 Enterprise - Extended Edition: При работе с DB2 Universal Database Enterprise - Extended Edition нужно также объявить, что добавляемый новый экземпляр является системой многораздельных баз данных. Для этого введите в командной строке команду `-s eee`.

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, CLIENT, DCS и DCE. 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. Допустимые типы экземпляров: ee, eee и client.

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.

Подробности создания экземпляров в Windows NT: При работе в операционной системе Windows NT у команды **db2icrt** можно задать следующие дополнительные параметры:

- –s тип_установки

  Позволяет создавать экземпляры разных типов. Допустимые типы экземпляров: ee, eee и client.

- /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 регистрационное имя, учетную запись и 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 NT Enterprise - Extended Edition:

```
db2icrt inst1 -s eee
   /p:\\machineA\db2mpp
   /u:ваше_имя,ваш_пароль /r:9010,9015
```

Примечание: Команда **db2icrt** предоставляет имени пользователя, использованному для создания экземпляра, следующие права:

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

Эти права требуются экземпляру для доступа к совместно используемому диску, аутентификации учетной записи пользователя и запуска DB2 в виде службы Windows NT.

Список экземпляров

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

1. Раскройте дерево объектов и найдите папку **Экземпляры** этой системы.
2. Щелкните правой кнопкой мыши по папке Экземпляры и выберите из всплывающего меню пункт **Добавить**.
3. В окне Добавить базу данных 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, введите команду:

```
set db2instance
```

Примечание: В операционных системах UNIX введите команду:

```
db2 get instance
```

Задание текущего экземпляра

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 95 они задаются в файле `autoexec.bat`. В OS/2 они задаются в файле `config.sys`.
- 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
  ```

Автоматический запуск экземпляров

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

```
db2iauto -on имя_экземпляра
```

где `имя_экземпляра` - регистрационное имя этого экземпляра.

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

```
db2iauto -off имя_экземпляра
```

где `имя_экземпляра` - регистрационное имя этого экземпляра.

Одновременное выполнение нескольких экземпляров

Можно запустить несколько экземпляров DB2, если они относятся к одной версии продукта.

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.

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.

Задание переменных среды и реестра профиля

Переменные среды и переменные реестра определяют среду базы данных.

Если перед началом работы с реестром профиля DB2 на рабочей станции (например, с системой Windows или OS/2) 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 с правами системного администратора (SYSADM) для данного экземпляра могут изменять значения реестра для этого экземпляра. 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: В некоторых операционных системах переменные среды DB2 `DB2INSTANCE`, `DB2NODE`, `DB2PATH` и `DB2INSTPROF` могут не сохраняться в реестрах профилей DB2. Для изменения значений этих переменных нужно использовать 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. Многие переменные среды и переменные реестра перечислены в разделе “Переменные реестра и среды DB2” руководства *Administration Guide: Performance*. Различные уровни поддержки обеспечиваются в настоящее время разными профилями среды. 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 EEE на каждом компьютере существует свой профиль глобального уровня.
- Реестр профиля DB2 уровня узла экземпляра. В системе, в которой база данных распределена по нескольким разделам базы данных, этот реестр располагается на каждом узле (то есть компьютере) и содержит значения переменных среды для всех экземпляров, хранящих данные на этом узле. Значения, определенные на этом уровне, переопределяют соответствующие значения на уровне экземпляра и глобальном уровне.
- Реестр профиля экземпляра DB2. Этот реестр содержит список всех имен экземпляров, распознаваемых системой.

Пользователи могут переопределить значения переменных среды реестра профиля экземпляра DB2 для своих сеансов, изменив значения переменных среды сеанса при помощи команды **set** (или команды export в системе UNIX).

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. Значения реестра, заданные в профиле уровня узла экземпляра (при помощи команды `db2set -i` с заданным номером узла, как показано ниже).
3. Значения реестра, заданные в профиле уровня экземпляра (при помощи команды `db2set -i`, как показано ниже).
4. Значения реестра, заданные в профиле глобального уровня (при помощи команды `db2set -g`, как показано ниже).

Использование команды db2set

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 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).

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 всех экземпляров в системе, используйте команду:

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

Для задания переменных реестра на уровне пользователя используйте команду:

```
db2set -ul
```

Для задания переменных реестра на уровне пользователя для определенного пользователя используйте команду:

```
db2set -ul имя_пользователя
```

Примечания:

1. Параметры ″-i″, ″-g″ и ″-ul″ нельзя использовать в одной команде.
2. Некоторые переменные всегда определяются в профиле глобального уровня. Они не могут быть заданы в профилях на уровне узла или экземпляра (например, переменные `db2system` и `db2instdef`).
3. 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.

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.

Чтобы задать эту переменную на глобальном уровне LDAP, используйте для команды `db2set` опцию `-gl`.

Примечание: Эта опция отличается от опции `-g`, используемой для задания переменных реестра DB2 на глобальном уровне компьютера. Опция `-gl` задает глобальный уровень LDAP. Задание переменной реестра DB2 в LDAP поддерживается только на платформах Windows.

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

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

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

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

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

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
```

Задание переменных среды в OS/2

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 системе OS/2 не задавайте переменные среды (кроме DB2PATH и DB2INSTPROF) в файле config.sys. Все переменные, за исключением настоящих переменных среды, следует задавать в реестрах профилей при помощи команды **db2set**.

Переменная DB2INSTANCE - это тоже настоящая переменная среды, но если используется переменная реестра DB2INSTDEF, она не требуется. Переменная реестра DB2INSTDEF определяет имя экземпляра по умолчанию, используемое, если не задано значение DB2INSTANCE.

Значения DB2INSTANCE и DB2PATH задаются при установке DB2; значение DB2INSTPROF можно задать после установки. Переменная среды DB2PATH должна быть обязательно задана; она задается во время установки и изменять ее нельзя. Переменные среды DB2INSTANCE и DB2INSTPROF задавать не обязательно.

Чтобы узнать значение переменной среды, введите команду:

```
set переменная
```

Чтобы изменить значение переменной среды, введите команду:

```
set переменная=значение
```

Чтобы задать переменные среды системы, измените файл config.sys и перезагрузите систему, чтобы внесенные изменения вступили в силу.

Положения различных реестров профилей:

- Файл реестра профиля DB2 уровня экземпляра:

  ```
  %DB2INSTPROF%\имя_экземпляра\PROFILE.ENV
  ```

- Реестр профиля DB2 глобального уровня:

  ```
  %DB2INSTPROF%\DEFAULT.ENV
  ```

- Реестр профиля экземпляров DB2:

  ```
  %DB2INSTPROF%\PROFILES.REG
  ```

Задание переменных среды в Windows NT и Windows 95

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 32-битных 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 - Extended Edition в Windows NT 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 95 и Windows 98: Отредактируйте файл *autoexec.bat* и перезагрузите систему, чтобы внесенные изменения вступили в силу.

В системе Windows NT 4.x: Переменные среды DB2 `DB2INSTANCE`, `DB2PATH` и `DB2INSTPROF` можно задать так:

- Выберите **Пуск**, **Настройка**, **Панель управления**.
- Дважды щелкните по значку **Система**.
- На Панели управления системы в разделе Переменные среды системы сделайте следующее:
  1. Если переменная DB2INSTANCE не существует:
     a. Выберите любую переменную среды системы.
     b. Измените имя в поле *Переменная* на DB2INSTANCE.
     c. Измените значение в поле *Значение* на 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. Нажмите кнопку Задать.
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
```

Реестры профилей располагаются в следующих местах:

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

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

  Примечание: *имя_экземпляра* - конкретное значение для используемого раздела базы данных.

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

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

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

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

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

DB2 UDB позволяет обращаться к переменным реестра DB2 UDB уровня экземпляра с удаленного компьютера. В настоящее время переменные реестра DB2 UDB хранятся на трех различных уровнях: уровень компьютера или глобальный уровень, уровень экземпляра и уровень узла. Переменные реестра, сохраненные на уровне экземпляра (включая уровень узла) можно перенаправить на другой компьютер, используя переменную DB2REMOTEPREG. Если задана переменная DB2REMOTEPREG, DB2 UDB будет обращаться к переменным среды DB2 UDB на компьютере, указанном в DB2REMOTEPREG. Например,

```
db2set DB2REMOTEPREG=rmtwkstn
```

где *rmtwkstn* - 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.

Задание переменных среды в системах UNIX

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 **db2profile** и **db2cshrc**.

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 оболочек Bourne и C:

  ```
  set db2instance inst1
  ```

Для правильного управления реестром профиля DB2 в операционных системах UNIX необходимо следовать следующим правилам принадлежности файлов. (Информацию о сервере администратора DB2 (DAS) смотрите в разделе “Создание сервера администратора DB2 (DAS)” на стр. 82.)

- Файл реестра профиля DB2 уровня экземпляра:

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

  Разрешения на доступ и владелец для этого файла должны быть:

  ```
  -rw-r--r-- Владелец_Экземпляра Группа_Экземпляра_DAS profile.env
  ```

  INSTHOME - начальный каталог владельца экземпляра.

- Реестр профиля DB2 глобального уровня:
  - */var/db2/<ID_версии>/default.env* для операционных систем AIX, Solaris, SINIX и NUMA-Q(Sequent) (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-r--r-- Владелец_Экземпляра_DAS Группа_Экземпляра_DAS default.env
  ```

  Чтобы изменить переменные реестра глобального уровня, пользователь должен зарегистрироваться как root или как владелец экземпляра DAS. Дополнительную информацию о сервере администратора DB2 смотрите в разделе “Создание сервера администратора DB2 (DAS)” на стр. 82.

- Реестр профиля DB2 уровня узла экземпляра:

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

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

  ```
  drwxrwxr-x Владелец_Экземпляра Группа_Экземпляра_DAS nodes

  -rw-r--r-- Владелец_Экземпляра Группа_Экземпляра_DAS номер_узла.env
  ```

  Примечание: *Владелец_Экземпляра* и *Владелец_Экземпляра_DAS* должны быть членами *Группа_Экземпляра_DAS*.

  INSTHOME - начальный каталог владельца экземпляра.

- Реестр профиля экземпляров DB2:
  - `/var/db2/<ID_версии>/profiles.reg` для операционных систем AIX, Solaris, SINIX и NUMA-Q(Sequent) (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
```

Создание сервера администратора DB2 (DAS)

Сервер администратора DB2 (DAS) - это специальная точка управления DB2, используемая только для задач управления другими серверами DB2. Сервер администратора DB2 должен быть запущен для использования Ассистента конфигурирования клиента или Центра управления. Сервер администратора DB2 работает вместе с Центром управления и Ассистентом конфигурирования клиента при выполнении следующие задачи 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. Центр управления используется для определения 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 Ассистентом конфигурирования клиента и Центром управления для упрощения и автоматизации процесса конфигурирования соединений клиента с базами данных DB2.

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

DAS выполняет на удаленной системе хоста запросы клиентов, полученные от Центра управления и Ассистента конфигурирования клиента. Для обращения к DAS клиент должен обладать полномочиями SYSADM. Все эти клиенты могут быть заданы в параметре конфигурации SYSADM_GROUP.

Для выполнения некоторых запрошенных операций могут требоваться особые полномочия. DAS выполняется под идентификатором конкретного пользователя. Этому пользователю нужно предоставить минимальный набор привилегий, необходимых для выполнения задач или операций, выполняемых администратором. Как правило эти требуемые задачи или операции включают:

- Запрос информации о конфигурации операционной системы (OS).
- Запрос информации о пользователях и группах OS.
- Запуск и остановка других экземпляров DB2.

- Выполнение заданий, внесенных в расписание.
- Сбор информации для конфигурации связи и протоколов.

Дополнительную информацию о настройке связи DAS смотрите в руководстве *Быстрый старт* для данной платформы.

Создание DAS

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 OS/2 или Windows NT:

  Зарегистрируйтесь на компьютере, на котором нужно создать DAS, с именем пользователя, обладающим полномочиями локального администратора. Если нужно задать особого пользователя, создайте пользователя с полномочиями локального администратора. Введите команду `db2admin create`. (Чтобы задать конкретные имя пользователя и пароль, для команды `db2admin create` нужно задать опции “/USER:” и “/PASSWORD:”.)

  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 имени 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**. Дополнительную информацию об этой команде смотрите в руководстве *Command Reference*.

  Если для автоматизации конфигурирования соединений с сервером DB2 в DB2 UDB for Windows NT Enterprise - Extended Edition используется Ассистент конфигурирования клиента или Центр управления, сервер раздела базы данных, находящийся на том же компьютере, что и сервер администратора, будет узлом координатора. Это означает, что все соединения от клиентов к базе данных будут направлены к серверу раздела базы данных на компьютере - владельце экземпляра, и только потом они будут перенаправляться к другим серверам разделов базы данных.

  Создание в DB2 UDB for Windows NT Enterprise - Extended Edition дополнительных серверов администратора на других компьютерах позволяет Ассистенту конфигурирования клиента или Центру управления сконфигурировать другие системы в качестве узлов координаторов при помощи программы поиска. Для этого:

  1. Зарегистрируйтесь на компьютере с именем пользователя, обладающим полномочиями локального администратора.

2. Создайте учетную запись пользователя Windows NT с полномочиями локального администратора, которую будет использовать сервер администратора. Имя пользователя этой учетной записи должно удовлетворять правилам именования DB2. При создании учетной записи для сервера администратора учтите следующее:
   – Не используйте эту учетную запись в качестве учетной записи пользователя.
   – Для пароля этой учетной записи задайте опцию `Пароль с неограниченным сроком действия`.
3. Выполните команду:

   ```
   db2admin create /user:имя_пользователя
      /password:пароль
   ```

   где `имя_пользователя` и `пароль` - это имя пользователя и пароль для сервера администратора.

- На платформах UNIX:
  1. Вы должны обладать полномочиями пользователя root.
  2. Находясь в подкаталоге `instance` каталога экземпляра DB2 Universal Database, введите в командной строке команду:

     ```
     dasicrt ASName
     ```

     – В AIX:

       ```
       /usr/lpp/db2_nn_00&/экземпляр/
         dasicrt ASname
       ```

     – В HP-UX, NUMA-Q(Sequent) или Solaris:

       ```
       /opt/IBMdb2/<ID_версии>/экземпляр/
         dasicrt ASname
       ```

     – В Linux:

       ```
       /usr/IBMdb2/<ID_версии>/экземпляр/
         dasicrt ASname
       ```

     где *ASName* - имя экземпляра этого сервера администратора, а db2_nn_00& или <ID_версии> - идентификатор текущей версии.

Примечание: При использовании информационных служб сети (NIS) или NIS+ нужно задать имена групп и пользователей так, чтобы:
   – Первичная группа сервера администратора была в списках вторичных групп всех экземпляров.
   – Список вторичных групп сервера администратора содержал все первичные группы экземпляров.

   Списки вторичных групп изменяются автоматически, только когда в системе не запущены NIS и NIS+.

Поскольку ID пользователя может владеть только одним экземпляром, необходимо создать отдельный ID пользователя для каждого создаваемого сервера администратора DB2.

Создав сервер администратора, используйте его для задания структуры каталогов и прав доступа.

Запуск и остановка сервера администратора

Чтобы вручную запустить или остановить сервера администратора, необходимо сначала зарегистрироваться на компьютере с учетным именем или ID пользователя, обладающим полномочиями локального администратора.

При работе с DB2 for OS/2 или DB2 for Windows NT необходимо сделать следующее:

- Чтобы запустить DAS, введите команду `db2admin start`
- Чтобы остановить DAS, введите команду `db2admin stop`

Примечание: Для выполнения обеих этих команд в Windows NT пользователь должен обладать полномочиями SYSADM, SYSCTRL или SYSMAINT.

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:

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

     где `INSTHOME` - начальный каталог экземпляра.
  3. Для запуска сервера администратора используйте команду **db2admin**:

     ```
     db2admin start
     ```

     Примечание: Сервера администратора автоматически запускается после каждой перезагрузки системы.
- Чтобы остановить сервер администратора:
  1. Зарегистрируйтесь как владелец сервера администратора.
  2. Запустите сценарий запуска одной из следующих команд:

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

     где `INSTHOME` - начальный каталог экземпляра.
  3. Остановите сервер администратора командой **db2admin**:

```
db2admin stop
```

Примечание: В обоих случаях для выполнения этих команд в UNIX необходимо зарегистрироваться с ID авторизации владельца сервера администратора.

Вывод имени сервера администратора

Чтобы узнать имя экземпляра сервера администратора на данном компьютере, введите команду:

```
db2admin
```

Эта команда используется также для запуска или остановки сервера администратора, создания новых пользователя и пароля, отбрасывания экземпляра сервера администратора и задания или изменения учетной записи пользователя, связанной с этим экземпляром сервера администратора.

Конфигурирование сервера администратора

Чтобы узнать текущие значения параметров конфигурации, относящихся к серверу администратора, введите команду:

```
db2 get admin cfg
```

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

Чтобы изменить отдельные относящиеся к серверу администратора записи в файле конфигурации менеджера баз данных, введите команду:

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

Дополнительную информацию о изменяемых параметрах конфигурации менеджера баз данных смотрите в руководстве *Command Reference*.

Чтобы вернуть значения параметров конфигурации к рекомендованным для менеджера баз данных значениям по умолчанию, введите команду:

```
db2 reset admin cfg
```

Изменения файла конфигурации менеджера баз данных вступают в силу только после загрузки их в память (для этого нужно ввести команду `db2admin stop` и затем `db2admin start`, а в Windows NT - остановить и запустить службу.)

Информацию о настройке протоколов связи для сервера администратора смотрите в руководстве *Быстрый старт* для данной платформы.

Особенности защиты для сервера администратора

Сначала необходимо зарегистрироваться на компьютере с учетным именем или ID пользователя, обладающим полномочиями локального администратора.

Примечание: В Windows NT для изменения учетной записи регистрации для DAS не следует использовать утилиту **Службы** на **панели управления**, так как при этом для этой учетной записи регистрации не будут заданы некоторые необходимые права доступа. Для задания или изменения учетной записи регистрации для сервера администратора DB2 всегда используйте команду **db2admin**.

После создания сервера администратора задать или изменить учетную запись регистрации можно при помощи команды **db2admin**:

```
db2admin setid имя_пользователя пароль
```

где *имя_пользователя* и *пароль* - это имя пользователя и пароль учетной записи с полномочиями локального администратора.

Рекомендуется, чтобы этот ID или имя пользователя обладали полномочиями SYSADM на каждом сервере системы - это позволит при необходимости запускать или останавливать другие экземпляры.

Изменение сервера администратора

В операционных системах UNIX при изменении DB2 с использованием временного исправления программы (PTF) или исправления кода должны быть обновлены все серверы администраторов DB2 и все существующие экземпляры. Чтобы исправить сервер администратора, в подкаталоге `instance` подкаталога конкретной установленной версии и выпуска DB2 выполните команду **dasiupdt**.

Сначала необходимо зарегистрироваться на компьютере в качестве пользователя “root” (в UNIX) или с ID пользователя, обладающим полномочиями локального администратора.

Эта команда используется так:

```
dasiupdt InstName
```

`InstName` - имя регистрации владельца экземпляра. У этой команды есть также необязательные параметры, которые помещаются перед `InstName` и отделяются пробелами:

- –h или –?

  Выводит для этой команды меню справки.

- –d

  Задает режим отладки, используемый для анализа ошибок.

Удаление сервера администратора

Сначала необходимо зарегистрироваться на компьютере в качестве пользователя “root” (в UNIX) или с ID пользователя, обладающим полномочиями локального администратора.

Чтобы удалить сервер администратора:

- В операционных системах OS/2 или Windows NT:
  1. Остановите сервер администратора, используя команду `db2admin stop`.
  2. Если нужно, сделайте резервную копию всех файлов из подкаталога `db2das00` подкаталога `sqllib`. Каталог экземпляра указывается переменной реестра *DB2INSTPROF*.

     Примечание: В этом примере подразумевается, что имя удаляемого сервера администратора - `db2das00`.
  3. Отбросьте сервер администратора, используя команду `db2admin drop`.

  Примечание: Для выполнения этой команды в Windows NT пользователь должен обладать полномочиями SYSADM, SYSCTRL или SYSMAINT.
- В операционных системах UNIX:
  1. Зарегистрируйтесь как владелец сервера администратора.
  2. Запустите сценарий запуска, используя одну из следующих команд:

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

     где `INSTHOME` - начальный каталог экземпляра.
  3. Остановите сервер администратора командой **db2admin**:

     ```
     db2admin stop
     ```
  4. Если нужно, сделайте резервную копию всех файлов из подкаталога `sqllib` начального каталога этого сервера администратора. Каталог экземпляра указывается переменной реестра *DB2INSTPROF*.
  5. Выйдите из системы.
  6. Зарегистрируйтесь как пользователь `root` и удалите сервер администратора, используя команду **dasidrop**:

     ```
     dasidrop ASName
     ```

     где `ASName` - имя экземпляра этого сервера администратора. Эта команда находится в подкаталоге `instance` подкаталога для конкретной установленной версии и выпуска DB2.

     Примечание: Команда **dasidrop** удаляет каталог `sqllib` из начального каталога сервера администратора DB2.

Настройка сервера администратора для работы с системами EEE

Ниже описаны шаги, необходимые для настройки серверов DB2 EEE (Solaris, NT, Sequent, HP-UX и AIX) для удаленного управления с помощью Центра управления.

Во время установки программа установки создает один сервер администратора на компьютере - владельце экземпляра. Можно создать дополнительные серверы администратора на других компьютерах, чтобы позволить Центру управления или Ассистенту конфигурирования клиента обращаться к другим узлам координатора. Затраты ресурсов на выполнение функций узла координатора могут быть распределены между несколькими узлами экземпляра.

Чтобы распределить функции координатора:

1. Создайте новые серверы администратора на выбранных дополнительных компьютерах системы многораздельных баз данных.
2. В Центре управления или Ассистенте конфигурирования клиента внесите 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 EEE V7.1

путь установки:
путь_установки

файл служб TCP:
файл_служб_tcp

Экземпляр DB2:

имя: db2inst

ID владельца:
db2inst

путь экземпляра:
путь_экземпляра

узлы: 3 узла, db2nodes.cfg:

- 0 hostA 0 hostA0switch
- 1 hostA 1 hostA1switch
- 2 hostB 0 hostBswitch

Имя базы данных:
db2instDB

Сервер администратора:

имя: db2as

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. В следующей таблице представлены примеры имен путей для каждой из поддерживаемых платформ EEE:

Таблица 1. Примеры имен путей для поддерживаемых платформ EEE

Пути	DB2 UDB EEE for AIX	DB2 UDB EEE for Solaris	DB2 UDB EEE for Windows NT
путь_установки	/usr/lpp/<v_r_ID>	/opt/IBMdb2/<v_r_ID>	C:\sqllib
путь_экземпляра	/home/db2inst/sqllib	/home/db2inst/sqllib	C:\profiles\db2inst
путь_das	/home/db2as/sqllib	/home/db2as/sqllib	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 EEE for AIX Версии 5.2 `<v_r_ID>` - `db2_05_00`.

При установке DB2 UDB EEE 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 от Центра управления или Ассистента конфигурирования клиента.

Конфигурирование сервера администратора: Сервер администратора - это точка управления, выполняющая определенные задачи для Центра управления. На физическом компьютере может быть не более одного сервера администратора. Чтобы Центр управления мог управлять экземпляром EEE, включающем несколько компьютеров, хотя бы на одном из этих компьютеров должен быть

запущен сервер администратора. Этот сервер администратора (db2as) “представляет” систему, отображаемую в дереве объектов Центра управления как родительская для управляемого экземпляра DB2 (db2inst).

В нашем примере db2inst состоит из трех узлов, размещенных на двух физических компьютерах или хостах. Минимальные требования можно выполнить, запустив db2das на хосте hostA **или** на хосте 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 сконфигурировано несколько логических узлов (MLN).
2. Не обязательно создавать ID сервера администратора (db2as) на всех хостах. Требуется только, чтобы он существовал на хосте, на котором он работает. Также не требуется, чтобы начальный каталог ID сервера администратора был смонтирован на всех хостах. В нашем примере ID db2as должен существовать на хосте hostA и не обязателен на хосте hostB, а начальный каталог db2as не требуется монтировать на хосте hostB.

Связь Центра управления с сервером администратора: Порты служб: Центр управления использует для связи с сервером администратора порт службы TCP - порт номер 523. Поскольку этот порт зарезервирован для использования только DB2 UDB, нет необходимости вставлять новые записи в *файл_служб_tcp*.

Межузловая связь для управления: Порты служб: Для некоторых задач управления сервер администратора должен установить связь со всеми узлами. Для этого в *файле_служб_tcp* должен быть определен конкретный порт TCP для каждого хоста, входящего в экземпляр.

Примечание: В Windows NT система EEE попытается сама добавить информацию о портах TCP в *файл_служб_tcp*.

В нашем примере раздел db2inst определен на двух хостах, hostA и hostB. Как указано в “Пример среды” на стр. 89, порт 16000 свободен на обоих хостах. Поэтому в *файл_служб_tcp* на обоих хостах hostA и hostB нужно вставить следующую строку.

```
db2ccmsrv  16000/tcp
```

Нужно задать именно это указанное выше имя порта `db2ccmsrv`; на всех хостах должен использоваться один и тот же выбранный номер порта.

Межузловая связь для управления: Серверы UNIX DB2 EEE: После того, как в *файл_служб_tcp* на хостах hostA и hostB вставлена строка с номером порта TCP, на всех хостах экземпляра необходимо запустить процесс или демон управляющей программы приема db2cclst. Это можно сделать вручную из

командной строки или можно сконфигурировать систему для автоматического запуска db2cclst при каждой загрузке системы:

Вручную:

Зарегистрировавшись с ID экземпляра, которым нужно управлять (db2inst), введите на хосте hostA или hostB команду:

```
rah '<путь_установки>/bin/db2cclst'
```

Например, в системе AIX эта команда может выглядеть так:

```
rah '/usr/lpp/<v_r_ID>/bin/db2cclst'
```

Команда `rah` находится в подкаталоге `instance` подкаталога для данной версии и выпуска продукта. Точное имя подкаталога для конкретной версии и выпуска зависит от операционной системы. Каталог `instance` - это начальный каталог нужного экземпляра.

В данном случае `<v_r_ID>` - это зависящий от платформы идентификатор версии и выпуска. Например, в DB2 UDB EEE for AIX Версии 5.2 `<v_r_ID>` - `db2_05_00`.

Автоматически:

От имени пользователя root добавьте соответствующую запись в файл `/etc/inittab`

. Например, в системе AIX эту команду надо вызвать на hostA и hostB:

```
mkitab "db2cclst::once:su - db2inst -c
   /usr/lpp/<v_r_ID>/bin/db2cclst"
```

Каждый раз при загрузке компьютера программа db2cclst будет запускаться автоматически.

В этой таблице `<v_r_ID>` - это зависящий от платформы идентификатор версии и выпуска. Например, в DB2 UDB EEE for AIX Версии 5.2 `<v_r_ID>` - `db2_05_00`.

Чтобы проверить, активен ли демон приема на каждом хосте, зарегистрировавшись с ID экземпляра, db2inst, введите команду:

```
rah 'ps -ef | grep db2cclst'
```

Если оказалось, что процесс db2cclst запущен не на всех хостах, можно добавить в файл `/etc/syslog.conf` на каждом хосте следующую строку, чтобы получить дополнительную диагностическую информацию:

```
*.info  /tmp/db2/user.info
```

где вместо файла `/tmp/db2/user.info` можно задать другой файл по вашему выбору.

Примечание: Этот файл должен существовать; необходимо запросить демон SYSLOG, чтобы он заново прочитал свой файл конфигурации после внесения в него изменений:

```
kill -1 <syslogd PID>
```

где `syslogd PID` можно узнать при помощи команды

```
ps -ef | grep syslogd
```

Затем, вручную запустив программу приема (как описано выше), можно посмотреть файл системного журнала `/tmp/db2/user.info` на хосте, на котором программа приема не была запущен автоматически, чтобы увидеть сообщения об ошибках, сгенерированные программой db2cclst.

Межузловая связь для управления: Серверы Windows NT DB2 EEE: Служба удаленных команд DB2 (db2rcmd.exe) автоматически поддерживает межузловую связь для управления. При возникновении ошибок диагностическая информация сохраняется в реестре Windows NT.

Защита: Чтобы сервер администратора мог выполнять задачи управления для экземпляра, он должен обладать достаточными полномочиями. В частности, этот сервер должен быть администратором системы (SYSADM) для управляемого экземпляра.

Необходимо предоставить серверу администратора такие полномочия для всех экземпляров DB2, которыми он будет управлять. Этими экземплярами могут быть экземпляры, установленные на том же компьютере, что и сервер администратора. Чтобы можно было управлять экземпляром DB2 EEE, по крайней мере один из серверов разделов базы данных должен находиться на том же компьютере, что и сервер администратора.

Например, в системе UNIX один из способов предоставить серверу администратора db2as требуемые полномочия для управления экземпляром db2inst - сделать совпадающими первичные группы db2inst и db2as. Другой способ - сделать первичную группу db2inst вторичной группой db2as, а первичную группу db2as - вторичной группой db2inst. Наконец, параметру конфигурации базы данных SYSADM_GROUP для db2inst можно присвоить имя первичной группы db2as.

В системе Windows NT db2as должен быть членом группы локальных администраторов на хостах hostA и hostB. Можно создать ID db2as и добавить его в группы локальных администраторов на обоих хостах или можно создать ID домена для db2as и добавить этот ID домена в группу локальных администраторов на каждом хосте.

Среда: При установке DAS нужно сконфигурировать некоторые переменные реестра, необходимые для его нормальной работы. Чтобы узнать текущие значения этих переменных, выполните следующие команды, зарегистрировавшись с ID экземпляра DB2, db2inst, или с ID DAS, db2das:

```
db2set -g
```

Должны быть определены следующие параметры со следующими значениями:

```
DB2SYSTEM=hostA
DB2ADMINSERVER=db2as
```

Также для связи Центра управления с сервером администратора переменная реестра DB2COMM должна иметь значение “TCPIP”. Чтобы проверить, задано ли это значение, зарегистрировавшись с ID сервера администратора (db2as), выполните следующую команду и проверьте глобальный уровень регистра (`-g`) и уровень экземпляра (`-i`) (достаточно, чтобы переменная была задана на одном из уровней):

```
db2set -all
```

Также проверьте, что параметр DB2COMM для экземпляра DB2 имеет значение “TCPIP” (что разрешает связь между Центром управления и db2inst); для этого, зарегистрировавшись с ID db2inst, введите команду:

```
db2set -all
```

Изменив этот параметр для сервера администратора, необходимо перезапустить его, чтобы изменения вступили в силу. Если этот параметр изменен для экземпляра DB2, нужно перезапустить этот экземпляр. Для db2inst используйте команды *db2stop* и затем *db2start*, а для DAS - команды *db2admin stop* и *db2admin start*.

Поиск серверов администратора, экземпляров и баз данных: Функция поиска KNOWN позволяет находить экземпляры и базы данных на 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.

Чтобы сервер поддерживал поиск KNOWN, задайте для параметра *discover* в файле конфигурации DAS значение `KNOWN`. Чтобы сервер поддерживал поиск SEARCH, задайте для этого параметра значение `SEARCH`. Чтобы запретить функции поиска для сервера и всех его экземпляров и баз данных, задайте для этого параметра значение `DISABLE`.

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**

  Позволяет Ассистенту конфигурирования клиента обновлять список известных систем и добавлять в этот список новые системы при нажатии кнопки **Добавить системы**. Если параметр *discover* имеет значение KNOWN, Ассистент конфигурирования клиента не сможет выполнять поиск в сети.

- **SEARCH**

  Разрешает все возможности функции поиска KNOWN и разрешает поиск в сети.

  Значок “Другие системы (искать в сети)” появляется, только если задано это значение параметра. Это значение 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. Параметр *discover* по умолчанию имеет значение SEARCH на всех серверах администратора DB2, за исключением сервера администратора в среде UNIX Enterprise - Extended Edition, где его значение по умолчанию - KNOWN.

Дополнительные параметры для функции поиска SEARCH: Для функции поиска SEARCH требуется, чтобы на сервере (в файле конфигурации сервера администратора DB2) и на клиенте (в файле конфигурации менеджера баз данных) был задан параметр *discover_comm*.

Параметр *discover_comm* используется для управления протоколами связи, которые сервер использует для приема входящих запросов поиска от клиентов, а клиент - для передачи своих запросов поиска. Параметр *discover_comm* может иметь значение TCP/IP или NetBIOS. В настоящее время поддерживаются только эти протоколы.

Значение параметра *discover_comm* для сервера администратора должно совпадать или быть подмножеством значения, заданного для переменной DB2COMM.

Примечание: Чтобы избежать проблем с Центром управления и Ассистентом конфигурирования клиента, убедитесь, что в реестре DB2 переменная реестра DB2COMM задана при помощи команды **db2set**. Не рекомендуется использовать для задания переменной реестра DB2COMM другие методы.

На сервере параметр *discover_comm* задается в файле конфигурации сервера администратора. На клиенте (или сервере, работающем в качестве клиента) параметр *discover_comm* задается в файле конфигурации менеджера баз данных.

Примечание: Для использования функции поиска SEARCH по крайней мере один протокол, заданный параметром *discover_comm* для клиента, должен совпадать со значением протокола, заданного параметром *discover_comm* для сервера администратора. Если совпадающих протоколов нет, сервер не будет отвечать на запросы клиента.

Чтобы узнать значение переменной реестра DB2COMM, введите команду:

```
db2set db2comm
```

Для настройки работы функции поиска SEARCH через протокол NetBIOS можно использовать две дополнительные переменные реестра профиля DB2: DB2DISCOVERYTIME и DB2NBDISCOVERYRECVBUFS. В большинстве случаев для этих переменных реестра можно использовать значения по умолчанию.

Переменные реестра профиля DB2DISCOVERYTIME и DB2NBDISCOVERRCVBUFS задаются в экземпляре клиента (или сервера, работающего в качестве клиента). Эти переменные реестра задаются так:

- Чтобы задать для переменной реестра DB2DISCOVERYTIME значение 60 секунд, введите команду:

  ```
  db2set db2discoverytime=60
  ```

  Это означает, что функция поиска SEARCH будет ждать ответ от серверов в течении 60 секунд.
- Чтобы задать для переменной реестра DB2NBDISCOVERRCVBUFS значение 20, введите команду:

  ```
  db2set db2nbdiscoverrcvbufs=20
  ```

  Эта переменная задает число буферов NetBIOS, которые будут выделены для одновременных ответных сообщений от найденных серверов.

Как скрыть экземпляры сервера и базы данных от функции поиска: На сервере может быть несколько экземпляров и несколько баз данных в этих экземплярах. Можно скрыть некоторые из них от процесса 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.

Задание параметров функции поиска: В файле конфигурации сервера администратора в системе сервера и в файле конфигурации менеджера баз данных на клиенте задаются параметры *discover* и *discover_comm*. Чтобы задать эти параметры:

- На сервере администратора:

  Измените файл конфигурации сервера администратора введя в командной строке:

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

  Остановите и заново запустите сервер администратора, введя следующие команды:

  ```
  db2admin stop
  db2admin start
  ```

  Примечание: Функция поиска SEARCH будет работать только с протоколами NetBIOS и TCP/IP.

- Используя Центр управления:

1. Запустите Ассистент конфигурирования клиента.
2. Нажмите кнопку **Параметры клиента**.
3. Выберите закладку **Связь**.
4. В окне **Параметры** выберите параметры, которые нужно изменить.
5. В поле **Значение** выберите значение для изменяемого параметра.
6. Нажмите кнопку **OK**, чтобы закрыть окна **Параметры клиента**. Откроется окно сообщения DB2.
7. Нажмите кнопку **OK** и перезапустите прикладные программы, чтобы сделанные изменения вступили в силу.

Примечание: Если в значение параметра *discover_comm* входит NETBIOS, параметр имени рабочей станции (*nname*) должен быть задан и на

клиенте, и на сервере администратора. Кроме этого, в переменной реестра DB2NBADAPTERS должен быть задан номер используемого адаптера.

При помощи Центра управления задайте параметры *discover_inst* и *discover_db*:

1. Раскройте дерево объектов и найдите папку **Экземпляры**.
2. Щелкните правой кнопкой мыши по экземпляру и выберите из всплывающего меню пункт **Конфигурировать**.
3. На странице "Среда" выберите параметр *discover_inst*.
4. Чтобы разрешить клиентам находить этот экземпляр сервера, выберите значение **ENABLE** и нажмите кнопку **OK**.
5. В дереве объектов щелкните правой кнопкой мыши по базе данных и выберите из всплывающего меню пункт **Конфигурировать**.
6. На странице "Среда" выберите параметр *discover_db*.
7. Чтобы разрешить клиентам находить эту базу данных, выберите значение **ENABLE** и нажмите кнопку **OK**.

Настройка DAS для работы с Ассистентом конфигурирования клиента и Центром управления

Необходимо сконфигурировать функцию поиска DB2 для получения информации о системах в сети. Функция поиска DB2 используется Ассистентом конфигурирования клиента и Центром управления. Чтобы функция поиска DB2 получала правильную информацию, может понадобиться обновление списков экземпляров и конфигурации сервера администратора DB2.

Обновление списков экземпляров: Сервер администратора DB2 может не знать обо всех экземплярах в системе многораздельных баз данных, так как изначально после создания экземпляра о нем знает только сервер администратора на компьютере - владельце экземпляра.

Если экземпляр создан на компьютере, на котором нет сервера администратора, можно создать сервер администратора на этом компьютере, чтобы сделать этот экземпляр известным.

Если создано несколько серверов администратора и нужно, чтобы каждый сервера администратора знал обо всех экземплярах в системе распределенных баз данных, выполните следующие шаги:

1. **Для каждого сервера администратора**

   Выполните команду **db2ilist** на компьютере этого сервера администратора, чтобы получить список экземпляров, известных этому серверу.

Примечание: Если этот список содержит все экземпляры, не нужно выполнять оставшиеся шаги и можно перейти к следующему разделу.

2. **Для каждого экземпляра, который отсутствует в списке экземпляров, полученном на предыдущем шаге**

   Выполните команду **db2nlist** на компьютере - владельце этого экземпляра, чтобы увидеть, есть ли запись для компьютера, на котором находится сервер администратора. Если такой записи нет, необходимо выполнить команду **db2ncrt**, чтобы добавить этот компьютер к данному экземпляру.

   Примечание: На компьютере сервера администратора должен быть доступен совместно используемый сетевой диск для данного экземпляра.

Обновление конфигурации сервера администратора

По умолчанию программа установки задает в качестве значения переменной реестра DB2SYSTEM имя компьютера Windows NT. 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.

Для обновления конфигурации сервера администратора можно использовать два способа:

- Если нужно иметь возможность выбирать узел координатора из списка систем DB2, задайте DISCOVER=SEARCH (это значение по умолчанию) в каждом из файлов конфигурации серверов администраторов DB2.

  Если есть несколько серверов администратора, в интерфейсе Ассистента конфигурирования клиента или Центра управления один экземпляр может выводиться нескольких системах, однако эти системы будет иметь разные пути доступа к экземпляру. Пользователи могут выбирать разные системы DB2 в качестве узлов координаторов для связи и таким образом перераспределять рабочую нагрузку.
- Если не нужно, чтобы пользователи могли выбирать узел координатора, задайте значение DISCOVER=KNOWN на всех серверах администратора, кроме одного, в конфигурации сервера администратора которого задайте DISCOVER=SEARCH. Сервер раздела баз данных, на котором находится этот последний сервер администратора, используется функцией поиска качестве узла координатора при установлении соединений.

Создание файла конфигурации узлов

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 для Windows NT: При использовании DB2 Enterprise - Extended Edition в Windows NT файл конфигурации узлов создается автоматически при создании экземпляра. Не следует пытаться создавать или изменять файл конфигурации узлов вручную.

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`. (Информацию о хранимых процедурах смотрите в главе “Хранимые процедуры” в руководстве *Administration Guide: Performance*.) Второе исключение - использование особых пользовательских функций (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_узла имя_хоста [логический_порт [сетевое_имя]]
```

Переменные разделяются пробелами. Переменные:

номер_узла

Номер узла (может быть от 0 до 999), уникально определяющий узел. Номера узлов должны идти в возрастающем порядке. В последовательности номеров могут быть пропуски.

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

После удаления узла его номер можно повторно использовать для нового добавляемого узла.

Номер узла используется для генерации имени узла в каталоге баз данных. Его формат:

```
NODEnnnn
```

nnnn - это номер узла, дополненный слева нулями. Этот номер узла используется также командами CREATE DATABASE и DROP DATABASE.

имя_хоста
Имя хоста IP-адреса для межраздельной связи. (Исключение - когда задан параметр сетевое_имя. В этой ситуации для большинства связей используется сетевое_имя, а имя_хоста используется только для команд DB2START, DB2STOP и db2_all.)

логический_порт
Этот необязательный параметр задает номер логического порта для этого узла. Этот номер и имя экземпляра менеджера баз данных используются для идентификации записи имени службы TCP/IP в файле etc/services.

Комбинация IP-адреса и логического порта используется в качестве общеизвестного адреса; для поддержки соединений между узлами она должна быть уникальной 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 несколько *номеров_узлов*), нужно присвоить этим логическим узлам номера *логических_портов*, начиная с нуля в восходящем порядке 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.

В следующем примере показан возможный файл конфигурации узлов для системы RS/6000 SP, в которой у хоста SP2EN1 есть несколько интерфейсов TCP/IP и два логических узла; он использует сетевое имя SP2SW1 для интерфейса DB2 Universal Database. Также в нем заданы номера узлов, начиная с 1 (а не с 0), и в последовательности *номеров_узлов* есть пропуск:

```
номер_узла   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 NT.) Будьте осторожны, чтобы не нарушить целостность информации в этом файле - для распределения данных требуется, чтобы номера узлов не менялись. Файл конфигурации узлов блокируется после выполнения команды DB2START и разблокируется, когда команда DB2STOP останавливает менеджер баз данных.

Чтобы изменить этот файл, когда он заблокирован, можно использовать команду DB2START. Например, можно ввести команду DB2START с опцией RESTART или с опцией ADDNODE.

Примечание: Если выполнение команды DB2STOP было неуспешным и файл конфигурации узлов остался заблокированным, используйте для его разблокирования команду DB2STOP FORCE.

Создание файла конфигурации базы данных

Для каждой базы данных создается также *файл конфигурации 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).

Эти параметры подробно описаны в главе “Конфигурирование DB2” руководства *Administration Guide: Performance*.

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 многораздельной базы данных файлы конфигурации на всех разделах базы данных должны быть одинаковы. 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**.

Использование файлов ответов для копирования конфигурации

Утилита генерации файла ответов, *db2rspgn*, позволяет создать файл ответов, который можно использовать для повторной установки этой системы или для конфигурирования других систем с теми же переменными реестра, параметрами

конфигурации менеджера баз данных и параметрами конфигурации администратора, что и в текущей системе.

Установив в системе один или несколько продуктов DB2 и настроив параметры для этой среды, можно использовать утилиту *db2rspgn*, чтобы записать нужные значения в файл ответов. Этот файл ответов можно затем использовать для создания идентичной системы.

В командной строке задается каталог назначения для файлов ответов и всех сопутствующих файлов. Кроме этого можно (необязательно) задать экземпляры, конфигурацию которых нужно скопировать, а также запретить копирование конфигурации экземпляров сервера администратора и/или сервера связей данных. Дополнительную информацию о разворачивании системы смотрите в книге *Administering Satellites Guide and Reference*.

Подробное описание синтаксиса этой утилиты и использования сгенерированных файлов ответов смотрите в соответствующем руководстве *Быстрый старт*.

Включение 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 в Windows NT

При использовании DB2 Enterprise - Extended Edition в среде Windows NT диапазон портов TCP/IP автоматически добавляется в файл служб; это делает:

- Программа установки, когда она создает экземпляр или добавляет новый узел
- Утилита db2icrt, когда она создает новый экземпляр
- Утилита db2ncrt, когда она добавляет первый узел на данном компьютере

Дополнительную информацию смотрите в руководстве *DB2 Enterprise - Extended Edition for Windows Quick Beginnings*.

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 `/etc/services` используется совместно, число заданных в нем портов должно быть не меньше наибольшего числа разделов многораздельных баз данных этого экземпляра. При выделении портов не забудьте учесть все компьютеры, которые могут использоваться в качестве резервных.

Если файл `/etc/services` не используется совместно, применяются те же правила, а кроме того, заданные для этого экземпляра DB2 записи должны быть одинаковыми во всех файлах `/etc/services` (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. Чтобы сделать это, задайте в файле `etc/services` две строки, чтобы указать 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.

```
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 (_).

Глава 3. 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.

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"
```

Если нужно создать базу данных в другом, возможно, удаленном, менеджере баз данных, смотрите раздел “Использование нескольких экземпляров менеджера баз данных” на стр. 55. В нем также кратко описана команда,

которую нужно использовать для выполнения любых задач администратора уровня экземпляра для экземпляров, отличных от экземпляра по умолчанию, в том числе для удаленных экземпляров.

Примечание: Информацию о положении базы данных по умолчанию и о задании в команде CREATE DATABASE другого положения смотрите в руководстве *Command Reference*.

В следующих разделах описываются задачи, выполняемые вами или автоматически выполняемые менеджером баз данных при создании базы данных:

Дополнительную информацию о физической реализации базы данных смотрите в руководстве *Administration Guide: Planning*.

Определение исходных групп узлов

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 NODE и DROP NODE.

Определяются три группы узлов:

- 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.

Определение исходных табличных пространств

При создании базы данных определяются три табличных пространства:

- SYSCATSPACE для таблиц системного каталога (смотрите раздел “Определение таблиц системного каталога” на стр. 108)
- 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 это базы данных (дополнительную информацию о физических каталогах базы данных смотрите в руководстве *Administration Guide: Planning*). Размеру экстента для этих табличных пространств присваивается значение по умолчанию.

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 OS/2 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.

Условие MANAGED BY команды CREATE DATABASE имеет тот же формат, что и условие MANAGED BY команды CREATE TABLESPACE. Дополнительные примеры смотрите в разделе “Создание табличного пространства” на стр. 115.

Перед созданием базы данных прочтите руководство *Administration Guide: Planning* и информацию о проектировании и выборе табличных пространств.

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:

- В схеме SYSFUN создается набор пользовательских функций (UDF). Дополнительную информацию об этих создаваемых системой функциях смотрите в руководстве *SQL Reference*.
- В схеме SYSCAT для таблиц системного каталога создается набор производных таблиц только для чтения. Информацию об этих производных таблицах смотрите в разделе “Catalog Views” (Производные таблицы каталога) справочника *SQL Reference*.
- 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. Информацию об этих производных таблицах смотрите в разделе “Updatable Catalog Views” (Изменяемые производные таблицы каталога) справочника *SQL Reference*.

После создания базы данных можно ограничить доступ к производным таблицам системного каталога, как описано в разделе “Защита производных таблиц системного каталога” на стр. 288.

Определение каталогов баз данных

При создании или настройке новой базы данных используются три каталога.

- Локальный каталог баз данных
- Системный каталог баз данных
- Каталог узлов

Локальный каталог баз данных

Файл *локального каталога баз данных* существует в каждом пути (называемом в некоторых операционных системах “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.

Чтобы увидеть содержимое этого файла для конкретной базы данных, используйте следующую команду (*положение* задает положение этой базы данных):

```
LIST DATABASE DIRECTORY ON положение
```

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. Информацию в том, как внести в каталог базы данных, смотрите в разделе “Внесение базы данных в каталог” на стр. 114.

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.

Чтобы увидеть содержимое этого файла, используйте команду LIST DATABASE DIRECTORY, *не* задавая положение файла каталога баз данных.

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. Эти файлы описаны в разделе “Разрешение разделения данных” на стр. 59.

Каталог узлов

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 подключению 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. Локальный узел нужно внести в каталог, если с этого клиента пользователя должны быть доступны несколько экземпляров на одном компьютере.

Службы каталога DCE

DCE (Distributed Computing Environment, распределенная вычислительная среда) - это разработанная организацией Open Systems Foundation** (OSF**) архитектура API, содержащая инструменты и службы для создания, использования и поддержки прикладных программ в разнородной распределенной вычислительной среде. DCE - это промежуточный слой между операционной системой, сетью и распределенными прикладными программами, позволяющий прикладным программам клиента обращаться к удаленным серверам.

При использовании локальных каталогов информация о физическом положении базы данных назначения хранится на каждой отдельной рабочей станции клиента в каталоге баз данных и каталоге узла. Поэтому администратор баз данных тратит много времени на обновление и изменение этих каталогов. Службы каталога DCE позволяют использовать вместо локальных каталогов один централизованный каталог. Они позволяют записывать информацию о базе данных или экземпляре менеджера баз данных в одном месте и вносить все обновления и изменения только в этом одном месте.

DCE не обязательна для работы DB2; если вы используете эту среду, смотрите дополнительную информацию в разделе “Приложение B. Использование служб каталогов среды DCE” на стр. 339.

Службы каталогов протокола LDAP

Протокол LDAP (Lightweight Directory Access Protocol - протокол упрощенного доступа к каталогам) - это промышленный стандарт для метода доступа к службам каталогов. Служба каталогов хранит информацию о ресурсах для множества систем и служб в распределенной среде и обеспечивает клиенту и серверу доступ к этим ресурсам. 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.

LDAP не обязателен для работы DB2; если вы используете эту среду, смотрите дополнительную информацию в разделе “Приложение J. Службы каталогов протокола LDAP” на стр. 421.

Создание групп узлов

Для создания группы узлов используется оператор CREATE NODEGROUP. В этом операторе задается набор узлов, на которых располагаются контейнеры табличных пространств и данные таблиц. Этот оператор также:

- Создает карту разделения для этой группы узлов. Подробную информацию о карте разделения смотрите в руководстве *Administration Guide: Planning*.
- Генерирует ID карты разделения.
- Вставляет записи в следующие таблицы каталога:
  - SYSCAT.NODEGROUPS
  - SYSCAT.PARTITIONMAPS
  - SYSCAT.NODEGROUPDEF

Чтобы создать группу узлов с помощью Центра управления:

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

Чтобы создать группу узлов из командной строки, введите команду:

```
CREATE NODEGROUP <имя> ON NODES (<значение>,<значение>)
```

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

```
CREATE NODEGROUP mixng12 ON NODES (1,2)
```

Дополнительную информацию о создании групп узлов смотрите в руководстве *SQL Reference*.

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

IBMTEMPGROUP. (Дополнительную информацию о группах узлов смотрите в руководстве *Administration Guide: Planning*.)

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.

Дополнительную информацию смотрите в разделе *Data Movement Utilities Guide and Reference*.

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. Информацию об этом смотрите в руководстве *Быстрый старт* для конкретной платформы.

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.

Внесение 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.

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 `INSTANCE_C`.

```
catalog database b as b at node instance_c
```

Примечание: Команда CATALOG DATABASE используется также на узлах клиентов для внесения в каталог баз данных, расположенных на компьютерах серверов. Дополнительную информацию смотрите в руководстве *Быстрый старт* для конкретной платформы.

Информацию о каталоге ячейки DCE смотрите в разделах “Службы каталога DCE” на стр. 111 и “Приложение B. Использование служб каталогов среды DCE” на стр. 339.

Примечание: Для улучшения производительности можно кэшировать в памяти файлы каталогов (включая каталог баз данных). (Информацию о том, как включить кэширование каталогов, смотрите в главе “Поддержка кэша каталогов (dir_cache)” руководства *Administration Guide: Performance*.) 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`.

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`.

Дополнительную информацию о кэшировании каталогов смотрите в главе “Поддержка кэша каталогов (dir_cache)” руководства *Administration Guide: Performance*.

Создание табличного пространства

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.

Информацию о проектировании табличных пространств смотрите в руководстве *Administration Guide: Planning*.

Синтаксис оператора CREATE TABLESPACE подробно описан в руководстве *SQL Reference*. Информацию о табличных пространствах SMS и DMS смотрите в руководстве *Administration Guide: Planning*.

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 OS/2 или Windows NT 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 OS/2 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 (дополнительную информацию о физических каталогах базы данных смотрите в руководстве *Administration Guide: Planning*).

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 узлов. Если в операторе не задан следующий параметр, используется группа узлов 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
   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. Описание параметров OVERHEAD (затраты) и TRANSFERRATE (скорость передачи) смотрите в главе “Настройка производительности прикладных программ” руководства *Administration Guide: Performance*.

DB2 может значительно улучшить производительность последовательного ввода-вывода при помощи последовательной предварительной выборки, которая использует параллельный ввод-вывод. Подробную информацию об этом средстве смотрите в главе “Последовательная предварительная выборка” руководства *Administration Guide: Performance*.

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 контейнеров к табличному пространству 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. Дополнительную информацию смотрите в главе "Последовательная предварительная выборка" руководства *Administration Guide: Performance*.

Создание системного временного табличного пространства

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".

Примечание: 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.

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')
```

При создании системного временного табличного пространства можно задавать только группу узлов IBMTEMPGROUP.

Создание пользовательского временного табличного пространства

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 узлов, кроме IBMTEMPGROUP. По умолчанию при создании пользовательского временного табличного пространства используется группа узлов IBMDEFAULTGROUP.

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

Создание табличных пространств в группах узлов

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 узлов. Табличное пространство создается в группе узлов. После того, как табличное пространство было определено в конкретной группе узлов, оно должно оставаться в этой группе узлов - его нельзя перевести в другую группу узлов. Для задания группы узлов для табличного пространства используется оператор CREATE TABLESPACE.

Непосредственный ввод-вывод

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). (Исключения - операционные системы Windows 95 и Windows 98.) В следующем списке показаны физические и логические методы определения устройств этого типа:

- 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:

  ```
  \\.\PhysicalDisk5
  ```

- В Windows для задания логического непосредственного раздела (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.

- **Примечание:** Чтобы была возможность записывать журналы на устройство, должна быть установлена операционная система Windows NT Версии 4.0 с Service Pack 3.

- В системах на основе UNIX используйте символьное имя последовательного устройства, например /dev/rhd0

Использование непосредственного ввода-вывода в Linux

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.

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

- Один или несколько свободных разделов диска IDE или SCSI
- Ядро Linux 2.4.0 или новее (в некоторых дистрибутивах Linux непосредственный ввод-вывод предлагается на ядре 2.2).
- 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
- DB2 Версии 7.1 FixPak 3 или новее

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

Дистрибутив	Узлы непосредственных устройств	Контроллер непосредственных устройств
RedHat 6.2	/dev/raw/raw1 - 255	/dev/rawctl
SuSE 7.0	/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.

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

   Device Boot   Start    End    Blocks   Id  System
/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 глобальные права чтения на контроллере непосредственного устройства и разделе диска. Задайте глобальные права чтения и записи на непосредственном устройстве:

```
# chmod a+r /dev/rawctl
# chmod a+r /dev/sdb1
# chmod a+rw /dev/raw/raw1
```

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.

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.

Синтаксис оператора CREATE SCHEMA подробно описан в руководстве *SQL Reference*. Задаваемое для новой схемы имя не должно существовать в системных каталогах и не может начинаться с ″SYS″.

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.

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.

Прямой 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.

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
```

Задание схемы

Можно задать схему по умолчанию, используемую для заданных без спецификаторов имен объектов в динамических операторах SQL в конкретном соединении DB2. Для этого 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.

Синтаксис оператора SET SCHEMA подробно описан в руководстве *SQL Reference*.

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.

Создание и заполнение таблицы

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.

Синтаксис оператора CREATE TABLE подробно описан в руководстве *SQL Reference*. Информацию о создании сводных таблиц смотрите в разделе “Создание сводной таблицы” на стр. 154. Информацию о правилах именования таблиц, столбцов и других объектов базы данных смотрите в разделе “Приложение A. Правила именования” на стр. 331.

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 узлов 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, описанный в руководстве *SQL Reference*
- Команды LOAD или IMPORT, описанные в руководстве *Command Reference*
- Утилиту autoloader при работе в среде многораздельной базы данных, как описано в книге *Data Movement Utilities Guide and Reference*.

Перемещение данных в таблицы и из таблиц подробно рассмотрено в разделе *Data Movement Utilities Guide and Reference*.

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. Дополнительную информацию смотрите в руководстве *SQL Reference*.

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. Информацию о правилах именования смотрите в разделе “Приложение A. Правила именования” на стр. 331.

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 (описаны в разделе “Создание пользовательского типа” на стр. 149).

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.

В следующих разделах на основе предыдущего примера рассмотрены различные стороны создания таблицы:

- “Особенности столбцов больших объектов (LOB)”
- “Определение ограничений уникальности” на стр. 128
- “Определение генерируемых столбцов в новых таблицах” на стр. 133
- “Создание пользовательской временной таблицы” на стр. 135
- “Определение столбца идентификации в новой таблице” на стр. 136
- “Создание последовательности” на стр. 137
- “Сравнение столбцов идентификации (IDENTITY) и последовательностей” на стр. 139
- “Создание типизированной таблицы” на стр. 140
- “Заполнение типизированной таблицы” на стр. 140
- “Таблица иерархии” на стр. 140
- “Создание таблицы в нескольких табличных пространствах” на стр. 140
- “Создание таблицы в многораздельной базе данных” на стр. 141

Можно также создать таблицу, определенную на основе результата запроса. Такая таблица называется *таблицей сводки*. Дополнительную информацию смотрите в разделе “Создание сводной таблицы” на стр. 154.

Особенности столбцов 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 таких платформах, как OS/2, 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.

Определение ограничений

В этом разделе описывается, как определить ограничения:

- "Определение ограничений уникальности"
- "Определение реляционных ограничений" на стр. 130
- "Определение проверочных ограничений таблицы" на стр. 133.

Дополнительную информацию об ограничениях смотрите в разделе, посвященном планированию ограничений, в руководстве *Administration Guide: Planning*, а также в руководстве *SQL Reference*.

Определение ограничений 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.

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. Однако нельзя определить ограничение уникальности для подтаблицы.

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). У таблицы может быть только один первичный ключ. 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.

Определение реляционных ограничений

Для обеспечения реляционной целостности в определения таблицы и столбцов добавляются реляционные ограничения. Для задания реляционных ограничений используются условия FOREIGN KEY и REFERENCES операторов CREATE TABLE или ALTER TABLE. Дополнительную информацию о влиянии реляционных ограничений на типизированные таблицы или на родительские таблицы, являющиеся типизированными, смотрите в руководстве *SQL Reference*.

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 существуют 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. Смотрите раздел “Изменение структуры и содержимого таблицы” на стр. 199.

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 (смотрите раздел “Добавление первичных и внешних ключей” на стр. 203). Можно также использовать оператор CREATE SCHEMA, чтобы создать обе таблицы EMPLOYEE и DEPARTMENT одновременно (смотрите пример в руководстве *SQL Reference*).

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.

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. Дополнительную информацию о правилах DELETE при реализации реляционной целостности смотрите в руководстве *Administration Guide: Planning*.

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.

Определение проверочных ограничений таблицы

Проверочное ограничение таблицы задает условие поиска, применяемое для каждой строки таблицы, для которой определено это проверочное ограничение таблицы. 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. Смотрите раздел “Изменение структуры и содержимого таблицы” на стр. 199.

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. 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. Дополнительную информацию смотрите в разделе “Создание и заполнение таблицы” на стр. 123.

Создание пользовательской временной таблицы

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.

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.

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.

Дополнительную информацию об операторе DECLARE GLOBAL TEMPORARY TABLE смотрите в руководстве *SQL Reference*.

Примечание: 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

Определение столбца идентификации в новой таблице

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.

После создания таблицы нельзя изменить описание таблицы, включив в нее столбец идентификации.

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, будут гарантированно уникальными. Его значения всегда будут генерироваться системой 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.

Примечание: В настоящее время столбцы идентификации не поддерживаются в среде многораздельных баз 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.

Дополнительную информацию об определении столбца идентификации для новой таблицы смотрите в руководстве *SQL Reference*.

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.

В отличии от атрибута столбца идентификации, последовательность не привязана к конкретному столбцу таблицы и не связана с столбцом уникальности таблицы, доступ к ней возможен не только через этот столбец таблицы.

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 заданное целочисленное приращение. Значение приращения по умолчанию - 1.
- Значение счетчика допускает восстановление. Когда требуется восстановление, значение счетчика воссоздается по журналам.
- Значения могут для повышения производительности помещаться в кэш. 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 при этом потеряно.

Если база данных, содержащая одну или несколько последовательностей, восстанавливается до более раннего момента времени, для некоторых последовательностей возникает опасность генерации повторных значений. Чтобы исключить появление повторяющихся значений, не следует подвергать базу данных с последовательностями восстановлению до более раннего момента времени.

Последовательности поддерживаются только в одноузловых базах данных.

Для работы с последовательностями используются два выражения.

Выражение PREVVAL возвращает последнее значение, сгенерированное для данной последовательности предыдущим оператором в текущем сеансе.

Выражение NEXTVAL возвращает следующее значение данной последовательности. Новый член числовой последовательности генерируется, когда в выражении NEXTVAL задается имя этой последовательности. Однако если в запросе есть несколько экземпляров выражения NEXTVAL с одинаковыми именами последовательности, счетчик последовательности увеличивается 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 таблице ссылка на член числовой последовательности делается при помощи выражения NEXTVAL, а во всех остальных таблицах - 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)
```

Выражения NEXTVAL и PREVVAL можно использовать в следующих контекстах:

- оператор INSERT, условие VALUES
- оператор SELECT, список SELECT
- оператор присваивания SET
- оператор UPDATE, условие SET
- оператор VALUES или VALUES INTO

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.

Характеристики столбцов идентификации:

- Столбец идентификации может быть определен в таблице только в момент ее создания. 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 последовательность генерирует последовательные значения, которые можно использовать в любом операторе SQL.
- Объект последовательность может использоваться любой прикладной программой, и есть два выражения, дающие либо очередное значение в данной последовательности, либо значение, сгенерированное 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 позволяет использовать одно и то же значение 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 работающих с ней прикладных программ.

Создание типизированной таблицы

Для создания типизированной таблицы можно использовать разновидность оператора CREATE TABLE. Всю необходимую информацию о типизированных таблицах можно найти в руководстве *Application Development Guide*.

Заполнение типизированной таблицы

После того, как созданы структурированные типы, созданы соответствующие таблицы и подтаблицы, можно заполнить типизированную таблицу данными. Всю необходимую информацию о типизированных таблицах можно найти в руководстве *Application Development Guide*.

Таблица иерархии

Таблица иерархии - это таблица, связанная с реализацией иерархии типизированных таблиц. Она создается одновременно с корневой таблицей иерархии. Всю необходимую информацию о таблицах иерархии можно найти в руководстве *Application Development Guide*.

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. Все табличные пространства должны существовать перед выполнением оператора CREATE TABLE. Раздельное размещение частей таблицы возможно только при использовании табличных пространств DMS.

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.

Дополнительную информацию об использовании для одной таблицы нескольких табличных пространств DMS смотрите в руководстве *Administration Guide: Planning*.

Дополнительную информацию об операторе CREATE TABLE смотрите в справочнике *SQL Reference*.

Создание таблицы в многораздельной базе данных

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 узлов.
- Таблицы могут быть расположены вместе в одном табличном пространстве или же в другом табличном пространстве, которое вместе с первым табличном пространством связано с этой же группой узлов. Дополнительную информацию смотрите в руководстве *Administration Guide: Planning*.

При создании таблицы в среде многораздельных баз данных нужно выбрать, кроме всего прочего, *ключ разделения*. Ключ разделения - это ключ, являющийся частью определения таблицы. Он определяет раздел, где хранится данная строка таблицы.

Важно правильно выбрать ключ разделения, потому что *впоследствии его нельзя будет изменить*. Более того, ключ разделения должен быть входить во все индексы уникальности (и, следовательно, в ключи уникальности и в первичный ключ). То есть если определен ключ разделения, ключи уникальности и первичные ключи должны включать в себя все столбцы, входящие в ключ разделения (но они могут содержать и дополнительные столбцы).

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.

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.

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

Нельзя использовать триггеры с псевдонимами.

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.

Преимущества использования триггеров:

- Ускорение разработки прикладных программ: Поскольку триггер хранится в базе данных, не требуется кодировать его действие в каждой прикладной программе.
- 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 нужно изменить только триггер, а не все прикладные программы.

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. Полную информацию о синтаксисе оператора CREATE TRIGGER смотрите в руководстве *SQL Reference*. Информацию о создании и использовании триггеров смотрите в руководстве *Application Development Guide*.

Примечание: 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.

Зависимости триггера

Все зависимости триггера от других объектов записываются в каталоге SYSCAT.TRIGDEP. Триггер может зависеть от многих объектов. Эти объекты и зависимые от них триггеры подробно описаны в главе об операторе DROP руководства *SQL Reference*.

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. (Дополнительную информацию смотрите в разделе “Зависимости операторов при изменении объектов” на стр. 230.)

Создание пользовательской функции 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 к данным, не являющимся данными DB2, или преобразовывать такие данные в таблицу DB2.

Например, табличная функция может брать файл и преобразовывать его в таблицу, может вносить в таблицу примеры данных из WWW или обращаться к базе данных Lotus Notes и возвращать информацию о письмах, например, дату, от кого получено письмо и текст сообщения. 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. Информацию о том, как обновлять статистики пользовательских функций в системном каталоге, смотрите в главе “Обновление статистики для пользовательских функций” смотрите в руководстве *Administration Guide: Performance*.

Подробную информацию об использовании оператора CREATE FUNCTION для написания пользовательских функций для конкретной прикладной программы

смотрите в руководстве *Application Development Guide*. Подробную информацию о синтаксисе пользовательских функций смотрите в руководстве *SQL Reference*.

Создание отображения функции

В базе данных объединения для отображения локальной функции или локального шаблона функции (описанного в разделе “Создание шаблона функции” на стр. 148) на функцию на одном или нескольких источниках данных нужно создать отображение функции. 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 между функцией Oracle AVGNEW и эквивалентной ей функцией 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.

Дополнительную информацию об использовании и создании отображений функций смотрите в руководстве *Application Development Guide*. Подробное описание синтаксиса оператора CREATE FUNCTION MAPPING смотрите в руководстве *SQL Reference*.

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.

Поскольку для такой функции нет локального выполняемого файла, вызов ее шаблона может завершиться с ошибкой, даже если эта функция доступна на 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.

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')
```

Дополнительную информацию об использовании и создании шаблонов функций смотрите в руководстве *Application Development Guide*. Подробное описание синтаксиса оператора CREATE FUNCTION смотрите в руководстве *SQL Reference*.

Создание пользовательского типа

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. Полный список всех типов данных смотрите в руководстве *SQL Reference*.

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.

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.

Следующий оператор SQL создает особый тип `t_educ` на основе типа smallint:

```
CREATE DISTINCT TYPE T_EDUC AS SMALLINT WITH COMPARISONS
```

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.

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.

Полную информацию о синтаксисе оператора CREATE DISTINCT TYPE смотрите в руководстве *SQL Reference*. Информацию о создании и использовании особых типов смотрите в руководстве *Application Development Guide*.

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. Подробную информацию об этом смотрите в руководстве *SQL Reference*.

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. Всю необходимую информацию о структурированных типах можно найти в руководстве *Application Development Guide*.

Создание отображения типов

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.

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 типов:

```
CREATE TYPE MAPPING MY_ORACLE_DEC FROM SYSIBM.DECIMAL(10,2)
TO SERVER ORACLE1 TYPE NUMBER([10..38],2)
```

Нельзя создать отображение типов для типов данных большой объект, LONG VARCHAR/VARGRAPHIC, DATALINK, структурированных или особых типов.

Дополнительную информацию об использовании и создании отображений типов смотрите в руководстве *Application Development Guide*. Подробное описание синтаксиса оператора CREATE TYPE MAPPING смотрите в руководстве *SQL Reference*.

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.

Производную таблицу можно создать для ограничения доступа к критической информации и разрешении свободного доступа к другим 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. Дополнительную информацию об операторе INSERT смотрите в руководстве *SQL Reference*.

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. Дополнительную информацию об общих табличных выражениях смотрите в руководстве *SQL Reference*.

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. Информацию о производных таблицах *только для чтения* смотрите в руководстве *SQL Reference*.

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, в которой перечислены все сотрудники отдела A00 с личными номерами и номерами телефонов:

```
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
```

Можно создать производную таблицу, используя в ее определении пользовательскую функцию. Однако чтобы обновить такую производную таблицу при изменении функций, придется отбросить эту производную таблицу и затем 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).

Создание типизированной производной таблицы

Для создания типизированной производной таблицы можно использовать оператор CREATE VIEW. Всю необходимую информацию о типизированных производных таблицах можно найти в руководстве *Application Development Guide*.

Создание сводной таблицы

Сводная таблица - это таблица, определение которой основано на результатах запроса. Сама сводная таблица содержит обычно предварительно вычисленные результаты, основанные на данных, существующих в таблице или таблицах, на основе которых создана эта сводная таблица. Если компилятор SQL определяет, что для выполнения запроса эффективнее использовать не базовую таблицу, а сводную, будет использована сводная таблица и результаты будут получены быстрее.

Для репликации сводной таблицы на всех узлах в среде многораздельных баз данных можно создать эту сводную таблицу с опцией репликации. Такие таблицы называются “реплицируемыми сводными таблицами”. Дополнительную информацию о таких таблицах смотрите в руководстве *Administration Guide: Planning*.

Примечание: Сводные таблицы нельзя использовать со статическим SQL или псевдонимами.

В общем случае сводная таблица или реплицируемая сводная таблица используется для оптимизации запроса, если уровень изоляции этой сводной таблицы или реплицируемой сводной таблицы больше или равен уровню изоляции запроса. Например, если запрос выполняется с уровнем изоляции стабильность на уровне указателя (CS), для оптимизации используются только сводные таблицы или реплицируемые сводные таблицы, определенные с уровнем изоляции CS или более высоким.

Для создания сводной таблицы используется оператор CREATE SUMMARY TABLE с условием AS *полная_выборка* и опциями IMMEDIATE или REFRESH DEFERRED.

Для имен столбцов сводной таблицы 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 создании сводной таблицы можно задать, должно ли содержимое этой таблицы обновляться автоматически при изменении базовой таблицы, или же для ее обновления будет использоваться оператор REFRESH TABLE. Чтобы сводная таблица автоматически обновлялась при изменении базовой таблицы (или таблиц), задайте ключевое слово REFRESH IMMEDIATE. Такое немедленное обновление удобно, когда:

- Существуют запросы, для выполнения которых с использованием базовой таблицы требуется много времени
- Базовая таблица (или таблицы) изменяется редко
- Обновление сводной таблицы не занимает много времени.

В этом случае сводная таблица содержит заранее вычисленные результаты. Если нужно, чтобы обновление сводной таблицы было отложено, задайте ключевое слово REFRESH DEFERRED. Сводные таблицы, для которых задано ключевое слово REFRESH DEFERRED, **не** будут отражать изменения базовых таблиц. Такие сводные таблицы следует использовать в тех случаях, когда это не существенно. Например, при выполнении запросов DSS может использоваться сводная таблица, содержащая унаследованные данные.

Сводная таблица, определенная с условием 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.

Специальному регистру SQL под именем CURRENT REFRESH AGE присваивается значение ANY или 99999999999999. Все девятки - это максимальное значение, которое может иметь этот специальный регистр, содержащий значение длительности времени типа данных DECIMAL(20,6).

Примечание: Сводные таблицы, определенные с REFRESH DEFERRED, не используются для оптимизации статического SQL.

Специальный регистр CURRENT REFRESH AGE используется для задания интервала времени, в течение которого сводная таблица, определенная с REFRESH DEFERRED, может использоваться для динамических запросов, прежде чем будет необходимо ее обновление. Для задания значения специального регистра CURRENT REFRESH AGE можно использовать оператор SET CURRENT REFRESH AGE. Дополнительную информацию о специальном регистре CURRENT REFRESH AGE и операторе SET CURRENT REFRESH AGE смотрите в руководстве *SQL Reference*.

Сводные таблицы, определенные с REFRESH IMMEDIATE, могут применяться как для статических, так и для динамических запросов; для таких сводных таблиц не нужно использовать специальный регистр CURRENT REFRESH AGE.

Примечание: Будьте осторожны, задавая для специального регистра CURRENT REFRESH AGE ненулевое значение. Когда для оптимизации обработки запросов разрешено использовать сводную таблицу, которая может не отражать текущие значения базовой таблицы, результаты запроса могут *не* точно отражать данные 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.

После выполнения операций, изменяющих исходные данные, сводная таблица уже не будет содержать точные данные. Нужно будет использовать оператор REFRESH TABLE. Дополнительную информацию смотрите в справочнике *SQL Reference*.

Если на основе допустимой *полной выборки* нужно создать новую базовую таблицу, задайте при создании этой таблицы ключевое слово DEFINITION

ONLY. Когда таблица будет создана, она будет считаться не сводной, а базовой таблицей. 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
```

Некоторые главные ограничения, касающиеся сводных таблиц:

1. Сводную таблицу нельзя изменять.
2. Нельзя изменить длину столбца базовой таблицы, если для этой таблицы существует сводная таблица.
3. Нельзя импортировать данные в сводную таблицу.
4. Нельзя создать индекс уникальности для сводной таблицы.
5. Нельзя создать сводную таблицу на основе результатов запроса, в котором используется один или несколько псевдонимов.

Полный список ограничений для сводных таблиц смотрите в руководстве *SQL Reference*.

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.

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.

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.

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.

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) не может быть определено 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 MVS/ESA для алиасов используются два разных понятия - алиас (ALIAS) и синоним (SYNONYM). Эти понятия отличаются от алиасов в DB2 Universal Database:

- Алиасы в DB2 for MVS/ESA:
  - Требуют для создания специальных полномочий или привилегия
  - Не могут ссылаться на другие алиасы.
- Синонимы в DB2 for MVS/ESA:
  - 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.

Создание оболочки

В базе данных объединения оператор CREATE WRAPPER используется для регистрации оболочки. Этот оператор определяет механизм, который может использоваться сервером объединения для взаимодействия с определенной категорией источников данных.

Для конкретных типов и версий источников данных, протоколов связи и операционных систем должны использоваться особые библиотеки. Например, базы данных объединения, работающие в операционной системе Windows NT и применяющие связь APPC, используют для доступа к источникам данных AS/400 и DB2 for OS/390 библиотеку `libdrda.dll`.

Чтобы использовать оператор CREATE WRAPPER, необходимо обладать полномочиями SYSADM или DBADM для базы данных объединения.

При создании оболочки с помощью Центра управления или командной строки эта оболочка регистрируется в базе данных объединения.

Чтобы создать оболочку с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Объекты базы данных объединения**.
2. Щелкните правой кнопкой мыши по папке **Объекты базы данных объединения** и выберите из всплывающего меню пункт **Создать оболочку**.
3. Введите необходимую информацию и нажмите кнопку **OK**.

Чтобы создать оболочку из командной строки, введите команду:

```
CREATE WRAPPER <имя_оболочки> LIBRARY '<имя_библиотеки>'
```

Следующий оператор SQL регистрирует оболочку ORACLE8 в операционной системе Windows NT:

```
CREATE WRAPPER ORACLE8 LIBRARY 'libnet8.dll'
```

Подробную информацию об использовании оператора CREATE WRAPPER смотрите в руководстве *SQL Reference*.

Создание сервера

В базе данных объединения создайте серверы, чтобы определить источники данных для DB2 и описать их характеристики: имя, оболочка, тип, версия, положение и опции. Эта информация используется для отображения псевдонимов на конкретные системы управления данными и для оптимизатора DB2. Информация о серверах хранится в производных таблицах каталога SYSCAT.SERVERS и SYSCAT.SERVEROPTIONS.

Примечание: В этом разделе термином ″сервер″ обозначаются источники данных, а не серверы DRDA или серверы DB2. Для обращения к другим источникам данных (например, Oracle) требуется DB2 Connect.

Объект сервера можно создать, только если создана оболочка.

Чтобы использовать этот оператор, необходимо обладать полномочиями SYSADM или DBADM для базы данных объединения.

Чтобы указать отличия в процессах аутентификации в DB2 и на серверах источников данных, можно создать отображения пользователей. Отображения пользователей подробно обсуждаются в разделе “Отображения пользователей” на стр. 254.

Если сервер отброшен, отбрасываются все зависящие от него объекты (такие как отображения пользователей, псевдонимы, отображения функций, отображения типов и планы).

При создании сервера задайте опции сервера. Эти опции содержат необходимую подробную информацию о сервере (например, имя узла). Опции сервера позволяют также задать особые значения, описывающие производительность и защиту.

Для создания сервера можно использовать Центр управления или процессор командной строки.

Чтобы создать сервер с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Серверы** в папке **Объекты базы данных объединения**.
2. Щелкните правой кнопкой мыши по папке **Серверы** и выберите из всплывающего меню пункт **Создать сервер**.
3. Введите необходимую информацию и нажмите кнопку **OK**.

Чтобы создать сервер из командной строки, введите команду:

```
CREATE SERVER <имя_сервера> TYPE <тип_сервера>
   VERSION <версия_сервера> WRAPPER <имя_оболочки>
   OPTIONS (<имя_опции_сервера> <строчная_константа>)
```

Следующий пример оператора SQL создает сервер Oracle с именем ORA8:

```
CREATE SERVER ORA8 TYPE ORACLE VERSION 8 WRAPPER ORACLE8 OPTIONS
(NODE 'ONODE')
```

Следующий пример оператора SQL создает сервер DB2 с именем DB2TEST:

```
CREATE SERVER DB2TEST TYPE DB2 VERSION 6.1 WRAPPER DB2UDB OPTIONS
(NODE 'DB2TEST', DBNAME 'TEST1')
```

Определение узла (NODE) в операторе SQL CREATE SERVER зависит от типа источника данных. Если источник данных - это DBMS DB2, это значение задает экземпляр DB2, содержащий одну или несколько баз данных. Обратите внимание на то, что в предыдущем примере опция DBNAME задает имя базы данных. Если источник данных - это DBMS DB2 for OS/390, это значение задает имя положения (LOCATION) системы DB2 for OS/390. Если источник данных - это DBMS Oracle, опция DBNAME не требуется, поскольку экземпляр Oracle содержит только одну базу данных.

Дополнительную информацию о синтаксисе оператора CREATE SERVER смотрите в руководстве *SQL Reference*. Дополнительную информацию об использовании оператора CREATE SERVER смотрите в руководстве *Дополнение по установке и конфигурированию*.

Использование опций сервера для определения характеристик источников данных и настройки процесса аутентификации

Можно задать переменные, которые называются *опциями сервера*; их значения влияют на то, как сервер объединения обращается к источнику данных. В этом разделе:

- Объясняется назначение опций сервера
- Описываются операторы SQL, применяемые для задания опций сервера
- Перечислены опции сервера и их значения

Назначение опций сервера

Опции сервера используются для:

- Задания или изменения информации об источниках данных. Определение сервера содержит как основную информацию об источнике данных (например, имя), так и информацию, которая может со временем измениться. Для задания некоторых изменяемых значений можно использовать опции сервера. Например, значение опции cpu_ratio указывает, насколько быстрее или медленнее процессор источника данных по сравнению с процессором системы DB2. Если в систему DB2 добавлены один или несколько процессоров, нужно изменить это значение.

- Поддержки аутентификации. Можно задать некоторые опции сервера, чтобы ID пользователей и пароли передавались на источник данных в требуемом регистре. Например, можно задать опцию fold_id, чтобы перед тем, как послать ID пользователя источнику данных, сервер объединения преобразовал его в требуемый этим источником данных регистр (верхний или нижний). Или, если ID пользователя задан на сервере объединения в требуемом регистре, можно задать опцию fold_id, чтобы предотвратить попытки сервера изменить регистр и избежать затрат на этот процесс.
- Оптимизации запросов. Некоторые опции сервера и их значения способствуют оптимизации. Например, в операторе CREATE SERVER можно задать некоторые показатели производительности в качестве значений опций. В частности, для опции cpu_ratio можно задать значение, указывающее отношение скоростей процессоров источника данных и сервера объединения. Для опции io_ratio можно задать значение, указывающее отношение скоростей устройств ввода-вывода источника данных и сервера объединения. При выполнении оператора CREATE SERVER эти показатели будут добавлены в производную таблицу каталога SYSCAT.SERVEROPTIONS и оптимизатор будет использовать их при создании плана доступа для этого источника данных. Если значение показателя изменилось (это может произойти, например, при изменении процессора источника данных), можно при помощи оператора ALTER SERVER внести новое значение в SYSCAT.SERVEROPTIONS. При разработке нового плана доступа для этого источника данных оптимизатор будет использовать новое значение.

Операторы SQL для опций сервера

Есть три оператора SQL, в которых можно задать значения для опций сервера: CREATE SERVER, ALTER SERVER и SET SERVER OPTION.

Оператор CREATE SERVER используется для задания значения опции, которое будет действовать неопределенно долгое время для множества соединений с источником данных. Используя этот оператор, можно задать для опции значение, отличное от значения по умолчанию, или, если у опции нет значения по умолчанию, задать для нее исходное значение.

Оператор ALTER SERVER используется, если после задания значения опции с помощью оператора CREATE SERVER нужно задать для нее другое значение, сохраняющееся для многих соединений.

Оператор SET SERVER OPTION используется для временного изменения значения опции сервера в течение одного соединения с базой данных. Операторы SET SERVER OPTION должны выполняться первыми в первой единице работы после соединения с этим источником данных.

Например, чтобы временно разрешить использование рекомендаций планов для сервера Oracle ORASEB1, используйте оператор:

```
SET SERVER OPTION plan_hints TO 'Y' FOR SERVER ORASEB1
```

Опции сервера и их значения

Опции сервера и их возможные значения описаны в приведенной ниже таблице. Если не указано иное, все значения опций сервера должны быть заключены в простые кавычки.

Таблица 2. Опции сервера и их значения

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

Таблица 2. Опции сервера и их значения (продолжение)

Опция	Допустимые значения	Значение по умолчанию
comm_rate	Задает скорость связи между сервером объединения и связанными с ним источниками данных. Выражается в мегабайтах в секунду.	’2.0’
connectstring	Задает свойства инициализации, необходимые для соединения с провайдером OLE DB. Полный синтаксис и семантику строки соединения смотрите в главе ″Data Link API of the OLE DB Core Components″ в руководстве *Microsoft OLE DB 2.0 Programmer’s Reference and Data Access SDK, Microsoft Press, 1998*.	Нет
cpu_ratio	Указывает, насколько быстрее или медленнее работает процессор источника данных по сравнению с процессором сервера объединения. Значение “1.0” указывает на равные скорости процессора источника данных и процессора сервера объединения. Значение <1.0 указывает на более низкую скорость процессора источника данных. Значение >1.0 указывает на более высокую скорость процессора источника данных.	’1.0’
dbname	Имя базы данных источника данных, к которой будет обращаться сервер объединения. Требуется для источников данных семейства DB2; не используется для источников данных Oracle**.	Никаких действий не требуется.
fold_id (смотрите примечания 1 и 4 в конце этой таблицы.)	Применяется для ID пользователей, посылаемых сервером объединения на источники данных для аутентификации. Допустимые значения: ’U’ Перед тем, как посылать ID пользователя на источник данных, сервер объединения переводит его в верхний регистр. Для источников данных семейства DB2 и Oracle** разумно выбрать это значение (смотрите примечание 2 в конце этой таблицы). ’N’ Сервер объединения не изменяет ID пользователя перед передачей его на источник данных. (Смотрите примечание 2 в конце этой таблицы.) ’L’ Перед тем, как посылать ID пользователя на источник данных, сервер объединения переводит его в нижний регистр. Если не задана ни одна из этих опций, сервер объединения пытается послать ID пользователя, переведя его в верхний регистр. В случае неудачи сервер пытается послать ID пользователя в нижнем регистре.	Никаких действий не требуется.

Таблица 2. Опции сервера и их значения (продолжение)

Опция	Допустимые значения	Значение по умолчанию
fold_pw (смотрите примечания 1, 3 и 4 в конце этой таблицы.)	Применяется для паролей, посылаемых сервером объединения на источники данных для аутентификации. Допустимые значения: 'U' Перед тем, как посылать пароль на источник данных, сервер объединения переводит его в верхний регистр. Для источников данных семейства DB2 и Oracle** разумно выбрать это значение. 'N' Сервер объединения не изменяет пароль перед передачей его на источник данных. 'L' Перед тем, как посылать пароль на источник данных, сервер объединения переводит его в нижний регистр. Если не задана ни одна из этих опций, сервер объединения пытается послать пароль, переведя его в верхний регистр. В случае неудачи сервер пытается послать пароль в нижнем регистре.	Никаких действий не требуется.
io_ratio	Указывает, насколько быстрее или медленнее работает система ввода-вывода источника данных по сравнению с системой ввода-вывода сервера объединения. Значение "1.0" указывает на равные скорости процессора источника данных и процессора сервера объединения. Значение <1.0 указывает на более низкую скорость процессора источника данных. Значение >1.0 указывает на более высокую скорость процессора источника данных.	'1.0'
узел	Задает имя, под которым источник данных определен как экземпляр для своей реляционной СУБД. Требуется для всех источников данных. Для источника данных семейства DB2 это имя узла, заданное в каталоге `узлов DB2` базы данных объединения. Чтобы увидеть содержимое этого каталога, используйте команду **db2 list node directory**. Для источника данных Oracle** это имя сервера, заданное в файле Oracle** `tnsnames.ora`. Чтобы узнать это имя на платформе Windows NT, задайте опцию **View Configuration Information** инструмента Oracle** SQL Net Easy Configuration.	Никаких действий не требуется.

Таблица 2. Опции сервера и их значения (продолжение)

Опция	Допустимые значения	Значение по умолчанию
пароль	Задает, нужно ли посылать источнику данных пароль.	'Y'
	'Y' Пароли всегда посылаются источнику данных и проверяются. Это значение по умолчанию.	
	'N' Пароли не посылаются источнику данных (независимо от заданных отображений пользователей) и не проверяются.	
	'ENCRYPTION' Пароли всегда посылаются источнику данных в зашифрованном виде и проверяются. Может использоваться только для источников данных семейства DB2, поддерживающих шифрование паролей.	
plan_hints	Задает включение *советов по плану*. Советы по плану - это фрагменты оператора, передающие дополнительную информацию оптимизаторам источников данных. Эта информация может улучшить производительность для определенных типов запросов. Советы по плану помогают оптимизатору источника данных решить, использовать индекс или нет, какой индекс использовать или какую использовать последовательность объединения таблиц.	'N'
	'Y' Советы по плану нужно включать для источника данных, если он их поддерживает.	
	'N' Советы по плану не нужно включать для источника данных.	
pushdown	'Y' DB2 будет рассматривать возможность выполнения операций на источнике данных.	'Y'
	'N' DB2 будет получать от удаленного источника данных только значения столбцов и не будет разрешать выполнение на источнике данных других операций (например, объединений).	

Таблица 2. Опции сервера и их значения (продолжение)

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

Примечания для этой таблицы:

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

Использование непосредственных сеансов с серверами

Непосредственные сеансы позволяют прикладным программам напрямую связываться с серверами, используя метод доступа клиентов этого сервера и его собственный диалект SQL.

Непосредственные сеансы удобны, когда:

- Прикладные программы должны создавать объекты на источнике данных или выполнять операции INSERT, UPDATE или DELETE

- DB2 не поддерживает какую-то особую операцию источника данных

Для ссылок на объекты в непосредственном сеансе должны использоваться истинные имена объектов (не псевдонимы).

Чтобы запустить непосредственный сеанс и обращаться к серверу напрямую, используйте оператор SET PASSTHRU. Этот оператор должен выполняться динамически. Пример такого оператора:

```
SET PASSTHRU BACKEND
```

Этот оператор открывает непосредственный сеанс с источником данных BACKEND.

Дополнительную информацию об операторе SET PASSTHRU и обработке SQL в непосредственных сеансах смотрите в руководстве *SQL Reference*.

Создание псевдонима

В базе данных объединения псевдонимы - это идентификаторы таблиц, алиасов и производных таблиц источников данных. В распределенных запросах обычно используются псевдонимы, а не имена таблиц или производных таблиц источника данных.

Псевдонимы - это одно из средств, используемых DB2 для обеспечения прозрачности положения. Для поиска и эффективного обращения к источнику данных, на который указывает псевдоним, используется информация о положении источника данных из определения сервера. Например, с помощью оператора ALTER SERVER можно изменить для сервера данные производительности или информацию о версии, не оказывая влияния ни на пользователей, ни на прикладные программы, которым не потребуется использовать новые псевдонимы или изменять код программ.

Для создания псевдонимов можно использовать Центр управления или процессор командной строки. Для одной таблицы или производной таблицы источника данных можно определить несколько псевдонимов.

Псевдонимы нельзя использовать в статических операторах SQL.

Перед созданием псевдонима запустите на источнике данных эквивалент команды RUNSTATS и обновите статистику для объектов источника данных. При создании псевдонима статистическая информация собирается из источников данных и сохраняется в каталоге базы данных объединения. Эти данные каталога содержат определения таблиц и столбцов и, если они доступны, определения индексов и статистики.

Следующий оператор SQL создает псевдоним CUSTOMER:

```
CREATE NICKNAME CUSTOMER for OS390A.SHAWNB.CUSTLIST
```

Чтобы использовать этот оператор, необходимо обладать полномочиями SYSADM или DBADM или иметь привилегию базы данных IMPLICIT_SCHEMA или привилегию схемы CREATEIN (для текущей схемы) для базы данных объединения.

Дополнительную информацию об использовании оператора CREATE NICKNAME смотрите в руководстве *SQL Reference*.

Ссылки на псевдонимы и объекты источников данных

Для ссылок на объекты источника данных обычно используются определенные псевдонимы. Исключение - ссылки в непосредственном сеансе (дополнительную информацию смотрите в разделе “Использование непосредственных сеансов с серверами” на стр. 168). Например, если определен псевдоним DEPT для таблицы источника данных DB2MVS1.PERSON.DEPT, разрешен оператор SELECT * FROM DEPT, а оператор SELECT * FROM DB2MVS1.PERSON.DEPT не разрешен.

Работа с псевдонимами и объектами источников данных

Большинство команд утилит (LOAD, IMPORT, EXPORT, REORGCHK, REORGANIZE TABLE) не поддерживают псевдонимы.

Оператор COMMENT ON поддерживается для псевдонимов; он обновляет системный каталог на базе данных объединения.

Операции вставки, изменения и удаления не поддерживаются для псевдонимов.

Идентификация существующих псевдонимов и источников данных

Когда создано некоторое количество псевдонимов, следующая информация поможет вам узнать, какому источнику данных соответствует конкретный псевдоним, или узнать все псевдонимы для конкретного источника данных.

Как узнать источник данных для псевдонима

В этом примере предполагается, что вы знаете псевдоним (*PAYROLL*) и знаете, кто его создал (*ACCTG*), но вам нужна дополнительная информация об этом источнике данных. Сначала используйте следующий оператор SQL, чтобы получить информацию о том, под каким именем *PAYROLL* известен на своем источнике данных (SERVER).

```
select option, setting
       from syscat.taboptions
  where tabname = 'PAYROLL'
     and tabschema = 'ACCTG'
     and option in ('SERVER','REMOTE_SCHEMA','REMOTE_TABLE');
```

Набор ответов этого оператора - `DB2_MVS, FINANCE, DEPTJ35_PAYROLL`. Теперь вы знаете, что *PAYROLL* - псевдоним принадлежащей FINANCE таблицы

DEPTJ35_PAYROLL на сервере DB2_MVS. Эту информацию можно использовать в следующем операторе SELECT:

```
select option,setting
    from syscat.serveroptions
    where servername = 'DB2_MVS'
       and option in ('NODE','DBNAME');
```

Набор ответов этого оператора - REGIONW и DB2MVSDB3. Теперь вы знаете, что таблица DEPTJ35_PAYROLL находится в базе данных DB2MVSDB3 на узле REGIONW.

Используя эту информацию можно с помощью команды LIST NODE DIRECTORY получить информацию об узле REGIONW (например, узнать используемые протокол связи и тип защиты). Если этот узел используется источником данных, не относящимся к семейству DB2, для получения аналогичной информации нужно будет обратиться к файлам конфигурации этого источника данных. Например, если этот узел относится к источнику данных Oracle, подобную информацию можно получить из файла Oracle tnsnames.ora.

Подробную информацию о производных таблицах системного каталога смотрите в руководстве *SQL Reference*.

Как узнать все псевдонимы, известные DB2

Следующий оператор SQL выводит список всех псевдонимов, известных на базе данных объединения (для каждого псевдонима также выводятся имя схемы и удаленный сервер).

```
select tabname,tabschema, setting as remote_server
   from syscat.taboptions
   where option = 'SERVER';
```

Создание индекса, расширения индекса или спецификации 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, используемый с индексами для столбцов 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. При создании псевдонима спецификация индекса генерируется, если существующий для базовой таблицы источника данных индекс имеет формат, распознаваемый 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. Дополнительную информацию о том, когда может потребоваться индекс или спецификация индекса, смотрите в руководстве *Administration Guide: Performance*.
- 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.

Длина столбца, входящий в ключ индекса, не может быть больше 255 байт.

Примечание: Переменная реестра DB2_INDEX_2BYTEVARLEN дает разрешение включать в состав ключа индекса столбцы, длина которых превышает 255.

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 порядок столбцов в ключе индекса не имеет значения при создании ключа индекса, он может влиять выбор использования индекса оптимизатором.

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 разрешить слияние или реорганизацию этих конечных страниц в рабочем режиме. Оперативная реорганизация индекса используется для предотвращения ситуаций, когда после выполнения большого числа операций удаления и вставки на многих конечных страницах индекса остается лишь небольшое число ключей индекса. В такой ситуации, если не используется оперативная реорганизация, для восстановления неиспользуемого пространства потребуется выполнение реорганизации данных и индекса в автономном режиме. Принимая при создании индекса решение, разрешить ли оперативную реорганизацию его страниц, нужно ответить на следующий вопрос: Не превышают ли затраты на проверку необходимости слияния пространства при каждом удалении ключей и затраты на выполнение слияния в случаях, когда для этого имеется достаточно пространства, выигрыша от лучшего использования пространства индекса? Не лучше ли выполнять реорганизацию в автономном режиме?

Примечание: Страницы, освобождаемые при оперативной реорганизации, могут повторно использоваться для других индексов в этой же таблице. 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, при реорганизации в рабочем режиме не освобождаются промежуточные страницы индекса, а при полной реорганизации индекс делается как можно более компактным (уменьшается число промежуточных и конечных страниц, а также число уровней индекса).

Дополнительную информацию о реализации индекса, который будет реорганизовываться оперативно, смотрите в разделе “Использование оператора CREATE INDEX” на стр. 176.

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.

Совет по улучшению производительности: Если вы собираетесь выполнить следующий набор задач:

1. Создать таблицу
2. Загрузить таблицы
3. Создать индекс

4. Выполнить команду RUNSTATS

то эти задачи лучше выполнять в следующем порядке:

1. Создать таблицу
2. Создать индекс
3. Загрузить таблицу, используя опцию `statistics yes`.

Дополнительную информацию об улучшении производительности операции LOAD смотрите в руководстве *Data Movement Utilities Guide and Reference*.

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. 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, используемое для хранения данных индексов, определяется при создании таблицы (смотрите раздел “Создание таблицы в нескольких табличных пространствах” на стр. 140).

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>)
```

В следующих двух разделах, “Использование индексов” на стр. 176 и “Использование оператора CREATE INDEX” на стр. 176, содержится дополнительная информация о создании индексов.

Использование индексов

Индекс никогда не используется прикладной программой напрямую. Решение, использовать ли индекс и если да, какой из возможных индексов использовать, принимает оптимизатор.

Лучше всего использовать для таблицы индекс, который:

- Использует скоростные диски
- Сильно кластеризован
- Состоит из небольшого числа столбцов

Подробное описание возможных преимуществ использования индексов смотрите в главе “Принципы просмотра индексов” руководства *Administration Guide: Performance*.

Использование 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. Информацию о том, что требуется для создания типа индекса, смотрите в руководстве *SQL Reference*.

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. Дополнительную общую информацию об операторе ALTER TABLE смотрите в разделе “Изменение атрибутов таблиц” на стр. 213. Дополнительную подробную информацию об операторе ALTER TABLE смотрите в руководстве *SQL Reference*.

То же самое относится и к строкам, размер которых был увеличен операцией UPDATE и которые не умещаются на старое место.

Условие MINPCTUSED оператора CREATE INDEX задает порог минимального количества используемого пространства на конечных страницах индекса. Если использовано это условие, для индекса разрешается оперативная реорганизация. Если реорганизация разрешена, следующие условия определяют, будет ли она выполняться: Если после удаления ключа из конечной страницы индекса процент используемого пространства на этой странице меньше задаваемого значением порога, проверяются соседние конечные страницы индекса, чтобы определить, можно ли разместить ключи из этих двух конечных страниц на одной конечной странице индекса.

Например, следующий оператор SQL создает индекс, для которого разрешена оперативная реорганизация:

```
CREATE INDEX LASTN ON EMPLOYEE (LASTNAME) MINPCTUSED=20
```

Если после удаления ключа из этого индекса оставшиеся ключи на конечной странице занимают 20 или менее процентов пространства этой конечной страницы, предпринимается попытка удалить эту страницу индекса, разместив ее ключи на одной из соседних страниц индекса. Если ключи из этих двух страниц умещаются на одной странице, выполняется слияние страниц и одна страница индекса удаляется.

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.

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. Дополнительную информацию о реплицируемых сводных таблицах смотрите в руководстве *Administration Guide: Planning*.

В случае внутираздельного 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. Если параметр конфигурации базы данных *indexsort* имеет значение NO, для операции создания индекса не будет использоваться несколько процессоров.

В многораздельных базах данных уникальные индексы должны определяться как надмножества ключа разделения.

Создание пользовательских расширенных типов индексов

Для поддержки пользовательских типов индексов DB2 Universal Database позволяет создавать и применять свои собственные алгоритмы для основных компонентов, которые определяют работу 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 включенных (INCLUDE) столбцов

Подробности поддержки 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.

Подробности поиска 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.

Подробности применения 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 использоваться 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).

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. Когда известен примерный процент строк, которые может возвращать предикат, в вызове функции можно использовать опцию SELECTIVITY (избирательность), помогающую оптимизатору DB2 в выборе более эффективного пути 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`.

Глава 4. 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 характеристик логической и физической структуры”
- “Изменение лицензионной информации”
- “Изменение экземпляров”
- “Изменение переменных среды и переменных реестра профиля” на стр. 189
- “Изменение файла конфигурации узлов” на стр. 189
- “Изменение конфигурации базы данных” на стр. 189

Изменение характеристик логической и физической структуры

Перед тем, как вносить изменения, влияющие на всю базу данных, следует проверить все решения, касающиеся логической и физической структуры базы данных. Например, при изменении табличного пространства следует проверить выбор используемого типа хранения - SMS или DMS. (Дополнительную информацию смотрите в руководстве *Administration Guide: Planning*.)

Изменение лицензионной информации

В процессе управления лицензиями на продукты DB2 вы можете обнаружить, что нужно увеличить число лицензий. Чтобы проверить использование установленных продуктов и соответственно увеличить число лицензий, можно использовать Центр лицензий, входящий в Центр управления.

Изменение экземпляров

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 DB2 обновляется с помощью временного исправления программы (PTF) или исправления, нужно обновить все существующие экземпляры DB2, используя команду **db2iupdt**. Нужно также обновить сервер администратора, используя команду **dasiupdt**.

Прежде, чем пытаться изменить или удалить экземпляр, нужно узнать информацию об экземплярах и серверах разделов баз данных в экземплярах.

Список экземпляров

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

1. Раскройте дерево объектов и найдите папку **Базы данных**.
2. Щелкните правой кнопкой мыши по базе данных, для которой нужно вывести список экземпляров, и выберите из всплывающего меню пункт **Добавить**.
3. Нажмите кнопку **Обновить** и щелкните по стрелке в конце поля **Имя базы данных**, чтобы увидеть список экземпляров.
4. Нажмите кнопку **Отмена**.

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

```
db2ilist
```

Чтобы узнать, какой экземпляр используется в текущем сеансе (в OS/2 или на поддерживаемых платформах Windows), используйте команду:

```
set db2instance
```

Изменение конфигурации экземпляра

При выполнении команды **db2iupdt** указанный экземпляр изменяется 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.

Команда **db2iupdt** находится в подкаталоге `instance` подкаталога для данной версии и выпуска продукта (точное имя зависит от конкретной операционной системы).

Эта команда используется так:

```
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. Допустимые типы аутентификации: SERVER, CLIENT, DCS и DCE. Если этот параметр не задан, при установке сервера DB2 по умолчанию задается тип аутентификации SERVER. В противном случае задается тип аутентификации CLIENT. Тип аутентификации экземпляра применяется для всех баз данных этого экземпляра.

  В операционных системах UNIX нельзя задавать тип аутентификации DCE.

- –e

  Позволяет обновить все существующие экземпляры. Их можно увидеть с помощью команды **db2ilist**.

- –u FencedID

  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 FencedID нельзя использовать “root” или “bin”.

- –k

  Это параметр сохраняет текущий тип экземпляра. Если этот параметр не задан, тип текущего экземпляра обновляется обновляется до наивысшего 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 Edition)

  – Клиент (тип экземпляра по умолчанию для клиента DB2)

Примеры:

- Если после создания экземпляра установлена система DB2 Universal Database Workgroup Edition или DB2 Universal Database Enterprise Edition, введите следующую команду для обновления этого 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 использовать команду:

  ```
  db2iupdt db2inst1
  ```

Удаление экземпляров

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.

Изменение переменных среды и переменных реестра профиля

Необходимо определить, нужно ли изменить в вашей среде переменные среды и если да, то какие. Если были изменены какие-либо переменные среды, нужно (кроме систем UNIX) перезагрузить систему, чтобы новые переменные среды вступили в силу. Проверьте, не нужно ли перед изменением базы данных восстановить значения переменных реестра профилей в глобальном реестре профиля. Затем можно задать для переменных реестра наиболее подходящие значения для новой среды базы данных. Если были изменены только переменные реестра профиля, перезагрузка системы не требуется.

Изменение файла конфигурации узлов

Если вы планируете изменить какие-либо группы узлов (добавляя или удаляя узлы или перемещая существующие узлы), посмотрите в главе "Масштабирование конфигурации при добавлении процессоров" руководства *Administration Guide: Performance* подробное описание действий.

Изменение конфигурации базы данных

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 используйте мастер по настройке производительности Центра управления. 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**.

Чтобы изменить конфигурацию менеджера баз данных из командной строки, введите команду:

```
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.

Подробную информацию о том, как улучшить производительность и задать конфигурацию системы, смотрите в разделах “Измерение производительности” и “Конфигурирование DB2” руководства *Administration Guide: Performance*.

Для многораздельной базы данных файлы конфигурации на всех разделах базы данных должны быть одинаковы. 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 синхронизации файлов конфигурации на всех разделах базы данных используйте команду **db2_all**.

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 узлов" на стр. 192
- "Изменение табличного пространства" на стр. 192
- "Отбрасывание схемы" на стр. 199
- "Изменение структуры и содержимого таблицы" на стр. 199
- "Изменение пользовательского структурированного типа" на стр. 217
- "Удаление или изменение строк в типизированной таблице" на стр. 217
- "Переименование существующей таблицы" на стр. 217
- "Отбрасывание таблицы" на стр. 218
- "Отбрасывание триггера" на стр. 220
- "Отбрасывание пользовательской функции, отображения типов или метода" на стр. 221
- "Отбрасывание пользовательского типа или отображения типов" на стр. 222
- "Изменение или отбрасывание производной таблицы" на стр. 222
- "Отбрасывание сводной таблицы" на стр. 224
- "Изменение или отбрасывание сервера" на стр. 226
- "Изменение или отбрасывание псевдонима" на стр. 227
- "Отбрасывание индекса, расширения индекса или спецификации индекса" на стр. 228
- "Зависимости операторов при изменении объектов" на стр. 230

Отбрасывание базы данных

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.

Изменение группы узлов

Подробную информацию об изменении группы узлов можно найти в главе “Масштабирование конфигурации при добавлении процессоров” руководства *Administration Guide: Performance*.

После добавления или удаления узлов необходимо перераспределить текущие данные по новому набору узлов в этой группе узлов. Используйте для этого команду REDISTRIBUTE NODEGROUP. Информацию об этом смотрите в главе “Перераспределение данных по разделам базы данных” руководства *Administration Guide: Performance* и в руководстве *Command Reference*.

Изменение табличного пространства

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: Нельзя отбрасывать табличное пространство каталогов SYSCATSPACE или создавать его заново и всегда должно быть по крайней мере одно системное временное табличное пространство. Вы можете создать другие системные временные табличные пространства. Кроме того, нельзя изменить размер страниц или размер экстента табличного пространства после его создания.

В этом разделе описываются способы изменения табличных пространств:

- “Добавление контейнера к табличному пространству DMS”
- “Изменение контейнеров в табличном пространстве DMS” на стр. 194
- “Добавление к табличному пространству SMS контейнера на разделе” на стр. 195
- “Переименование табличного пространства” на стр. 196
- “Переключение состояния табличного пространства” на стр. 196
- “Отбрасывание пользовательского табличного пространства” на стр. 197
- “Отбрасывание системного временного табличного пространства” на стр. 197.

Информацию о проектировании табличных пространств смотрите в руководстве *Administration Guide: Planning*.

Добавление контейнера к табличному пространству DMS

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.

Содержимое табличного пространства равномерно распределяется по всем контейнерам. В процессе этого 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**.
4. Если это табличное пространство находится в среде многораздельных баз данных и для него нужно изменить параметры производительности, нажмите кнопку **Дополнительные**.
5. Нажмите кнопку **OK**.

Чтобы добавить контейнер к табличному пространству DMS из командной строки, введите команду:

```
ALTER TABLESPACE <имя>
   ADD (DEVICE '<путь>' <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. Дополнительную информацию смотрите в главе “Влияние табличного пространства на оптимизацию запросов” руководства *Administration Guide: Performance*.

Изменение контейнеров в табличном пространстве DMS

Можно увеличить размер контейнеров в табличном пространстве DMS (то есть табличном пространстве, при создании которого задано условие MANAGED BY DATABASE), изменив размер одного или нескольких контейнеров или расширив один или несколько контейнеров, связанных с этим табличным пространством. Методом изменения размера следует пользоваться, если известен новый верхний предел размера контейнера. Методом расширения следует пользоваться, если вы не знаете (или вас не интересует) текущий размер контейнера.

Чтобы изменить размер одного или нескольких контейнеров в табличном пространстве DMS из командной строки, введите команду:

```
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. Так же, как при добавлении новых контейнеров, содержимое табличного пространства равномерно распределяется по всем контейнерам. 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 (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
  EXTEND (DEVICE '/dev/rhd11'  1000,
          DEVICE  '/dev/rhd12' 1000)
```

После этой операции размер этих двух устройств будет увеличен с 1000 до 2000 страниц. Так же, как при добавлении новых контейнеров, содержимое табличного пространства равномерно распределяется по всем контейнерам. 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.

Примечание: Размер контейнеров нельзя уменьшить.

Учтите, что оператор ALTER TABLESPACE позволяет изменить другие свойства табличного пространства, которые могут влиять на производительность. Дополнительную информацию смотрите в главе “Влияние табличного пространства на оптимизацию запросов” руководства *Administration Guide: Performance*.

Добавление к табличному пространству SMS контейнера на разделе

Чтобы добавить контейнер к табличному пространству SMS с помощью командной строки, введите команду:

```
ALTER TABLESPACE <имя>
ADD ('<путь>')
ON NODE (<номер_раздела>)
```

Раздел задается его номером и каждый раздел (или узел) в этом диапазоне разделов должен существовать в группе узлов, в которой определено это

| табличное пространство. номер_раздела может задавать только явно или
| внутри ровно одного *условия ON NODES* для 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)
```

Переименование табличного пространства

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.

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.

Переключение состояния табличного пространства

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:

```
ALTER TABLESPACE <имя>
   SWITCH ONLINE
```

Отбрасывание пользовательского табличного пространства

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. Подробную информацию о том, как отбросить табличные пространства, содержащие данные таблиц, занимающих несколько табличных пространств, смотрите в разделе *SQL Reference*.

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
```

Отбрасывание системного временного табличного пространства

Нельзя отбросить системное временное табличное пространство, не создав сначала другое системное временное табличное пространство, поскольку база данных всегда должна содержать по крайней мере одно такое табличное пространство. Например, чтобы добавить контейнер к временному табличному

пространству SMS, необходимо сначала создать новое системное временное табличное пространство и затем отбросить старое системное временное табличное пространство.

Чтобы отбросить системное табличное пространство с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Табличные пространства**.
2. Если у вас только одно системное временное табличное пространство, щелкните правой кнопкой мыши по папке **Табличные пространства** и выберите из всплывающего меню **Создать —> Табличное пространство при помощи 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**.

Если у вас только одно системное временное табличное пространство, перед его удалением необходимо создать другое такое табличное пространство. Это можно сделать из командной строки, введя команду:

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

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')
```

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.

Отбрасывание пользовательского временного табличного пространства

Пользовательское временное табличное пространство можно отбросить, только если в нем в настоящее время 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.

Отбрасывание схемы

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.

Изменение структуры и содержимого таблицы

Задачи по изменению структуры и содержимого таблицы:

- “Добавление столбцов в существующую таблицу” на стр. 200
- “Изменение определения столбца” на стр. 201
- “Удаление строк из таблицы или производной таблицы” на стр. 201
- “Изменение ограничения” на стр. 203
- “Определение генерируемых столбцов в существующих таблицах” на стр. 208
- “Объявление таблицы нестабильного объема” на стр. 212
- “Изменение ключей разделения” на стр. 213
- “Изменение атрибутов таблиц” на стр. 213

- “Обновление данных в сводной таблице” на стр. 216

Учтите, что нельзя изменять триггеры для таблиц; необходимо отбросить не соответствующие задачам триггеры (смотрите раздел “Отбрасывание триггера” на стр. 220) и создать вместо них новые (смотрите раздел “Создание триггера” на стр. 143).

Добавление столбцов в существующую таблицу

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 столбца 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)
```

Изменение определения столбца

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
```

Дополнительную информацию об операторе ALTER TABLE смотрите в руководстве *SQL Reference*.

Удаление строк из таблицы или производной таблицы

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. Дополнительная информация о том, что происходит со строками, зависимыми от выбранных строк, приводится в справочнике *SQL Reference*.

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.

Изменение определения столбца идентификации

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.

Изменение ограничения

Чтобы изменить ограничения, надо отбросить их и создать вместо них новые. Дополнительную информацию смотрите в разделах:

- “Добавление ограничения”
- “Отбрасывание ограничения” на стр. 206

Дополнительную информацию об ограничениях смотрите в разделе “Определение ограничений” на стр. 128.

Добавление ограничения

Для добавления ограничений используется оператор ALTER TABLE. Дополнительную информацию об этом операторе, включая его синтаксис, смотрите в руководстве *SQL Reference*.

Дополнительную информацию об ограничениях смотрите в разделе “Определение ограничений” на стр. 128.

Добавление ограничения уникальности: 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 этому 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)
```

Добавление первичных и внешних ключей: 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>
```

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.

Смотрите раздел “Зависимости операторов при изменении объектов” на стр. 230.

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
```

Добавление проверочных ограничений таблицы: 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 этому 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. Дополнительную информацию смотрите в разделе “Зависимости операторов при изменении объектов” на стр. 230.

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)
```

Отбрасывание ограничения

Для отбрасывания ограничений используется оператор ALTER TABLE. Дополнительную информацию об этом операторе, включая его синтаксис, смотрите в руководстве *SQL Reference*.

Дополнительную информацию об ограничениях смотрите в разделе “Определение ограничений” на стр. 128.

Отбрасывание ограничений уникальности: 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.

Отбрасывание первичных и внешних ключей: 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.

Дополнительную информацию смотрите в разделе “Зависимости операторов при изменении объектов” на стр. 230.

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
```

Информацию об операторе ALTER TABLE смотрите в руководстве *SQL Reference*.

Отбрасывание проверочных ограничений таблицы: 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. (Дополнительную информацию смотрите в разделе “Зависимости операторов при изменении объектов” на стр. 230.) Имена всех проверочных ограничений таблицы можно найти в производной таблице каталога SYSCAT.CHECKS. Перед тем, как отбрасывать проверочное ограничение таблицы, имя которого сгенерировано системой, посмотрите 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
```

Определение генерируемых столбцов в существующих таблицах

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.

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. Далее, в зависимости от действий, которые нужно выполнить для этой таблицы, есть несколько путей выполнения задачи:

   - Таблица имеет очень большой размер и вы не уверены, что объема журналов достаточно для выполнения этой задачи. После загрузки данных, но перед включением проверки целостности нужно выполнить принятие:

     ```
     COMMIT
     ```

     Затем нужно использовать утилиту db2gncol, чтобы вычислить значения генерируемых столбцов. Эта утилита находится в каталоге `sqllib` в подкаталоге `bin`. Эта утилита используется так:

     ```
     db2gncol -d <имя_базы_данных> -s <схема> -t <имя_таблицы>
              -c <число_принятия>
     ```

     `имя_базы_данных` задает алиас базы данных, в которой находится таблица. `схема` задает имя схемы таблицы (этот параметр регистрозависим). `имя_таблицы` задает таблицу, для столбцов которой нужно вычислить по формулам новые значения генерируемых столбцов. Как и `схема`, `имя_таблицы` регистрозависимо. `число_принятия` - это число обрабатываемых строк перед выполнением следующей операции принятия, очищающей журналы. Этот параметр влияет на размер пространства журналов, необходимого для выполнения генерации значений столбцов.

     Существуют также два необязательных параметра, которые не показаны в предыдущем примере. Это параметры `-u <имя_пользователя>` и `-p <пароль>`, которые задают пользователя и пароль. Этот пользователь должен обладать полномочиями SYSADM или DBADM. Если пользователь и пароль не заданы, используется текущий ID пользователя.

     Чтобы получить информацию справки для этой утилиты, введите команду:

     ```
     db2gncol -h
     ```

     Если задан параметр справки, все остальные параметры игнорируются.

Несмотря на то, что таблица находится в состоянии отложенной проверки, она блокируется на все время этого процесса. Смысл блокировки в том, что к таблицам, находящимся в состоянии отложенной проверки, могут обращаться другие утилиты. Блокировка предотвращает конфликты между этими утилитами.

- Вы предполагаете, что объем журналов для обновления генерируемых столбцов достаточен для выполнения команды SET INTEGRITY. Это типичная ситуация. После загрузки данных выполните вычисление и присваивание значений генерируемым столбцам, используя команду:

  ```
  SET INTEGRITY FOR t1 IMMEDIATE CHECKED
  FORCE GENERATED
  ```

  Примечание: В этот момент можно использовать таблицы исключений.

- Таблица имеет очень большой размер, вы не уверены, что объема журналов достаточно для выполнения этой задачи, но не хотите использовать первый из описанных методов. После загрузки данных, но перед включением проверки целостности нужно:

  a. Установить монопольную блокировку этой таблицы. Это запретит все обращения к этой таблице, кроме транзакций чтения без принятия.

     ```
     LOCK TABLE t1
     ```

  b. Переведите таблицу в рабочий режим без проверки данных.

     ```
     SET INTEGRITY FOR t1 ALL IMMEDIATE UNCHECKED
     ```

  c. Обновите генерируемые столбцы, используя чередующиеся предикаты и операции принятия, чтобы предотвратить переполнение журналов.

     ```
     UPDATE t1 SET (c3, c4) = (DEFAULT, DEFAULT) WHERE <предикат>
     ```

  d. Переведите таблицу в рабочий режим с проверкой ее целостности.

     ```
     SET INTEGRITY FOR t1 OFF
     SET INTEGRITY FOR t1 IMMEDIATE CHECKED
     ```

  e. Разблокируйте таблицу, завершив транзакцию с помощью оператора принятия.

     ```
     COMMIT
     ```

- Известно, что таблица была создана с опцией NOT LOGGED INITIALLY. При это способе 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 GENERATED
     ```

  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
```

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.

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.
- Для генерируемых столбцов не разрешены уникальные индексы. Сюда относятся ограничения уникальности и первичные ключи.

Объявление таблицы нестабильного объема

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 будет 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
```

Изменение ключей разделения

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
```

Дополнительную информацию смотрите в главе об операторе ALTER TABLE руководства *SQL Reference*.

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.

Изменение атрибутов таблиц

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.

Задание большего значения PCTFREE обеспечит поддержку кластеризации на большее время, однако потребует использования большего объема дискового пространства.

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.

Изменение столбца идентификации

Для изменения атрибутов существующего столбца идентификации используйте оператор ALTER TABLE. Дополнительную информацию об этом операторе, включая его синтаксис, смотрите в справочнике *SQL Reference*.

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 точного типа. Числовая константа может представлять собой любое положительное или отрицательное значение без ненулевых цифр после десятичной запятой, которое можно присвоить столбцу идентификации.

Изменение последовательности

Для изменения атрибутов существующей последовательности используется оператор ALTER SEQUENCE. Дополнительную информацию об этом операторе, включая его синтаксис, смотрите в руководстве *SQL Reference*.

Возможны следующие изменения атрибутов последовательности:

- Изменение приращения между будущими значениями
- Установка новых минимальных или максимальных значений
- Изменение числа кэшируемых членов последовательности
- Установка или отмена циклического повторения последовательности
- Установка или отмена генерации членов последовательности по запросу
- 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 ее создании.
- 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.

Отбрасывание последовательности

Чтобы удалить последовательность, используйте оператор DROP. Дополнительную информацию об этом операторе, включая его синтаксис, смотрите в справочнике *SQL Reference*.

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.

Изменение свойств сводной таблицы

С некоторыми ограничениями можно преобразовать сводную таблицу в обычную или обычную таблицу в сводную. Другие типы таблиц нельзя преобразовать; преобразовать можно только обычные и сводные таблицы. Например, нельзя преобразовать реплицируемую сводную таблицу в обычную таблицу или наоборот.

После преобразования обычной таблицы в сводную она переводится в состояние отложенной проверки. При таком изменении полная выборка в определении сводной таблицы должна совпадать с определением оригинальной таблицы, то есть:

- Должно совпадать число столбцов.
- Должны совпадать имена и позиции столбцов.
- Должны совпадать типы данных.

Если для исходной таблицы определена сводная таблица, саму эту исходную таблицу нельзя преобразовать в сводную таблицу. Если исходная таблица имеет триггеры, проверочные ограничения, реляционные ограничения или определенные уникальные индексы, ее нельзя преобразовать в сводную таблицу. Изменяя свойства таблицы, чтобы определить сводную таблицу, нельзя в этом же операторе ALTER TABLE еще как-то изменять эту таблицу.

При преобразовании обычной таблицы в сводную полная выборка в определении сводной таблицы не должна ссылаться на исходную таблицу прямо или косвенно через производные таблицы, алиасы или сводные таблицы.

Чтобы изменить сводную таблицу в обычную, используйте оператор:

```
ALTER TABLE сводная_таблица
   SET SUMMARY AS DEFINITION ONLY
```

Чтобы изменить обычную таблицу в сводную, используйте оператор:

```
ALTER TABLE обычная_таблица
   SET SUMMARY AS <полная_выборка>
```

Ограничения на полную выборку при преобразовании обычной таблицы в сводную очень близки к ограничениям при создании сводной таблицы с помощью оператора CREATE SUMMARY TABLE.

Дополнительную информацию об операторе CREATE SUMMARY TABLE смотрите в руководстве *SQL Reference*.

Обновление данных в сводной таблице

С помощью оператора REFRESH TABLE можно обновить данные в одной или нескольких сводных таблицах. 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.

В следующем примере показано, как обновить данные в сводной таблице:

```
REFRESH TABLE SUMTAB1
```

Дополнительную информацию об операторе REFRESH TABLE смотрите в руководстве *SQL Reference*.

Изменение пользовательского структурированного типа

После создания структурированного типа может потребоваться добавление или удаление атрибутов этого структурированного типа. Для этого используется оператор ALTER TYPE (структурированный). Всю необходимую информацию о структурированных типах можно найти в руководстве *Application Development Guide*.

Удаление или изменение 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. Всю необходимую информацию о типизированных таблицах можно найти в руководстве *Application Development Guide*.

Переименование существующей таблицы

Существующей таблице можно дать новое имя в схеме, сохранив полномочия и индексы, созданные для исходной таблицы.

Существующая таблица, которую нужно переименовать, может задаваться алиасом. Задаваемое имя таблицы, которую нужно переименовать, не должно быть именем таблицы каталога, сводной таблицы, типизированной таблицы или какого-либо иного объекта, кроме таблицы или алиаса.

На эту существующую таблицу не должны ссылаться:

- Производные таблицы
- Триггеры
- Реляционные связи
- Сводная таблица
- 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, чтобы убедиться, что переименуемая таблица не нарушает каких-либо их этих ограничений.

Для пакетов, ссылающихся на только что переименованную таблицу, необходимо заново выполнить связывание. Повторное связывание этих пакетов может быть выполнено неявно, если:

- Другая таблица переименована с использованием старого имени этой таблицы или
- Создан алиас или производная таблица с использованием старого имени этой таблицы.

Перед попыткой выполнения неявного или явного повторного связывания необходимо выполнить одно из этих двух действий. Если ни одно из них не выполнено, повторное связывание завершится с ошибкой.

Чтобы переименовать существующую таблицу с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой по таблице, которую нужно переименовать, и выберите из всплывающего меню пункт **Переименовать**.
3. Введите новое имя 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
```

Дополнительную информацию об операторе RENAME TABLE смотрите в руководстве *SQL Reference*.

Отбрасывание таблицы

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. (Дополнительную информацию смотрите в разделе “Восстановление неработоспособных производных таблиц” на стр. 224.)
- 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. (Дополнительную информацию смотрите в разделе “Зависимости операторов при изменении объектов” на стр. 230.)
- 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.

Дополнительную информацию об операторе DROP смотрите в руководстве *SQL Reference*.

Отбрасывание пользовательской временной таблицы

Есть некоторые особенности, которые нужно учитывать при отбрасывании пользовательской временной таблицы (то есть таблицы, созданной с помощью оператора DECLARE GLOBAL TEMPORARY TABLE).

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.

Пакеты не могут зависеть от этого типа таблиц, поэтому при отбрасывании такой таблицы не возникает недействительных пакетов.

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.

Дополнительную информацию об операторе DROP смотрите в руководстве *SQL Reference*.

Отбрасывание триггера

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.

Отбрасывание пользовательской функции, отображения типов или метода

Пользовательскую функцию, шаблон функции или отображение функции можно отбросить с помощью оператора DROP.

Отображение функции можно отключить с помощью опции отображения DISABLE. Дополнительную информацию о том, как это сделать, смотрите в руководстве *SQL Reference*.

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.

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). Для таких пакетов не выполняется неявное связывание. Нужно выполнить для них связывание, используя команды BIND или REBIND, или же подготовить их, используя команду PREP. Дополнительную информацию об этих командах смотрите в руководстве *Command Reference*. 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.

(Дополнительную информацию смотрите в разделе “Зависимости операторов при изменении объектов” на стр. 230.)

Отбрасывание пользовательского типа или отображения типов

С помощью оператора DROP можно отбросить пользовательский тип или отображение типов. 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.

Если для пользовательского типа создано преобразование и вы собираетесь отбросить этот тип, возможно, следует отбросить это преобразование. Для этого используется оператор DROP TRANSFORM. Подробное описание этого оператора смотрите в руководстве *SQL Reference*. Учтите, что можно отбросить только преобразования, определенные вами или другими разработчиками прикладных программ; встроенные преобразования и связанные с ними определения групп нельзя отбросить.

Дополнительную информацию о пользовательских типах смотрите в руководствах *SQL Reference* и *Application Development Guide*.

Изменение или отбрасывание производной таблицы

С помощью оператора ALTER VIEW можно изменить существующую производную таблицу, изменив столбец ссылочного типа, чтобы задать для него область видимости. Для внесения в производную таблицу любых других изменений необходимо ее отбросить и затем создать заново.

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.

Столбец, имя которого задано в операторе ALTER VIEW, должен иметь тип данных REF (тип, содержащий имя типизированной таблицы или имя типизированной 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. Дополнительную информацию смотрите в разделе “Зависимости операторов при изменении объектов” на стр. 230.

Дополнительную информацию об операторе ALTER VIEW смотрите в руководстве *SQL Reference*.

Чтобы изменить производную таблицу 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 отбрасываемой производной таблицы, становятся неработоспособными. (Дополнительную информацию смотрите в разделе “Восстановление неработоспособных производных таблиц” на стр. 224.)

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
```

Дополнительную информацию об отбрасывании и создании производных таблиц смотрите в руководстве *SQL Reference*.

Восстановление неработоспособных производных таблиц

Производные таблицы могут стать *непригодными к использованию*:

- В результате отзыва привилегии для базовой таблицы
- Если отброшена таблица, алиас или функция
- Если стала неработоспособной производная надтаблица
- 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.

Отбрасывание сводной таблицы

Сводную таблицу нельзя изменить, но ее 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. Дополнительную

информацию о зависимостях пакетов смотрите в разделе “Зависимости операторов при изменении объектов” на стр. 230.

Чтобы отбросить сводную таблицу с помощью Центра управления:

1. Раскройте дерево объектов и найдите папку **Таблицы**.
2. Щелкните правой кнопкой мыши по сводной таблице, которую нужно отбросить, и выберите из всплывающего меню пункт **Отбросить**.
3. Включите переключатель **Подтверждение** и нажмите кнопку **OK**.

Чтобы отбросить сводную таблицу из командной строки, введите команду:

```
DROP TABLE <имя_таблицы>
```

Следующий оператор SQL отбрасывает сводную таблицу XT:

```
DROP TABLE XT
```

Восстановление неработоспособных сводных таблиц

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.

Отбрасывание оболочки

Для удаления оболочки из базы данных можно использовать оператор DROP. В следующем примере показано, как отбросить оболочку DRDA:

```
DROP WRAPPER DRDA
```

Отбрасываются все определения серверов, отображения пользовательских функций и отображения пользовательских типов данных, зависящие от оболочки. Отбрасываются также все пользовательские отображения, псевдонимы, отображения пользовательских типов и отображения пользователей, зависимые от отброшенных определений серверов. Отбрасываются все спецификации индекса, зависимые от отброшенных псевдонимов, а все производные таблицы, зависимые от этих псевдонимов, помечаются как непригодные к использованию. Все пакеты, зависимые от отброшенных объектов и непригодных к использованию производных таблиц, объявляются недействительными.

Для отбрасывания оболочки необходимо обладать полномочиями SYSADM или DBADM.

Дополнительную информацию об отбрасывании оболочек смотрите в руководстве *SQL Reference*.

Изменение или отбрасывание сервера

Оператор ALTER SERVER изменяет существующее определение сервера в каталоге базы данных объединения. Используйте этот оператор, чтобы:

- Изменить определение конкретного источника данных.
- Изменить определение нескольких источников данных конкретного типа или версии.
- Внести изменения в конфигурацию конкретного источника данных. Например, если СУБД конкретного сервера перенесена на новую рабочую станцию с более быстрым процессором, нужно изменить значение опции cpu_ratio.

Этот оператор нельзя использовать для изменения опций сервера *dbname* или *node*.

В следующем примере показано, как изменить сервер ORA1:

```
ALTER SERVER ORA1 OPTIONS (SET CPU_RATIO '5.0')
```

Серверы можно отбросить из базы данных объединения. В следующем примере показано, как отбросить сервер ORALOC01:

```
DROP SERVER ORALOC01
```

Отбрасываются все псевдонимы для таблиц и производных таблиц, находящихся на источнике данных. Отбрасываются все спецификации индексов, зависимые от этих псевдонимов. Отбрасываются также все отображения пользовательских функций, отображения пользовательских типов и отображения пользователей, зависимые от отброшенных определений серверов.

Все пакеты, зависимые от отброшенного определения сервера, отображений функций, псевдонимов и спецификаций индексов, объявляются недействительными.

Для изменения или отбрасывания сервера необходимо обладать полномочиями SYSADM или DBADM.

Дополнительную информацию об отбрасывании и изменении серверов смотрите в руководстве *SQL Reference*.

Изменение или отбрасывание псевдонима

Для изменения хранящейся локально информации о таблице или производной таблице источника данных используется оператор ALTER NICKNAME. Например, этот оператор можно использовать, чтобы изменить локальное имя для столбца или отобразить тип данных столбца на другой тип данных. Этот оператор можно также использовать для добавления опций столбцов. Дополнительную информацию о синтаксисе оператора ALTER NICKNAME смотрите в руководстве *SQL Reference*.

При отбрасывании псевдонима созданные на его основе производные таблицы отмечаются как неработоспособные. Если псевдоним используется в производной таблице, нельзя изменить его имена столбцов или типы данных.

Чтобы использовать этот оператор, необходимо обладать полномочиями SYSADM или DBADM, иметь привилегию базы данных CONTROL или ALL для этого псевдонима, привилегию схемы ALTERIN (для текущей схемы) или быть указанным, как определяющий этого псевдонима в базе данных объединения.

Изменение столбцов псевдонимов и отбрасывание псевдонимов

В следующем примере показано, как изменить псевдоним TESTNN, поменяв локальное имя столбца с COL1 на NEWCOL:

```
ALTER NICKNAME TESTNN ALTER COLUMN COL1 LOCAL NAME NEWCOL
```

В следующем примере показано, как отбросить псевдоним TESTNN:

```
DROP NICKNAME TESTNN
```

Изменение опций столбцов псевдонима

Информация о столбцах задается в виде значений, присваиваемых параметрам, называемым *опциями столбцов*. Все эти значения можно задавать в верхнем или в нижнем регистре. Описание этих значений и дополнительную информацию смотрите в следующей таблице.

Таблица 3. Опции столбцов и их значения

Опция	Допустимые значения		Значение по умолчанию
numeric_string	'Y'	Этот столбец содержит только строки числовых данных. ВНИМАНИЕ: Если этот столбец содержит только числовые строки, дополненные хвостовыми пробелами, не рекомендуется задавать значение 'Y'.	'N'
	'N'	Этот столбец может содержать не только строки числовых данных.	
	Задав для опции столбца numeric_string значение 'Y', вы указываете оптимизатору, что этот столбец не содержит пробелов, которые могут повлиять на сортировку данных столбца. Эта опция полезна, если последовательность слияния источника данных отличается от последовательности слияния DB2. Столбцы, для которых задана эта опция, не будут исключаться из локальных (на источнике данных) вычислений при несовпадении последовательностей слияния.		
varchar_no_trailing_blanks	Указывает отсутствие хвостовых пробелов для конкретного столбца VARCHAR:		'N'
	'Y'	Этот столбец VARCHAR не содержит хвостовых пробелов.	
	'N'	Этот столбец VARCHAR содержит хвостовые пробелы.	
	Если столбец источника данных с типом VARCHAR не содержит дополнительных пробелов, выбираемая оптимизатором стратегия доступа к этому столбцу несколько отличается в зависимости от того, содержит ли столбец хвостовые пробелы. По умолчанию оптимизатор "предполагает", что этот столбец в действительности содержит хвостовые пробелы. Исходя из этого предположения, оптимизатор разрабатывает стратегию доступа, включающую изменение запросов, чтобы возвращаемые значения этого столбца соответствовали ожидаемым пользователем. Однако если столбец VARCHAR не содержит хвостовых пробелов и вы сообщите это оптимизатору, он сможет разработать более эффективную стратегию доступа. Чтобы сообщить оптимизатору, что конкретный столбец не содержит хвостовых пробелов, задайте этот столбец в операторе ALTER NICKNAME (его синтаксис смотрите в руководстве *SQL Reference*).		

Отбрасывание индекса, расширения индекса или спецификации индекса

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.)

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
```

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.

Все пакеты и кэшируемые динамические операторы SQL, зависящие от отброшенных индексов, отмечаются как недействительные. Дополнительную

информацию смотрите в разделе “Зависимости операторов при изменении объектов”. Изменения, вызванные добавлением или отбрасыванием индексов, не влияют на прикладные программы.

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. Подробное описание создания пакетов смотрите в руководстве *Application Development Guide*.

Пакеты и кэшируемые динамические операторы SQL могут зависеть от многих типов объектов. Полный список таких объектов смотрите в руководстве *SQL Reference*.

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 отброшен, этот пакет или кэшируемый динамический оператор SQL переводится в состояние ″недействительный″. Если пакет зависит от пользовательской функции и эта функция отброшена, этот пакет переводится в состояние ″неработоспособный″.

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.

Дополнительную информацию о связывании и повторном связывании пакетов смотрите в руководстве *Application Development Guide*.

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, повторное связывание пакета невозможно. 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

Дополнительную информацию о зависимостях объектов смотрите в разделе об операторе DROP руководства *SQL Reference*.

Часть 3. Защита баз данных

Глава 5. 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 6. Аудит действий DB2” на стр. 291.

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. В руководствах *Quick Beginnings* (Быстрый старт) для конкретных платформ содержится информация о планировании, установке и настройке DB2.

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:

OS/2	Всем допустимым ID пользователей DB2 из групп администраторов UPM (User Profile Management - управление профилями пользователей) и локальных администраторов.
Windows 95 или Windows 98	Всем пользователям Windows 95 или Windows 98.
Windows NT или Windows 2000	Всем пользователям DB2 из группы администраторов.
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.

Информация о создании и назначении групп и ID пользователей находится в книгах *Быстрый старт* для конкретных платформ. 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.

Должна существовать группа, определенная как группа управления системой (полученная путем изменения SYSADM_GROUP). 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.

Особенности платформы Windows NT

При работе в Enterprise – Extended Edition для Windows NT полномочия управления системой (SYSADM) предоставляются всякой учетной записи пользователя, принадлежащей локальной группе администраторов на том компьютере, где определена учетная запись.

Например, если пользователь входит в систему по учетной записи домена и пытается обратиться к базе данных DB2, DB2 обратится к контроллеру домена, чтобы получить списки групп (включая группу администратора). Это поведение DB2 можно изменить следующими двумя способами:

1. Задать переменную реестра DB2_GRP_LOOKUP=local и добавить учетные записи домена (или глобальные группы) в локальную группу администраторов.
2. Добавить в файл конфигурации менеджера баз данных новую группу. Если вы хотите, чтобы список этой группы был на локальном компьютере, необходимо также задать переменную реестра DB2_GRP_LOOKUP.

По умолчанию в среде домена Windows NT права SYSADM на какой-либо экземпляр имеются только у тех пользователей домена, которые принадлежат

группе администраторов на первичном контроллере домена (Primary Domain Controller, PDC). 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.

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
```

Особенности платформы UNIX

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. Пользователь изолированных пользовательских функций и хранимых процедур задается как параметр сценария создания экземпляра (db2icrt ... -u<FencedID>). Этот параметр не требуется при установке клиентов DB2 и комплекта разработчика программ DB2.

Общие правила

Есть правила для именования всех объектов и пользователей. Некоторые из этих правил зависят от платформы, на которой вы работаете. Например, определенные правила регулируют использование в имени букв верхнего и нижнего регистров.

- На платформах UNIX имена должны быть в нижнем регистре.
- На платформах OS/2 имена должны быть в верхнем регистре.
- На платформах Windows имена могут быть в верхнем регистре, в нижнем регистре и смешанные.

Правила именования DB2 смотрите в разделе “Приложение A. Правила именования” на стр. 331.

Команда db2icrt создает главный каталог библиотек SQL (sqllib) в начальном каталоге владельца экземпляра.

Выбор метода аутентификации для сервера

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. Дополнительную информацию о параметре *authentication* конфигурации менеджера баз данных смотрите в разделе “Настройка DB2” в книге *Administration Guide: Performance*. 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. Дополнительную информацию смотрите в разделе “Обработка аутентификации базы данных объединения” на стр. 252.

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.

Если аутентификация типа SERVER задана на удаленном экземпляре, возможны два варианта аутентификации:

- ID и пароль пользователя задаются пользователем.
- DB2 получает ID и пароль пользователя, а затем передает их для аутентификации на сервер. (Пользователь уже зарегистрирован на локальном компьютере или в домене.)

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 аутентификация клиента - DCS или SERVER, клиент проходит аутентификацию, передавая серверу ID пользователя и пароль. Если аутентификация клиента - DCS_ENCRYPT или SERVER_ENCRYPT, клиент проходит аутентификацию, передавая ID пользователя и зашифрованный пароль.

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.

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 работающих на операционных системах Windows 95 и Windows 98.

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. Дополнительную информацию об этом параметре смотрите в разделе “Настройка DB2” в книге *Administration Guide: Performance*.

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/USING.

Чтобы защититься от всех клиентов, кроме клиентов DRDA из систем DB2 для MVS и OS/390, DB2 для VM и VSE и DB2 для OS/400, задайте для параметра *trust_allclnts* значение DRDAONLY. Только этим клиентам будет доверяться проведение аутентификации на стороне клиента. 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 4. Режимы аутентификации при использовании сочетаний параметров TRUST_ALLCLNTS и TRUST_CLNTAUTH.

TRUST_ ALLCLNTS	**TRUST_ CLNTAUTH**	**Аутентификация ненадежных не–DRDA клиентов без пароля**	**Аутентификация ненадежных не–DRDA клиентов с паролем**	**Аутентификация доверенных не–DRDA клиентов без пароля**	**Аутентификация доверенных не–DRDA клиентов с паролем**	**Аутентификация клиентов DRDA без пароля**	**Аутентификация клиентов DRDA с паролем**
YES	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT
YES	SERVER	CLIENT	SERVER	CLIENT	SERVER	CLIENT	SERVER
NO	CLIENT	SERVER	SERVER	CLIENT	CLIENT	CLIENT	CLIENT
NO	SERVER	SERVER	SERVER	CLIENT	SERVER	CLIENT	SERVER

Таблица 4. Режимы аутентификации при использовании сочетаний параметров TRUST_ALLCLNTS и TRUST_CLNTAUTH. (продолжение)

TRUST_ ALLCLNTS	TRUST_ CLNTAUTH	Аутентификация ненадежных не–DRDA клиентов без пароля	Аутентификация ненадежных не–DRDA клиентов с паролем	Аутентификация доверенных не–DRDA клиентов без пароля	Аутентификация доверенных не–DRDA клиентов с паролем	Аутентификация клиентов DRDA без пароля	Аутентификация клиентов DRDA с паролем
DRDAONLY	CLIENT	SERVER	SERVER	SERVER	SERVER	CLIENT	CLIENT
DRDAONLY	SERVER	SERVER	SERVER	SERVER	SERVER	CLIENT	SERVER

DCS Используется в основном для занесения в каталог базы данных при обращении с помощью DB2 Connect. (Подробности по этой теме смотрите в разделе *DB2 Connect. Руководство пользователя* в части, посвященной безопасности.) Если эта опция используется для задания типа аутентификации экземпляра в файле конфигурации менеджера баз данных, она действует также, как аутентификация **SERVER**, кроме случаев обращения к серверу прикладных программ (AS) через архитектуру DRDA по протоколу APPC. В этом случае задание **DCS** указывает, что аутентификация будет происходить на сервере, но только на уровне APPC. Дальнейшая аутентификация в коде DB2 не производится. Это значение поддерживается только тогда, когда параметр APPC SECURITY для соединения имеет значение SAME или PROGRAM.

DCS_ENCRYPT
Задает, что DB2 Connect принимает зашифрованные схемы аутентификации типа SERVER. Если не задана аутентификация клиента, клиент проходит аутентификацию по методу, выбранному на сервере.

Если аутентификация клиента - DCS или SERVER, клиент проходит аутентификацию, передавая DB2 Connect ID пользователя и пароль. Если аутентификация клиента - DCS_ENCRYPT или SERVER_ENCRYPT, клиент проходит аутентификацию, передавая ID пользователя и зашифрованный пароль.

Если у клиента задано DCS_ENCRYPT, а на сервере - DCS, будет возвращена ошибка несоответствия уровней аутентификации.

DCE Задает, что аутентификация пользователя производится с помощью служб защиты DCE. Дополнительную информацию о защите DCE смотрите в разделе “Использование служб защиты DCE для аутентификации пользователей” на стр. 245.

DCE_SERVER_ENCRYPT
Задает, что сервер принимает аутентификацию DCE или зашифрованные схемы аутентификации типа SERVER. Если аутентификация клиента -

DCE или не задана, аутентификация клиента производится с помощью служб защиты DCE. Дополнительную информацию о защите DCE смотрите в разделе “Использование служб защиты DCE для аутентификации пользователей” на стр. 245.

Если аутентификация клиента - SERVER или DCS, клиент проходит аутентификацию, передавая серверу ID пользователя и пароль. Если аутентификация клиента - SERVER_ENCRYPT или DCS_ENCRYPT, клиент проходит аутентификацию, передавая ID пользователя и зашифрованный пароль. Нельзя задать тип аутентификации клиента DCE_SERVER_ENCRYPT. Если задан тип аутентификации экземпляра DCE_SERVER_ENCRYPT, все локальные прикладные программы будут использовать как схему аутентификации DCE. Это относится также ко всем командам утилит, не требующим соединения с базой данных или подключения к экземпляру.

Тип аутентификации DCE_SERVER_ENCRYPT не только объединяет типы аутентификации DCE и SERVER_ENCRYPT, но также смягчает одно из ограничений при использовании групп с DCE. Когда задан тип аутентификации DCE_SERVER_ENCRYPT, предполагается, что когда список групп запрашивается не в момент аутентификации, его выдает базовая операционная система, а не DCE. Как администратор, вы можете задать на сервере пользователя с соответствующим коротким именем DCE, чтобы обеспечить поддержку списка групп, не ограничиваясь временем аутентификации.

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.

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 аутентификация клиента не KERBEROS, тип аутентификации в системе будет эквивалентен SERVER_ENCRYPT.

Примечание: Типы аутентификация Kerberos поддерживаются только на клиентах и серверах, работающих в среде Windows 2000.

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. Исключения возможны при использовании служб защиты DCE. Информацию о поддержке и использовании удаленных клиентов смотрите в вашем руководстве *Быстрый старт*.
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 платформах OS/2: администратор UPM
   - на других платформах нет локальной системы защиты, поэтому все пользователи все равно должны проходить проверку локальной защиты
3. Дополнительную информацию о системе защиты Windows NT смотрите в разделе “Приложение E. Как DB2 for Windows NT работает с защитой Windows NT” на стр. 383.

Особенности аутентификации для удаленных клиентов

При регистрации базы данных в каталоге для удаленного доступа тип аутентификации может быть задан в записи каталога баз данных.

Для баз данных, к которым обращается оператор DB2 Connect: Если значение не задано, принимается аутентификация SERVER.

Для баз данных, к которым обращаются удаленно, но без DB2 Connect: Тип аутентификации не требуется. Однако если он не задан, клиент должен сначала обратиться к серверу, чтобы получить значение; только после этого начинается процесс аутентификации. Если тип аутентификации задан, аутентификация может начаться немедленно, если переданное значение совпадает с имеющимся на сервере. Если обнаружено несовпадение, DB2 попытается выполнить восстановление, в результате чего могут возникнуть новые процессы для устранения расхождений или же ошибка, если восстановление DB2 невозможно. В случае несовпадения правильным считается значение на сервере.

Особенности распределенной базы 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.

Использование служб защиты DCE для аутентификации пользователей

Хороший вариант системы защиты для распределенной вычислительной среды - службы защиты DCE, обеспечивающие:

- Централизованное управление пользователями и паролями.
- Отсутствие передачи паролей и ID пользователей открытым текстом.
- Единовременную регистрацию для пользователей.

DB2 поддерживает контексты регистрации DCE по умолчанию, контексты регистрации связи и делегированные контексты. *Контекст регистрации по умолчанию* применяется, когда пользователь производит `dce_регистрацию` на клиенте. Дальнейшие команды DB2 имеют доступ к этому контексту и могут выполнять аутентификацию пользователя без его дополнительных действий (то есть без ввода ID пользователя и пароля). *Контекст регистрации связи* применяется для сеанса DB2, использующего ID пользователя и пароль, переданные операторами CONNECT или ATTACH с условием USER/USING. Наконец, *делегированный контекст регистрации* применяется, когда клиент DB2 используется как часть прикладной программы сервера DCE. Прикладная программа сервера DCE (которая также является клиентом DB2) принимает запросы от программы клиента DCE, из которой берутся первоначальные сведения о пользователе. Если клиент DCE и сервер DCE правильно настроены и

позволяют серверу DCE быть делегатом для клиента DCE, DB2 получит делегированный маркер, который перенаправит серверу DB2. Это позволяет серверу DB2 использовать для обработки запросов исходную информацию о клиенте DCE, а не информацию о сервере DCE. Сведения о том, как использовать делегированный контекст регистрации, смотрите в документации DCE для вашей платформы.

Примечание: DCE поддерживается продуктами разных производителей. Чтобы UDB DB2 for Windows NT могла работать с продуктом DCE IBM для служб защиты, поставляются две библиотеки DLL: `db2dces.ibm` и `db2dcec.ibm`. (Эти файлы DLL подходят только для Windows NT.) Если вы приобрели и используете как службы защиты продукт DCE IBM, нужно скопировать два файла с именами `db2dces.dll` и `db2dcec.dll` соответственно. Если вы собираетесь приобрести продукт другого производителя DCE, обратитесь к службам поддержки этого производителя и службам поддержки UDB DB2 и узнайте, сможет ли реализация служб защиты DCE, созданная этим производителем, работать с UDB DB2.

Как настроить пользователя DB2 для DCE

Прежде чем начать работу с DB2, пользователи должны быть зарегистрированы в реестре DCE с правильными атрибутами. Информацию о том, как создать принципал DCE, смотрите в документации DCE для соответствующей платформы.

У каждого пользователя DB2, который собирается использовать сервер аутентификации DCE, должен быть принципал DCE и учетная запись, определенная в реестре DCE, с установленным флагом клиента. У этого принципала также должна быть запись в разделе расширенных атрибутов реестра (ERA), показывающая, какое имя авторизации будет использоваться для этого принципала, когда он соединяется с определенным сервером аутентификации DCE.

Возможно, вы также захотите, чтобы принципалы пользователей были членами групп, чтобы использовать привилегии групп в базе данных. Аналогичная информация ERA группы отображает имя группы в имя авторизации DB2. Это имя авторизации - имя вторичной авторизации, но для него действуют те же ограничения. Дополнительную информацию о том, как создавать группы и добавлять члены, смотрите в вашей документации по DCE.

Информация в ERA отображает принципал DCE пользователя или имя группы в имя авторизации DB2 для имени принципала конкретного DCE сервера. Чтобы использовать ERA, нужно определить схему ERA, указывающую формат данного атрибута. Это необходимо сделать для каждой ячейки DCE, для чего надо выполнить следующие действия:

1. Зарегистрируйтесь в DCE как допустимый администратор DCE
2. Вызовите dcecp и введите в приглашении:

```
> xattrschema create /.:/sec/xattrschema/db2map \
> -aclmgr {{principal r m r m } {group r m r m }} \
> -annotation {Schema entry for DB2 database access} \
> -encoding stringarray \
> -multivalued no \
> -uuid 1cbe84ca-9df3-11cf-84cd-02608c2cd17b
```

При этом будет создан расширенный атрибут реестра db2map.

Чтобы просмотреть это отображение, введите следующую команду в приглашении dcecp:

```
> xattrschema show /.:/sec/xattrschema/db2map
```

Вы увидите:

```
{axlmgr
 {{principal {{query r} {update m} {test r} {delete m}}}
  {group     {{query r} {update m} {test r} {delete m}}}}}
{annotation {Schema entry for DB2 database access}}
{applydefs no}
{intercell rejects}
{multivalued no}
{reserved no}
{scope {}}
{trigbind {}}
{trigtype none}
{unique no}
{uuid 1cbe84ca-9df3-11cf-84cd-02608c2cd17b}
```

Примечание: Ограничения на содержание имени авторизации, записанное в ERA, не поддерживаются DCE. Если принципал DCE или группа получит недопустимое имя авторизации, попытка DB2 провести аутентификацию этого пользователя приведет к ошибке. (Не забудьте, что аутентификация может происходить при операциях CONNECT, ATTACH, DB2START и любых других операциях, требующих аутентификации.) Также настоятельно рекомендуется убедиться, что назначение имен авторизации принципалам DCE является взаимно-однозначным. DCE не проверяет соблюдения этого условия.

Если клиент DB2 должен обращаться к серверу UDB DB2, когда оба уже будут зарегистрированы как принципалы DCE, информация ERA должна быть дополнена, чтобы отобразить имя принципала в имя авторизации. Это необходимо сделать для каждого пользователя или каждой группы, для чего надо выполнить следующие действия:

- Зарегистрируйтесь в DCE как допустимый администратор DCE
- Вызовите dcecp и в ответ на приглашение введите:

```
> principal modify имя_принципала \
> -add {db2map отображение_1 отображение_2...отображение_n}
```

где `отображение_n` имеет следующий вид:

```
принципал_сервера_DCE,id_авт_DB2
```

где `принципал_сервера_DCE` - допустимое имя принципала DCE для сервера UDB DB2 (или символ подстановки *, который указывает, что это отображение применимо для всякого сервера DB2, еще не заданного в другой записи `отображение_n`), а `id_авт_DB2` - допустимое имя авторизации DB2.

Если группа DCE должна использоваться как принципал DCE, она должна отображаться в ID авторизации DB2 с нужными полномочиями, например, SYSADM или SYSCTRL.

Обратите внимание на то, что идентификатор авторизации (authid), заданный в схеме DCE, используемой для отображения имени принципала DCE в ID авторизации DB2, должен задаваться *только* в верхнем регистре. Символы нижнего регистра в ID авторизации приведут к ошибке.

Как настроить сервер DB2 для DCE

Прежде чем начать работу с DB2, надо зарегистрировать серверы в реестре распределенной вычислительной среды (DCE) как принципалы с правильными атрибутами. Информацию о том, как создать принципал сервера DCE, смотрите в документации по DCE для соответствующей платформы.

Программа клиента защиты DCE времени выполнения должна быть установлена и доступна для экземпляра сервера.

Каждый сервер DB2, который хочет использовать DCE как механизм аутентификации, должен зарегистрироваться с помощью DCE во время запуска DB2START. Чтобы не делать это вручную, DCE поддерживает метод, при котором сервер хранит собственную информацию ID пользователя и пароля (ключ) в специальном файле, называемом файлом *keytab*. При выполнении DB2START DB2 читает файл конфигурации менеджера баз данных и узнает тип аутентификации экземпляра. Если тип аутентификации - DCE, сервер DB2 вызывает DCE для получения информации из файла keytab. Эта информация используется для регистрации сервера на DCE.Такая регистрация позволяет серверу принимать от клиентов DCE маркеры DCE и использовать их для аутентификации этих пользователей.

Администратор экземпляра должен создать файл keytab для экземпляра с помощью команд DCE. Подробная информация о том, как создать файл keytab, содержится в документации по DCE для вашей платформы. Посмотрите в этом документе подробности, связанные с файлом keytab и командами *dcecp keytab* или *rgy_edit*. Файл keytab DB2 должен называться *keytab.db2* и располагаться в

подкаталоге security каталога sqllib этого экземпляра. (Для систем на базе Intel файл должен находиться в подкаталоге security подкаталога INSTANCENAME каталога sqllib. INSTANCENAME - имя экземпляра, с которым вы работаете.) Он должен содержать только одну запись для принципала сервера заданного экземпляра; нарушение приведет к ошибке во время выполнения DB2START. На платформах с операционной системой UNIX этот файл должен быть защищен разрешениями файла, допускающими чтение/запись только владельцем экземпляра.

Ниже приведен пример создания файла keytab:

- Зарегистрироваться в DCE как допустимый пользователь DCE
- Вызвать rgy_edit и ввести в ответ на приглашение:

```
> ktadd -p имя_принципала -pw пароль_принципала \
> -f keytab.db2
```

Чтобы запустить DB2 с аутентификацией DCE, после завершения настройки DCE нужно сообщить DB2 об использовании аутентификации DCE, изменив в файле конфигурации менеджера баз данных тип аутентификации на “DCE”. Для этого в командной строке введите команду:

```
db2 update database manager configuration using authentication DCE
 группа_sysadm имя_группы_DCE
```

Затем выполните dce_login для допустимого пользователя DCE DB2 с полномочиями SYSADM и запустите DB2START.

Примечание: Прежде чем запускать DB2 с аутентификацией DCE, убедитесь, что вы определили использование принципала пользователя DCE в качестве SYSADM экземпляра, то есть у вас есть допустимый ID пользователя DCE, который сможет запускать, останавливать экземпляр и управлять им. Как это сделать, смотрите в разделе “Как настроить пользователя DB2 для DCE” на стр. 246.

Помимо этих указаний, убедитесь, что созданный принципал является членом группы системных администраторов экземпляра. По умолчанию имя этой группы для аутентификации DCE - DB2ADMIN, если ни одной группы не задано явно (то есть группа системных администраторов пуста), но его можно до изменения типа аутентификации экземпляра поменять на имя группы (имя авторизации) по вашему выбору. Выбранная вами группа DCE должна иметь заданную ERA, которая отображает ее в заданное имя авторизации группы системных администраторов.

Одна из функций сервера администратора DB2 - запуск экземпляров DB2. Если применяется аутентификация, принципал DCE, используемый в файле keytab DB2 экземпляра, должен

отображаться в допустимый ID аутентификации DB2. Это необходимо, чтобы сервер администрации DB2 запускал экземпляр DB2. Такое отображение позволяет ID работать как клиентом, так и сервером.

Как настроить экземпляр клиента DB2 для DCE

Установить использование аутентификации DCE экземпляру рядового клиента для локальных операций можно, изменив файл конфигурации менеджера баз данных и задав тип аутентификации DCE. Для экземпляра рядового клиента файл keytab не требуется, поскольку нет сервера, которому нужно регистрироваться на DCE. В общем случае не рекомендуется (и не требуется), чтобы экземпляр DB2 рядового клиента использовал аутентификацию DCE, хотя это и поддерживается.

Клиенту, который должен обращаться к удаленной базе данных с помощью системы защиты DCE, требуется доступ к соответствующему продукту системы защиты DCE. При желании клиент может зарегистрировать тип аутентификации для базы данных назначения в каталоге баз данных. Если клиент задает тип аутентификации DCE, необходимо также задать полный путь и имя главного сервера DCE. Если в каталоге не указана аутентификация DCE, информация об аутентификации и принципале поступает от сервера во время выполнения операции CONNECT.

Ограничения DB2 при использовании системы защиты DCE

Использование аутентификации DCE накладывает ограничения на некоторые функции SQL, обеспечиваемые DB2 и связанные с поддержкой группы. Существуют следующие ограничения при использовании аутентификации DCE:

- При использовании операторов GRANT и REVOKE **обязательны** ключевые слова USER и GROUP, задающие имя авторизации; их отсутствие приводит к ошибке.
- При использовании условия AUTHORIZATION в операторе CREATE SCHEMA членство в группе заданного имени авторизации **не** будет учитываться при оценке авторизации, требуемой для выполнения операторов, следующих за этим условием. Это может привести к ошибке авторизации при выполнении оператора CREATE SCHEMA.
- При повторном связывании пакета пользователем, отличным от первоначального связывателя пакета, привилегии первоначального связывателя проверяются еще раз.В этом случае членство в группе первоначального связывателя не учитывается при повторной проверке привилегий. Это может привести к ошибке авторизации при повторном связывании.

Аутентификация DCE, выполняемая DB2, передает билеты DCE, полученные с помощью интерфейса Generic Security Services Application Programming Interface (GSSAPI) среды OSF DCE. Таким образом, вся аутентификация для системы защиты DCE происходит в слое протокола базы данных. Некоторые механизмы

связи могут обеспечивать дополнительную защиту слоя связи, возможно, не интегрированную с DCE. Если аутентификация слоя связи может оставаться полностью независимой от аутентификации слоя протокола базы данных, никакие ограничения не поддерживаются. Однако прежде чем связь будет успешно установлена, требуется выполнение критериев аутентификации как слоя протокола базы данных, так и слоя связи. Если механизмы аутентификации слоя протокола базы данных и слоя протокола связи взаимодействуют, их использование может быть ограничено, поскольку некоторые сочетания создают брешь в защите.

Аутентификация DCE может использоваться в сочетании с поддержкой TCPIP SOCKS; однако эти два механизма защиты будут работать независимо друг от друга. Отсюда следует, что пользователю, возможно, придется не только обеспечить допустимый контекст регистрации DCE, но и регистрировать в локальной операционной системе ID пользователя, удовлетворяющий критериям сервера SOCKS.

Аутентификация DCE может использоваться в сочетании с именованными конвейерами NT, однако эти два механизма защиты будут работать независимо друг от друга. Пользователю придется не только обеспечить допустимый контекст регистрации DCE, но и регистрировать на домене NT ID пользователя, удовлетворяющий критериям поддержки именованных конвейеров NT.

Чтобы не допустить возможных недоразумений, когда для аутентификации используются и принципалы DCE, и ID пользователей локальной операционной системы, как в двух приведенных выше примерах, можно использовать интегрированную регистрацию DCE. В этом случае при регистрации в системе пользователь автоматически регистрируется и на соответствующем принципале DCE. Подробности о том, как использовать эту возможность, смотрите в документации DCE для вашей платформы. Обратите внимание на то, что этот подход предполагает одно и то же имя для принципала DCE и ID локальной операционной системы. Это может привести к тому, что значение, которое содержится в зашифрованном билете DCE, передается в то же время в незашифрованном виде в слое связи.

Аутентификация DCE может использоваться только совместно со связью APPC, когда параметр SECURITY имеет значение NONE. Это делается для того, чтобы не посылать незашифрованный принципал и/или пароль на слое связи при использовании шифрованного маркера DCE для того же принципала на слое протокола базы данных. Система защиты DCE на слое APPC в настоящее время не поддерживается DB2.

Обработка аутентификации базы данных объединения

Если вы установили функцию распределенного объединения данных и задали для переменной конфигурации менеджера баз данных *federated* значение 'YES', ваша система DB2 работает как система объединения. Настройка аутентификации баз данных в системе объединения слегка отличается от стандартных определений DB2. Кроме того, в системе объединения нужно учитывать требования аутентификации источников данных. В общем случае источники данных (DB2, Oracle, DB2 для OS/390 и так далее) настроены так, что требуют аутентификации. Поэтому нужно убедиться, что запрашиваемые ID и пароли могут передаваться источникам данных. DB2 предлагает несколько методов поддержки аутентификации в источниках данных, и все они описаны в этом разделе.

Параметры аутентификации

SERVER

Задает, что клиенты, устанавливающие связь с DB2, для получения доступа предоставляют ID пользователя и пароль. В этом случае ID пользователя и пароль доступны для передачи источникам данных. Передаваемой источникам данных информацией можно управлять с помощью опций сервера и отображений пользователей, но информация аутентификации будет доступна для передачи источникам данных.

CLIENT

Задает, что аутентификация производится на разделе базы данных при вызове программы с использованием средств защиты операционной системы. Никакие пароли не становятся доступными для прямой передачи источникам данных. В этом случае, если источник данных требует аутентификации, нужно создать одно или несколько отображений пользователей. Надо должны убедиться, что заданы опции сервера, обеспечивающие передачу источнику данных корректной информации ID пользователя и пароля.

Будьте крайне осторожны, используя аутентификацию CLIENT. Прибегайте к этой форме аутентификации только в безопасных сетях. Пользователь имеет полномочия SYSADM для базы данных объединения, если соблюдены следующие условия:

- Задан тип аутентификации CLIENT.
- Пользователь имеет на клиенте статус root.
- Пользователь знает имя авторизации SYSADM.
- Пользователь определяет имя авторизации на клиенте, совпадающее с именем SYSADM на DB2.

DCS Задает, что аутентификация производится на источнике данных, а не в DB2. В этом случае аутентификация происходит в обход стандартного процесса DB2.ID пользователей и пароли передаются прямо источникам

данных в зависимости от настройки опций сервера. Аутентификация производится только на источниках данных семейств Oracle и DB2.

Будьте осторожны, когда задан тип аутентификации DCS. Аутентификация не производится ни у клиента, ни в DB2. Любой пользователь, который знает имя аутентификации SYSADM, может получить полномочия SYSADM для сервера объединения.

DCE Если задан тип аутентификации DCE, только ID пользователя доступен для передачи источникам данных. Пароли недоступны. Если источник данных требует обработать аутентификацию (ID пользователя и пароль), нужно определить отображение пользователей, которое передаст пароль (и, возможно, ID пользователя) источнику данных. Если источник данных доверяет связи с DB2, отображения пользователей не требуются, поскольку источнику данных может передаваться ID, полученный из внешней системы защиты.

Возможны другие настройки аутентификации DB2, и некоторые могут приводить к тому, что пароль DB2 станет доступен для передачи источникам данных. Если настройки аутентификации DB2 и клиента приводят к передаче пароля DB2, этот пароль становится доступен для дополнительных процессов аутентификации в источниках данных. Дополнительную информацию смотрите в разделе Табл. 4 на стр. 241.

Передача ID пользователей и паролей источникам данных

Есть четыре способа регулировать передачу информации аутентификации источникам данных - параметры аутентификации DB2, отображения пользователей, опции сервера и параметры защиты APPC:

Параметры аутентификации

Задача этого раздела - объяснить, как параметры аутентификации влияют на процесс глобальной аутентификации в системе объединения (определения для параметров аутентификации содержатся в разделе “Параметры аутентификации” на стр. 252). Так, если задан тип аутентификации DB2 SERVER или DCS, для соединения требуется ID пользователя и пароль. Поэтому ID пользователя и пароль становятся доступны для передачи на источники данных. Если задан тип аутентификации DCE или CLIENT, и аутентификация не производится в системе DB2, содержащей объединенную базу данных, доступен только ID пользователя. Если процесс аутентификации источника данных требует пароля (или, возможно, другого ID пользователя и пароля), нужно создать отображение пользователей. Если задан тип аутентификации CLIENT и параметр *trust_clntauth* SERVER, возможно, чтобы пароль посылался DB2 и был доступен для передачи источникам данных.

Отображения пользователей

DB2 может посылать либо имя авторизации, используемое для соединения с DB2, либо имя авторизации, определенное в DB2. Отображения пользователей хранят имена авторизации, определенные в DB2. Они создаются с помощью оператора CREATE USER MAPPING.

Отображения пользователей обладают гибкостью: можно отобразить ID в новый ID и пароль, а можно просто в пароль. С их помощью можно восполнить недостающую информацию или изменить ID и пароль на значения, приемлемые на источнике данных.

Чтобы создать или изменить отображение пользователей, нужно, чтобы у вас были полномочия SYSADM или DBADM или чтобы ваш ID аутентификации совпадал с именем авторизации, заданном в операторе.

Пример оператора отображения пользователей:

```
CREATE USER MAPPING FOR "SHAWN" SERVER DB21 OPTIONS (REMOTE_AUTHID "SHAWNBCA",
REMOTE_PASSWORD "MAPLELEAF")
```

здесь ID аутентификации DB2 (SHAWN) отображается в удаленный ID SHAWNBCA и удаленный пароль MAPLELEAF для сервера с именем DB21.

Если потребность в передаче строки связана только с различием между именем авторизации (или паролем) на DB2 и именем авторизации (или паролем) в источнике данных, рассмотрите возможность обойтись опциями сервера, чтобы добиться нужной настройки, не создавая новые ID и пароли. Дополнительную информацию смотрите в разделе “Опции сервера”.

Создать отображение пользователей нужно, если задан тип аутентификации DCE и источник данных требует проведения аутентификации (ожидая пароля). DB2 передает источникам данных только ID пользователя DCE. Пароль нужно отобразить в этот ID пользователя и затем послать на источник данных.

Опции сервера

Опции сервера могут обеспечить общую поддержку аутентификации. С их помощью можно указать, будут ли пароли передаваться источникам данных (в типичном случае - да) и должны ли ID пользователей и пароли переводится в верхний регистр или в нижний регистр. Опции сервера задаются с помощью операторов CREATE SERVER, ALTER SERVER и SET SERVER OPTION.

Оставшаяся часть этого раздела посвящена опциям сервера для конкретных процессов аутентификации. Более полный список опций сервера приведен в разделе “Использование опций сервера для определения характеристик источников данных и настройки процесса аутентификации” на стр. 161.

Опция сервера password: Значение password по умолчанию - 'Y' (пароли посылаются источникам данных). Оставьте или задайте значение 'Y' во всех случаях, когда источник данных должен выполнить аутентификацию и не ожидает зашифрованного пароля.

DB2 может передавать зашифрованные пароли. Задайте для опции сервера password значение 'ENCRYPTION', если пароли должны посылаться источникам данных семейства DB2 в зашифрованном виде. Рекомендуется задать password 'ENCRYPTION', если задан тип аутентификации DB2 DCS_ENCRYPT или SERVER_ENCRYPT.

ID пользователя всегда посылается источникам данных.

Опции преобразования ID и пароля: Имена и пароли авторизации в некоторых случаях нужно изменять. Разные источники данных могут иметь разные требования к имени авторизации и паролю (в части использования верхнего и нижнего регистров в ID и паролях).

В DB2 есть две опции сервера, помогающие преодолеть различия в системах именования. Эти опции называются **fold_id** и **fold_pw**; для них возможны следующие значения:

'U' DB2 преобразует имя авторизации и пароль в верхний регистр, прежде чем посылать их источнику данных.

'N' DB2 не преобразует имя авторизации и пароль.

'L' DB2 преобразует имя авторизации и пароль в нижний регистр, прежде чем посылать их источнику данных.

null DB2 сначала посылает имя авторизации и пароль в верхнем регистре; если возникает ошибка, DB2 преобразует их в нижний регистр и посылает снова.

Пустое значение может показаться привлекательным, поскольку охватывает много возможностей. Однако по соображениям производительности лучше задать эти опции так, чтобы соединения устанавливались с одной попытки. Если обе опции fold_id и fold_pw заданы пустыми, DB2 может предпринять до четырех попыток послать имя авторизации и пароль:

1. И имя авторизации, и пароль в верхнем регистре.
2. Имя авторизации в верхнем регистре, а пароль в нижнем.
3. Имя авторизации в нижнем регистре, а пароль в верхнем регистре.
4. И имя авторизации, и пароль в нижнем регистре.

Параметры защиты APPC

Если вы соединяетесь с источником данных DRDA через APPC (для чего требуется ID пользователя и пароль), или если задан тип аутентификации DCS и

аутентификация происходит на источнике данных DRDA, убедитесь, что для связи между DB2 и этим источником данных параметр защиты APPC имеет значение PROGRAM.

Пример аутентификации базы данных объединения

В этом разделе дается обзор аутентификации системы объединения и шаги авторизации. Обзор процесса аутентификации и авторизации базы данных объединения смотрите в разделе рис. 3.

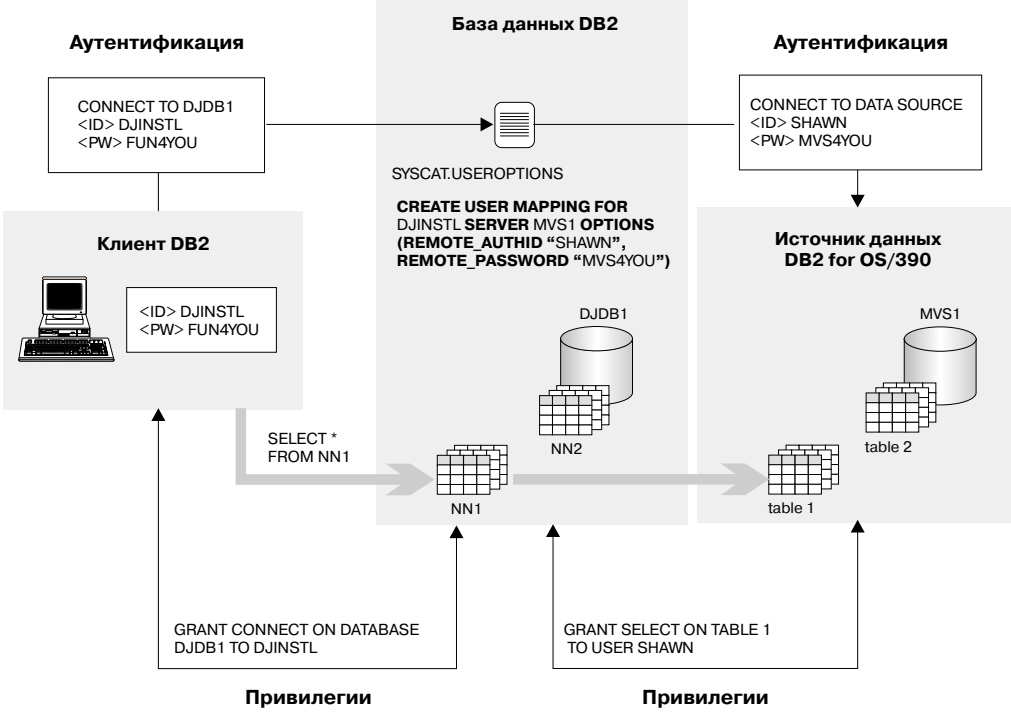


Рисунок 3. Процесс аутентификации и авторизации базы данных объединения

Задача следующего сценария - разрешить пользователю DJINSTL выполнить операцию UNION над двумя псевдонимами NN1 и NN2. Псевдонимы относятся к двум таблицам.Один источник данных - система DB2 для OS/390, где у DJINSTL есть другой ID пользователя и пароль (смотрите рис. 3) с именем MVS1. Для доступа к информации MVS1 потребуется отображение пользователей. Другой источник данных - система DB2, где ID DJINSTL и пароль те же самые. Этот источник данных, DB21, требует только, чтобы ID пользователя и пароль посылались в верхнем регистре.

Задан тип аутентификации DB2 SERVER. DJINSTL будет обращаться к DB2 из клиента Windows NT через соединение TCP/IP. Связь DB2 с DB2 для OS/390 - тоже TCP/IP. Имя базы данных объединения - DJDB1.

Вначале убедитесь, что DB2 ожидает пароль и что пароль посылается. Убедитесь также, что типы аутентификации клиента и сервера совпадают. Для проверки типа аутентификации сервера DB2 введите команду:

```
GET DATABASE MANAGER CONFIGURATION
```

на сервере DB2. Для проверки типа аутентификации клиента введите команду:

```
LIST DATABASE DIRECTORY
```

на клиенте. В обоих случаях убедитесь, что тип аутентификации - SERVER. Если параметр клиента - DCS или CLIENT, его можно изменить с помощью команд UNCATALOG DATABASE и CATALOG DATABASE.

Далее убедитесь, что пароли будут посылаться источникам данных. После соединения с объединенной базой данных DJDB1 введите команды:

```
ALTER SERVER MVS1 OPTIONS (SET password 'Y')
ALTER SERVER DB21 OPTIONS (SET password 'Y')
```

Затем убедитесь, что пароли посланы источнику данных DB21 в соответствующем регистре:

```
ALTER SERVER DB21 OPTIONS (ADD fold_id 'U')
ALTER SERVER DB21 OPTIONS (ADD fold_pw 'U')
```

На следующем шаге присвойте привилегии, позволяющие пользователю DJINSTL связаться с объединенной базой данных DJDB1 и заданными псевдонимами:

```
 GRANT CONNECT ON DATABASE DJDB1 TO DJINSTL;
```

Теперь отобразите ID и пароль DB2 DJINSTL в корректные ID и пароль для сервера MVS1:

```
CREATE USER MAPPING FOR "DJINSTL" SERVER MVS1 OPTIONS (REMOTE_AUTHID "SHAWN",
REMOTE_PASSWORD "MVS4YOU")
```

Теперь ID пользователя DB2 DJINSTL может посылать запросы на источники данных. Для доступа к объектам источников данных по псевдонимам могут потребоваться дополнительные действия (обычно для обращения к таблицам и производным таблицам по псевдонимам требуются привилегии).

Привилегии, полномочия и авторизация

Привилегии позволяют пользователям создавать ресурсы баз данных и обращаться к ним. *Уровни полномочий* позволяют группировать привилегии работы высокого уровня с менеджером баз данных и операций с утилитами. 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.

Существуют следующие полномочия:

- “Полномочия управления системой (SYSADM)” на стр. 260
- “Полномочия управления системой (SYSCTRL)” на стр. 261
- “Полномочия обслуживания системы (SYSMAINT)” на стр. 262
- “Полномочия управления базой данных (DBADM)” на стр. 263
- “Полномочия LOAD” на стр. 263

Существуют следующие типы привилегий:

- “Привилегии базы данных” на стр. 264
- “Привилегии схем” на стр. 266
- “Привилегии табличных пространств” на стр. 267
- “Привилегии таблиц и производных таблиц” на стр. 267
- “Привилегии псевдонимов” на стр. 270
- “Привилегии сервера” на стр. 271
- “Привилегии пакетов” на стр. 271
- “Привилегии индексов” на стр. 272.

На рис. 4 на стр. 259 показана связь между полномочиями и их областью действия (база данных, менеджер баз данных).

Полномочия

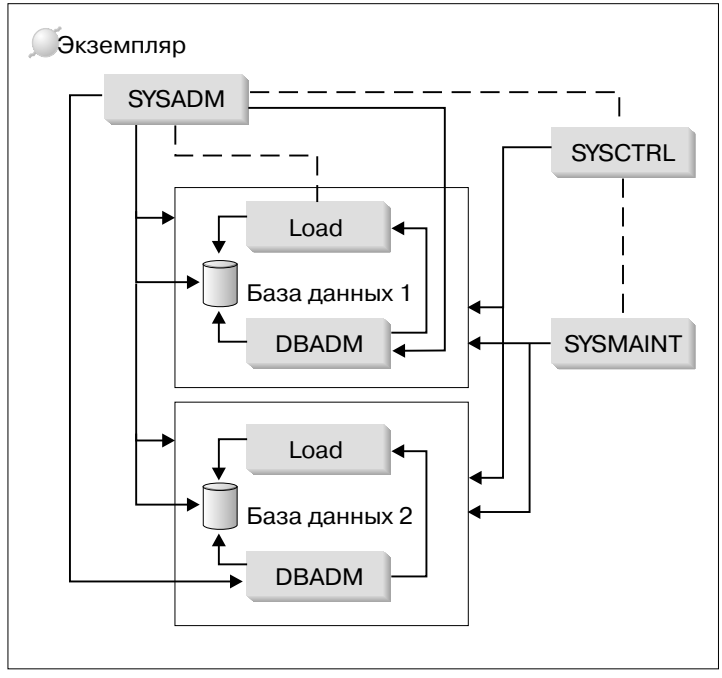


Рисунок 4. Иерархия полномочий

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. Дополнительную информацию смотрите в разделе “Предоставление косвенных привилегий посредством пакета” на стр. 277.

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. (Смотрите раздел “Управление доступом к объектам баз данных” на стр. 272.) 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.

Информацию об авторизации, требуемой для конкретных команд, интерфейсов API и операторов SQL, смотрите в книгах *Command Reference*, *Administrative API Reference* или *SQL Reference*.

Полномочия управления системой (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 узлов, пулы буферов и мониторы событий.

Полномочия SYSADM присваиваются группе, заданной параметром конфигурации *sysadm_group* (смотрите раздел “Настройка DB2” в *Administration Guide: Performance*). Членство в этой группе управляется вне менеджера баз данных через утилиту защиты, используемую на вашей платформе. Информацию о том, как использовать вашу системную утилиту защиты для создания, изменения и удаления полномочий SYSADM, смотрите в книге *Быстрый старт*.

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.

Кроме того, пользователь с полномочиями SYSADM может выполнять функции пользователей со следующими полномочиями:

- "Полномочия управления системой (SYSCTRL)"
- "Полномочия обслуживания системы (SYSMAINT)" на стр. 262
- "Полномочия управления базой данных (DBADM)" на стр. 263

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.

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 присваиваются группе, заданной параметром конфигурации *sysctrl_group* (смотрите раздел "Настройка DB2" в *Administration Guide: Performance*). 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 “Полномочия 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.

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 присваиваются группе, заданной параметром конфигурации *sysmaint_group* (смотрите раздел “Настройка DB2” в *Administration Guide: Performance*). 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
- Собирать 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. Когда предоставляются полномочия DBADM, одновременно 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
- Собирать статистику каталога с помощью утилиты RUNSTATS.

Примечание: DBADM может выполнять перечисленные функции только на той базе данных, для которой у него есть полномочия DBADM.

Полномочия LOAD

Пользователи, имеющие полномочия LOAD на уровне базы данных, а также привилегию INSERT для таблицы, могут выдавать команду LOAD и запускать утилиту AutoLoader для загрузки данных в таблицу.

Пользователи, имеющие полномочия LOAD на уровне базы данных, а также привилегию INSERT для таблицы, могут выдавать команды LOAD RESTART и LOAD TERMINATE, если предыдущая операция загрузки - вставка данных.

Если предыдущая операция загрузки - загрузка с заменой, пользователю нужна привилегия DELETE, чтобы он мог выдавать команды LOAD RESTART и LOAD TERMINATE.

Если при операции LOAD используются таблицы исключений, пользователю нужна привилегия INSERT для таблиц исключений.

Пользователь с этими полномочиями может выполнять команды QUIESCE TABLESPACES FOR TABLE, RUNSTATS и LIST TABLESPACES.

Привилегии базы данных

В рис. 5 приведен список привилегий базы данных.

Привилегии баз данных

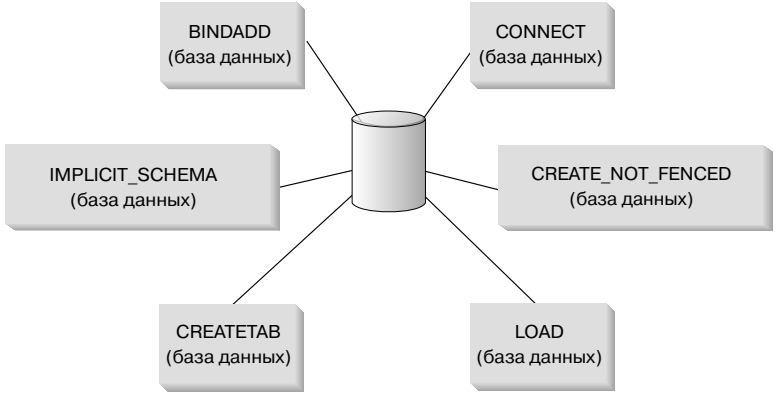


Рисунок 5. Привилегии базы данных

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.) (Дополнительную информацию можно найти в справочниках *Application Development Guide* и *SQL Reference*.)

- 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.

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.

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.

Привилегии схем

Привилегии схем относятся к категории привилегий объектов. Привилегии объектов перечислены в разделе рис. 6.

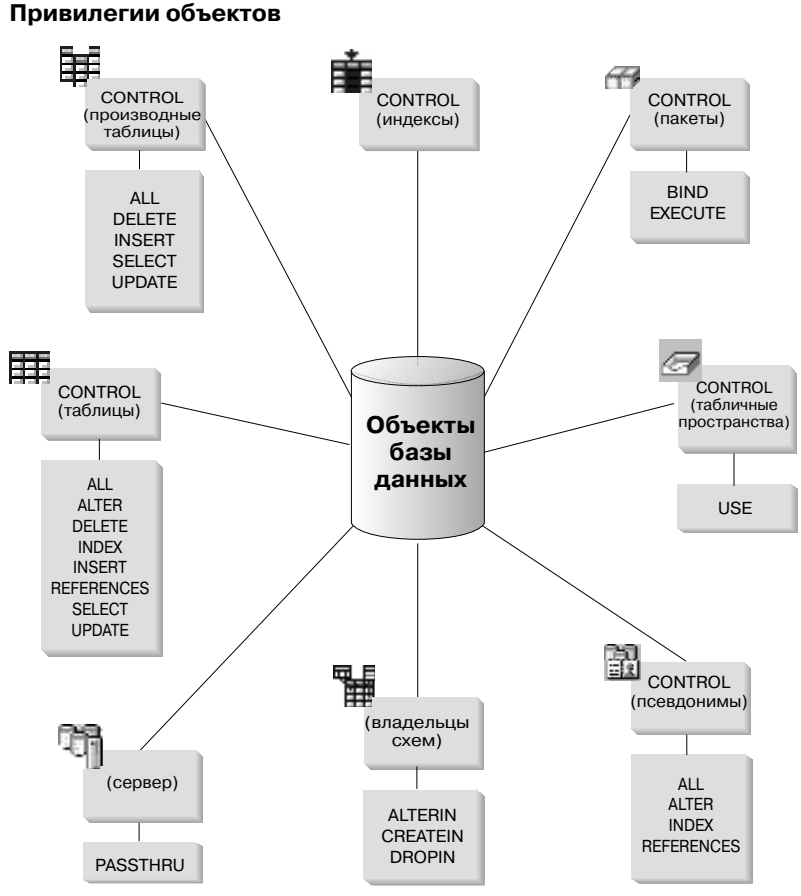


Рисунок 6. Привилегии объектов

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.

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.

Привилегии таблиц и производных таблиц

Привилегии таблиц и производных таблиц касаются действий для таблиц или производных таблиц в базе данных. Пользователю нужна привилегия CONNECT для базы данных, чтобы использовать какие-либо из следующих привилегий:

- 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 создавать индексы для таблицы. Создатель индекса автоматически получает привилегию CONTROL для этого индекса. Дополнительную информацию смотрите в разделе “Привилегии индексов” на стр. 272.
- 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.

В следующих руководствах содержится информация об авторизациях, требуемых для выполнения конкретных команд, интерфейсов API и операторов SQL:

- *SQL Reference*

- *Command Reference*
- *Administrative API Reference*.

Информацию об авторизации, требуемой для изменения статистики каталога, смотрите в книге *Administration Guide: Performance*.

Информацию о том, как определяются привилегии производной таблицы, смотрите в разделе об операторе CREATE VIEW в руководстве *SQL Reference*.

Привилегии псевдонимов

Привилегии псевдонимов касаются действий для псевдонимов в базе данных. Эти привилегии не затрагивают привилегий для объектов - источников данных, обозначаемых псевдонимами. Пользователю нужна привилегия CONNECT для базы данных, чтобы использовать какие-либо из следующих привилегий:

- Привилегия CONTROL предоставляет пользователю все привилегии для псевдонимов, включая возможность отбросить псевдоним, а также давать и отзывать отдельные привилегии псевдонимов. Чтобы предоставить привилегию CONTROL, нужно иметь полномочия SYSADM или DBADM. Создатель псевдонима автоматически получает привилегию CONTROL для этого псевдонима.
- Привилегия ALTER позволяет пользователю изменять имена столбцов в псевдониме, добавлять и изменять тип DB2, в который отображается тип данных столбца, и задавать опции столбца в столбцах псевдонима.
- Привилегия INDEX позволяет пользователю создать спецификацию индекса на псевдониме. Создатель спецификаций индекса автоматически получает привилегию CONTROL для этого индекса.
- Привилегия REFERENCES позволяет пользователю создать и отбросить внешний ключ, задав родительский статус псевдонима по отношению к другим псевдонимам. 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 псевдонима, все остальные привилегии для этого псевдонима автоматически предоставляются с опцией WITH GRANT OPTION. Если затем привилегия пользователя CONTROL для псевдонима отзывается, у пользователя сохраняются остальные привилегии, которые были предоставлены автоматически.

Чтобы получить доступ к данным источника данных, надо также иметь соответствующую авторизацию для объектов в источниках данных, на которые указывают псевдонимы.

Когда пользователь обращается к производной таблице, содержащей ссылки на один или несколько псевдонимов, этому пользователю должен быть разрешен доступ к производной таблице и объектам в источниках данных, обозначенным псевдонимами.

Привилегии сервера

Существует только одна привилегия сервера: PASSTHRU. Эта привилегия определяет, какие ID авторизации могут выдавать операторы DDL и DML непосредственно (минуя сервер) источникам данных.

DB2 поддерживает два оператора SQL, управляющих операциями через промежуточный сервер:

- GRANT PASSTHRU, который предоставляет полномочия выполнять операторы SET PASSTHRU для источника данных и передавать операторы DML и DDL этому источнику данных, минуя сервер.
- REVOKE PASSTHRU, который отзывает полномочия выполнять операторы SET PASSTHRU для источника данных и передавать операторы DML и DDL этому источнику данных, минуя сервер.

Пример оператора, предоставляющего привилегию PASSTHRU пользователю SHAWN для сервера ORACLE1:

```
GRANT PASSTHRU ON SERVER ORACLE1 TO USER SHAWN
```

Полную информацию о синтаксисе операторов PASSTHRU смотрите в справочнике *SQL Reference*.

Привилегии пакетов

Пакет - это объект базы данных с информацией, позволяющей менеджеру баз данных эффективно обращаться 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 позволяет пользователю повторно связать существующий пакет.
- Привилегия EXECUTE позволяет пользователю выполнить пакет.

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.

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. Дополнительную информацию о том, как DB2 обрабатывает статические и динамические SQL, смотрите в справочнике *SQL Reference*.

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 создать индекс для этой таблицы (смотрите раздел “Привилегии таблиц и производных таблиц” на стр. 267).

Привилегии последовательностей

Создатель последовательности автоматически получает привилегию USAGE. Привилегия USAGE позволяет использовать для этой последовательности выражения NEXTVAL и PREVVAL. Чтобы разрешить другим пользователям использовать выражения NEXTVAL и PREVVAL, привилегии последовательности надо предоставить всем (то есть группе PUBLIC). Это позволит использовать выражения с данной последовательностью всем пользователям.

Управление доступом к объектам баз 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. Методы аудита реализации плана управления доступом к базе данных описаны в разделе “Использование системного каталога” на стр. 285.

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.

Далее описаны следующие темы:

- “Предоставление привилегий”
- “отзыв привилегий” на стр. 274
- “Управление неявными авторизациями при создании и отбрасывании объектов” на стр. 276
- “Предоставление косвенных привилегий посредством пакета” на стр. 277
- “Управление доступом к данным с помощью производных таблиц” на стр. 279
- “Управление доступом к данным с помощью утилиты аудита” на стр. 282.

Предоставление привилегий

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 привилегии SELECT для таблицы EMPLOYEE предоставляются пользователю HERON:

```
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
```

Для большинства объектов право предоставлять привилегии принадлежит пользователям с полномочиями SYSADM или DBADM или привилегией CONTROL для данного объекта; кроме того, привилегию может предоставить пользователь, обладающей 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.

Дополнительную информацию об операторе GRANT смотрите в руководстве *SQL Reference*.

отзыв привилегий

Оператор REVOKE позволяет авторизованным пользователям отзывать привилегии, ранее предоставленные другим пользователям. Чтобы отозвать привилегии для объектов баз данных, нужно иметь полномочия DBADM, полномочиями SYSADM или привилегией CONTROL для этих объектов. Обратите внимание на то, что обладание привилегией 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. Дополнительную информацию о полномочиях, требуемых для отзыва привилегий, смотрите в руководстве *SQL Reference*.

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).

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.

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. Когда предпринимается попытка использовать пакет, помеченный 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. Дополнительную информацию об операторах REVOKE и REBIND PACKAGE смотрите в справочнике *SQL Reference*.

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.

Управление неявными авторизациями при создании и отбрасывании объектов

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. Аналогичным образом при отбрасывании объекта привилегии уничтожаются.

Если создаваемый объект - табличное пространство, таблица, индекс или пакет, пользователь получает привилегию CONTROL для объекта. Если этот объект - производная таблица, привилегия CONTROL для производной таблицы неявно предоставляется, только если пользователь располагает привилегией CONTROL для всех таблиц и производных таблиц, которые упоминаются в определении этой 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.

Информацию о том, как определяются привилегии производной таблицы, смотрите в разделе об операторе CREATE VIEW в руководстве *SQL Reference*.

Установление прав собственности на план или пакет

Команды BIND и PRECOMPILE создают или изменяют пакет прикладной программы. В каждой из них можно опцией OWNER назвать владельца создаваемого пакета. Есть несложные правила именования владельца пакета:

- Всякий пользователь может назвать владельцем себя самого. Это значение используется по умолчанию, если для опция OWNER не задана.
- ID с полномочиями SYSADM или DBADM может назвать владельцем любой ID авторизации с помощью опции OWNER.

Не все операционные системы могут связать пакет с помощью продуктов баз данных DB2, которые поддерживают опцию OWNER.

Дополнительную информацию о командах BIND и PRECOMPILE смотрите в руководстве *Command Reference*.

Предоставление косвенных привилегий посредством 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 на многих объектах 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.

Предоставление косвенных привилегий посредством пакета, содержащего 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. Когда другой пользователь захочет выполнить пакет, имея для пакета привилегию EXECUTE, этот пользователь не должен проходить дополнительных проверок привилегий в связи с оператором со ссылкой на таблицу. Однако поскольку есть оператор со ссылкой на псевдоним, пользователь, выполняющий пакет, должен пройти аутентификацию и проверку привилегий на источнике данных.

Когда файл .SQC состоит только из динамических операторов SQL и смеси ссылок на таблицы и псевдонимы, процедуры проверки авторизации DB2 для локальных объектов и псевдонимов аналогичны. 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. Дополнительную информацию о том, как DB2 обрабатывает статические и динамические SQL, смотрите в справочнике *SQL Reference*.

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 вы обращаетесь к источникам данных по псевдонимам, вы можете 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. Дополнительную информацию смотрите в разделе “Создание производной таблицы” на стр. 151.

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 изменять и просматривать всю таблицу.

  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
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

NAME	LOCATION
Plotz	Chicago
Yamaguchi	Chicago
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

Управление доступом к данным с помощью утилиты аудита

Утилита аудита DB2 генерирует и позволяет вам поддерживать файл аудита для ряда заранее определенных событий базы данных. 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.

Подробное описание утилиты аудита DB2 смотрите в разделе “Глава 6. Аудит действий DB2” на стр. 291.

Шифрование данных

Составной частью плана защиты может быть шифрование данных. Для этого можно использовать встроенные функции шифрования и дешифровки: ENCRYPT, DECRYPT_BIN, DECRYPT_CHAR и GETHINT. Дополнительную информацию об этих функциях, включая их синтаксис, смотрите в справочнике *SQL Reference*.

Функция ENCRYPT шифрует данные при помощи метода шифрования на основе паролей. Эти функции позволяют также задать подсказку к паролю. Подсказка к паролю встраивается в зашифрованные данные. После шифрования

единственная возможность расшифровки данных - это использование точного пароля. Разработчики, решившие использовать эти функции, должны запланировать мероприятия на случай забытых паролей и недоступности данных.

Результат функций ENCRYPT имеет тот же тип данных, что и первый аргумент.

Шифрование возможно 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, используемый для шифрования данных, определяется одним из двух способов:

- Аргументом пароля. Пароль - это строка, явно передаваемая при вызове функции ENCRYPT. Данные будут шифроваться и дешифроваться при помощи данного пароля.
- Паролем из специального регистра. Оператор SET ENCRYPTION PASSWORD шифрует значение пароля и посылает 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 (значение по умолчанию) специального регистра - пустая строка.

Допустимые длины паролей - от 6 до 127 включительно. Допустимые длины подсказок - от 0 до 32 включительно.

Когда специальный регистр ENCRYPTION PASSWORD задается со стороны клиента, пароль шифруется у клиента, посылается серверу базы данных, а затем расшифровывается. Чтобы не оставлять пароль в читаемом виде, его повторно шифруют на сервере базы данных. Функции DECRYPT_BIN и DECRYPT_CHAR должны расшифровывать специальный регистр перед использованием. Значение в ENCRYPTION PASSWORD также не оставляют в читаемом виде. Защита шлюза не поддерживается.

Задачи и требуемая авторизация

Не все организации одинаково распределяют обязанности между должностями. В Табл. 5 показан список некоторых распространенных названий должностей, обязанностей, обычно соответствующих этим должностям, и полномочий и привилегий, необходимых для выполнения этих обязанностей.

Таблица 5. 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.
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.

Таблица 5. Типичные названия должностей, обязанности и требуемая авторизация (продолжение)

ДОЛЖНОСТЬ	ОБЯЗАННОСТИ	ТРЕБУЕМАЯ АВТОРИЗАЦИЯ
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.

Использование системного каталога

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.

Шесть из этих производных таблиц содержат списки привилегий, предоставленных пользователям, а также 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

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. В следующем примере показано, как получить информацию о привилегиях с помощью запросов SQL:

- “Получение имен авторизации с предоставленными привилегиями”
- “Получение всех имен с полномочиями DBADM” на стр. 287
- “Получение имен с правом доступа к таблице” на стр. 287
- “Получение всех привилегий, предоставленных пользователям” на стр. 287
- “Защита производных таблиц системного каталога” на стр. 288.

Получение имен авторизации с предоставленными привилегиями

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.

Получение всех имен с полномочиями DBADM

Следующий оператор получает все имена авторизации, которым были непосредственно предоставлены полномочия DBADM:

```
SELECT DISTINCT GRANTEE FROM SYSCAT.DBAUTH
   WHERE DBADMAUTH = 'Y'
```

Получение имен с правом доступа к таблице

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.

Получение всех привилегий, предоставленных пользователям

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. Дополнительную информацию о специальных регистрах смотрите в справочнике *SQL Reference*.

Защита производных таблиц системного каталога

При создании базы данных привилегия SELECT для производных таблиц системного каталога предоставляется группе PUBLIC. (О других привилегиях, автоматически предоставляемых группе PUBLIC, смотрите в разделе “Привилегии базы данных” на стр. 264.) 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.

Нужно также исследовать столбцы, для которых собирается статистика (смотрите раздел “Статистика каталога” в руководстве *Administration Guide: Performance*). В статистике, записываемой в системный каталог, может содержаться конфиденциальная информация о вашей среде. 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
```

Глава 6. Аудит действий DB2

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 предлагается 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 генерирует и позволяет вам поддерживать файл аудита для ряда заранее определенных событий базы данных. Записи, генерируемые этой утилитой, хранятся в файле журнала аудита. Анализ этих записей позволяет выявить типичные случаи неправильного 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.
- Конфигурировать поведение утилиты аудита, в том числе выбирать категорий записываемых утилитой событий.
- 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.

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. Это число зависит также от того, используется эта утилита для успешных успешных событий, для неудачных событий или для тех и других. По этой причине важно правильно отбирать события для аудита.

Поведение утилиты аудита

Утилита аудита записывает возможные для аудита события, включая события, влияющие на экземпляры баз данных. По этой причине утилита аудита является независимой частью DB2 и остается активной, даже если останавливается экземпляр 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. Внутренний буфер используется для хранения определенного числа записей, чтобы потом записать

их на диск одной операцией. 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. Дополнительную информацию о параметре утилиты аудита ERRORTYPE и других связанных с ним параметрах смотрите в разделе “Сценарий использования возможности аудита” на стр. 295.

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.

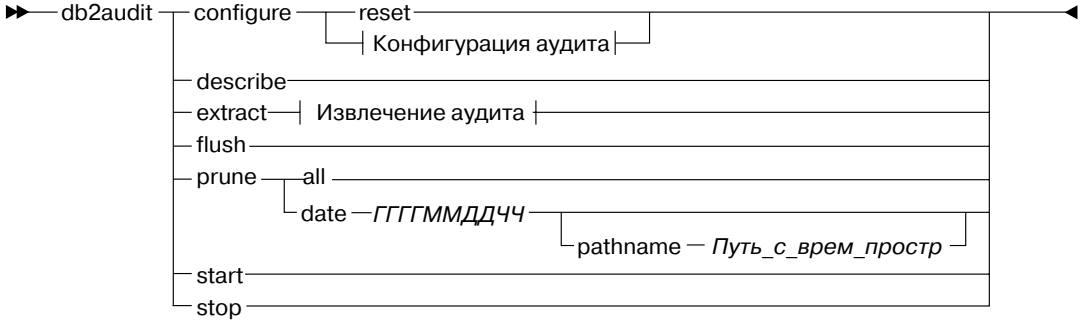
Для пакета, содержащего только статические операторы DML, единственное событие аудита, которое может генерировать запись аудита - это проверка авторизации, выясняющая наличие у пользователя привилегии на выполнение данного пакета. Проверка авторизации и возможное создание записи аудита, требуемые для операторов статического SQL в пакете, выполняется во время прекомпиляции 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.

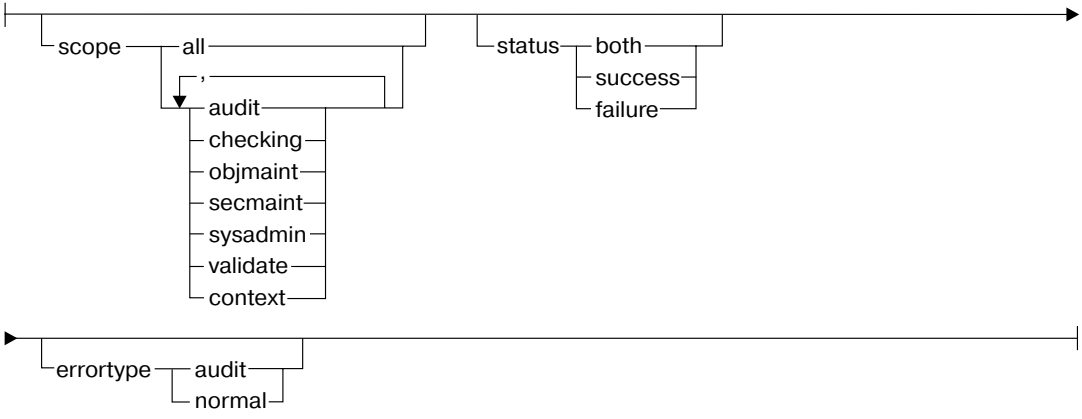
Примечание: При выполнении DDL номер раздела, записанный для всех событий в записи аудита (кроме событий контекста), будет нулевым (0) независимо от того, какой номер раздела был у оператора.

Сценарий использования возможности аудита

Посмотрите каждую часть приведенных ниже синтаксических диаграмм - это поможет вам понять, как использовать возможность аудита.



Конфигурация аудита:



Извлечение аудита:

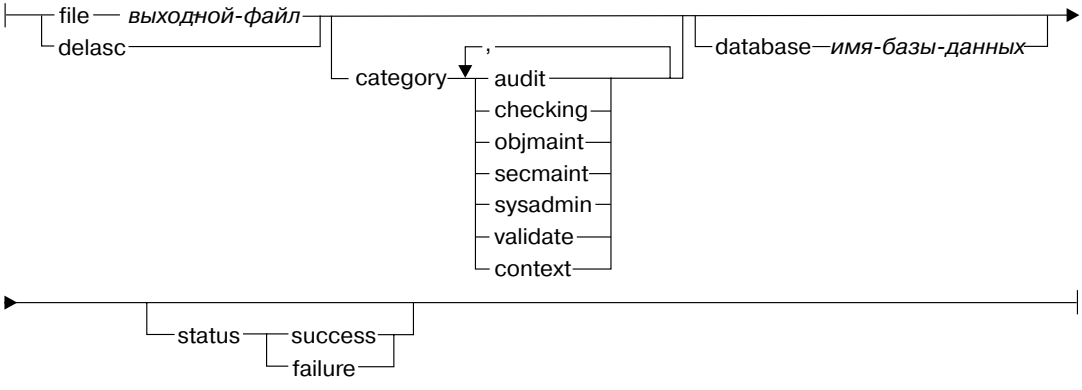


Рисунок 7. Синтаксис DB2AUDIT

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.out*, расположенный в подкаталоге `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 помещается в отдельные файлы: по одному файлу на категорию. 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). Дополнительную информацию смотрите в разделе “Советы и приемы для работы с утилитой аудита” на стр. 315.
- 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 при сокращении журнала аудита под временное пространство. 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.

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, где находится журнал, аудита нет места.

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

SQL1323N Ошибка при доступе к файлу конфигурации функции аудита.

Объяснение: Файл конфигурации аудита (db2audit.cfg) или не может быть открыт, или неверен. 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

Структура записей утилиты аудита

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.

Таблица 6. Структура 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;`		

НАЗВАНИЕ	**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. Возможные 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.

Таблица 6. Структура записей аудита для событий AUDIT (продолжение)

```
 timestamp=1998-06-24-11.54.05.151232;category=AUDIT;audit event=START;
 event correlator=0;event status=0;
  userid=boss;authid=BOSS;
```

НАЗВАНИЕ	ФОРМАТ	ОПИСАНИЕ
Authorization ID	VARCHAR (128)	ID авторизации на момент события.

Таблица 7. Структура 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;
```

НАЗВАНИЕ	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.

Таблица 7. Структура 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;
```

НАЗВАНИЕ	ФОРМАТ	ОПИСАНИЕ
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. Возможные значения: TABLE, VIEW, ALIAS, FUNCTION, INDEX, PACKAGE, DATA_TYPE, NODEGROUP, SCHEMA, STORED_PROCEDURE, 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.

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; программа/пользователь - определящий данного объекта или функции.

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.

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 выполнить или запустить программу.

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.

Таблица 8. Структура 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;
```

НАЗВАНИЕ	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. Возможные значения: CREATE_OBJECT, RENAME_OBJECT и DROP_OBJECT
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. Структура 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;
```

НАЗВАНИЕ	ФОРМАТ	ОПИСАНИЕ
Object Type	VARCHAR (32)	Тип объекта, для которого было сгенерировано событие аудита. Возможные значения: TABLE, VIEW, ALIAS, FUNCTION, INDEX, PACKAGE, DATA_TYPE, NODEGROUP, SCHEMA, STORED_PROCEDURE, 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. Структура 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;
```

НАЗВАНИЕ	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.

Таблица 9. Структура 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;
```

НАЗВАНИЕ	ФОРМАТ	ОПИСАНИЕ
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. Возможные значения: TABLE, VIEW, ALIAS, FUNCTION, INDEX, PACKAGE, DATA_TYPE, NODEGROUP, SCHEMA, STORED_PROCEDURE, 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.

Таблица 9. Структура 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;
```

НАЗВАНИЕ	ФОРМАТ	ОПИСАНИЕ
Privilege or Authority	CHAR(18)	Указывает тип предоставляемой или отзываемой привилегии или полномочий. Допустимые значения показаны в списке после таблицы.

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.

0x0000000000008000 Package BIND
Предоставлена или отозвана привилегия BIND для пакета.

0x0000000000010000 Package EXECUTE
Предоставлена или отозвана привилегия EXECUTE для пакета.

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.

Таблица 10. Структура 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;		

НАЗВАНИЕ	**ФОРМАТ**	**ОПИСАНИЕ**
Timestamp	CHAR(26)	Дата и время события аудита.
Category	CHAR(8)	Категория события аудита.Возможны следующие значения: SYSADMIN
Audit Event	VARCHAR (32)	Конкретное событие аудита. Допустимые значения показаны в списке 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.

Ниже приведен список возможных событий аудита SYSADMIN:

Таблица 11. 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

Таблица 12. Структура 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;
```

НАЗВАНИЕ	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. Структура 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);
```

НАЗВАНИЕ	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 показаны в списке 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.
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.

Ниже приведен список возможных событий аудита CONTEXT:

Таблица 14. 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

Советы и приемы для работы 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). При этом объект остается может быть в результате не создан, или действия GRANT или REVOKE - не выполнены. В этом случае имеет смысл использовать события CONTEXT. Записи аудита события CONTEXT, особенно оператор в конце события, укажут, чем завершилась предпринятая операция.

При извлечении записей аудита в формате ASCII с ограничителями, удобном для загрузки в реляционную таблицу DB2, необходимо точно указать используемый ограничитель 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”.

Управление действиями утилиты аудита DB2

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 крайняя ситуация показана здесь только для демонстрации возможностей, и мы не рекомендуем так конфигурировать утилиту аудита.

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;
application name=testapp;auth type=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;
application name=testapp;auth type=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;

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. В этом случае утилиту аудита можно сконфигурировать так:

```
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;
application name=testapp;auth type=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;
application name=testapp;auth type=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. Изменяя конфигурацию аудита, вы можете управлять типом и характером генерируемых записей.

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)

Часть 4. Перемещение данных

Глава 7. Утилиты для перемещения данных

Утилита LOAD перемещает данные в таблицы, расширяет существующие индексы и генерирует статистику. При перемещении больших объемов данных LOAD работает гораздо быстрее, чем утилита IMPORT. Данные, выгруженные утилитой EXPORT, можно загрузить при помощи утилиты LOAD.

Утилита AutoLoader разделяет большие большие объемы данных и загружает разделенные данные в различные разделы многораздельной базы данных.

Утилиты IMPORT и EXPORT перемещают данные между таблицей или производной таблицей и другой СУБД или программой электронных таблиц; между базами данных DB2; между базами данных DB2 и базами данных хостов при помощи DB2 Connect.

Data Replication (старое название - DataPropagator Relational (DPROPR)) - это компонент DB2 Universal Database, который позволяет автоматически копировать изменения таблиц в другие таблицы в других реляционных базах данных DB2.

Примечание: Вся информация по данным темам и связанные с ней 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.

Дополнительную информацию о репликации смотрите в руководстве *Replication Guide and Reference*.

Часть 5. Восстановление

Глава 8. Восстановление базы данных

Если происходят неприятности, вам понадобится возможность восстановить базу данных. Возможные ошибки - отказы питания, ошибки программ, носителей и аппаратуры. Чтобы обеспечить возможность восстановления в подобных случаях, вам понадобятся резервные копии всей базы данных или всех ее табличных пространств. Если с базой данных что-либо случается, эти резервные копии используются для ее восстановления.

Воссоздание базы данных после повреждения называется восстановлением. Возможность восстановления после сбоя позволяет автоматически восстановить базу данных после ошибки, если это возможно. Восстановление после сбоев защищает базу данных от возможности остаться в несогласованном или непригодного к использованию состоянии. При отказе транзакция, применяемая к базе данных, может остаться незавершенной. Восстановление после сбоев выполняет либо откат для незавершенной транзакции, либо принятие для завершенной транзакции. Эти действия переводят базу данных в согласованное и пригодное к использованию состояние.

Есть еще два варианта восстановления, когда база данных повреждена и вы не знаете, уцелело ли ее содержание. При повреждении базы данных существует два способа ее восстановления: восстановление версии и восстановление с повтором транзакций. Если вы работаете с базой данных только для чтения или если транзакции, записанные в базу данных, вас не интересуют, достаточно применить восстановление версии. Если вы сняли с базы данных резервную копию, вы можете восстановить по ней эту базу данных. Это называется восстановлением версии. Если вы работаете с базой данных, к которой применяются транзакции, и необходимо восстановить все изменения, которые внесли эти транзакции, надо выполнить восстановление с повтором транзакций. Восстановление с повтором транзакций включает в себя восстановление резервной копии базы данных. После этого к ней применяются записи журнала, где регистрируются транзакции для этой базы данных. При этом повторяются все действия с базой данных, в результате чего она переводится в состояние непосредственно перед моментом отказа. При этом методе никакие изменения в базе данных не будут утеряны. Для этого метода восстановления решающую роль имеет запись транзакций в журнал.

Примечание: Вся информация по данным темам и связанные с ней вопросы из книг *Command Reference* и *Administrative API Reference* собраны в книге *Data Recovery and High Availability Guide and Reference*.

Книга *Data Recovery and High Availability Guide and Reference* - это основной источник информации по данным темам.

Часть 6. Приложения

Приложение A. Правила именования

Перейдите к разделу, описывающему требуемые вам правила именования для:

- “Общие правила именования”
- “Правила именования объектов”
- “Сохранение регистрозависимых значений в системе объединения” на стр. 336

Общие правила именования

Если не сказано иного, все имена могут содержать следующие символы:

- От A до Z. В большинстве имен символы от A до Z преобразуются из строчных в прописные.
- От 0 до 9
- @, #, $ и _ (подчеркивание)

Имена не могут начинаться с цифры или с символа подчеркивания.

Не используйте зарезервированные слова SQL для имен таблиц, производных таблиц, столбцов, индексов или ID авторизации. Список зарезервированных слов языка SQL приведен в справочнике *SQL Reference*.

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.

Правила именования объектов

На все объекты распространяются общие правила именования. Кроме того, для некоторых объектов есть дополнительные ограничения, описанные ниже.

Таблица 15. 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 положения, в котором они занесены в каталог. В реализациях DB2 для систем на основе UNIX положением является путь, а в реализациях под Windows - буква диска. • 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 системах Windows NT и Windows 2000 имя экземпляра не должно совпадать с именем какой-либо службы. **Примечание:** Во избежание возможных проблем не используйте символы @, # и $ в именах баз данных, к которыми предполагается обращаться по связи. Кроме того, поскольку эти символы имеются не на всех клавиатурах, не используйте их, если с базой данных будут работать на другом языке.

Таблица 16. Правила именования объектов баз данных

Объекты	Рекомендации
• Алиасы • Пулы буферов • Столбцы • Мониторы событий • Индексы • Методы • Группы узлов • 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 столбцов, которые могут содержать до 30 байт • Имен схем, которые могут содержать до 30 байт • Имена объектов могут также содержать: – допустимые символы национальных алфавитов (например, ö) – многобайтные символы, за исключением многобайтных пробелов (в многобайтных средах)

Дополнительная информация об именах схем

- Таблицы с именами схем, длина которых больше 18 байт, нельзя реплицировать.
- 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 программ объявит временную таблицы 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.

Таблица 17. Правила именования пользователей, ID пользователей и групп

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 настоящее время в Windows NT и в Windows 2000 имена не могут быть длиннее 20 символов. • При использовании аутентификации DCE имена пользователей не могут быть длиннее 8 символов. • Если не используется аутентификация DCE или аутентификация клиента, соединения клиентов из других систем (не 32-битных систем Windows) с Windows NT и Windows 2000 с именами пользователей длиннее 8 символов поддерживаются, если имя пользователя и пароль заданы в явном виде. • Имена и ID не должны: – Совпадать с именами USERS, ADMINS, GUESTS, PUBLIC, LOCAL, а также с зарезервированными словами SQL, перечисленными в справочнике *SQL Reference*. – Начинаться с IBM, SQL или SYS. – Содержать символы национальных алфавитов. В системах на основе UNIX у групп и у пользователей могут быть одинаковые имена. В операторе GRANT надо указать, имеете ли вы в виду группу или же пользователя. В операторе REVOKE необходимость задания пользователя или группы зависит от того, есть ли в таблицах каталогов авторизации строки с одинаковыми GRANTEE, но разными значениями GRANTEETYPE. В системах Windows NT у локальных групп, глобальных групп и у пользователя не может быть одинаковых имен. В системах OS/2 у группы и у пользователя не может быть одинаковых имен. **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. Дополнительную информацию смотрите в описании оператора CONNECT в справочнике *SQL Reference*.

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 есть способ изменить и проверить пароль с клиента. Например, DB2 for OS/390 Версии 5 поддерживает следующий метод изменения пароля пользователя. Если вы получили сообщение SQL1404N “Срок действия пароля истек”, измените пароль при помощи оператора CONNECT так:

```
CONNECT TO <база_данных> USER <id_пользователя> USING <пароль>
   NEW <новый_пароль> CONFIRM <новый_пароль>
```

Пароль можно изменить также при помощи диалогового окна “Изменение пароля” Ассистента конфигурирования клиента DB2. Дополнительные сведения об этих методах изменения пароля смотрите в справочнике *SQL Reference* и в электронной справке Ассистента конфигурирования клиента.

Таблица 18. Правила именования объектов баз 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) – В многобайтных средах - многобайтные символы, за исключением многобайтных пробелов.

Использование идентификаторов с ограничителями в качестве имен объектов

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. Подробности об ограничителях смотрите в разделе ″Идентификаторы SQL″ справочника *SQL Reference*.

Сохранение регистрозависимых значений в системе объединения

В распределенных требованиях бывает необходимым задание идентификаторов и паролей, которые на источнике данных рассматриваются как регистрозависимые. Чтобы они правильно передавались на источник данных, выполняйте следующие правила:

- Задавайте идентификаторы и пароли в нужном регистре, заключая в двойные кавычки.
- При задании ID пользователя установите для опции сервера *fold_id* значение ″n″ (“Нет, не менять регистр”) для этого источника данных. При задании пароля установите для опции сервера *fold_pw* значение ″n″ для этого источника данных.

  Есть и другие варианты для ID пользователей и паролей. Если источник данных требует задавать ID пользователя в нижнем регистре, вы можете задать его в любом регистре и установить для опции сервера *fold_id* значение ″l″ (“Посылать ID на источник данных в нижнем регистре”). Если источник данных требует задавать ID пользователя в верхнем регистре, вы можете задать его в любом регистре и установить для опции сервера *fold_id* значение ″u″ (“Посылать ID на источник данных в верхнем регистре”). Аналогичным образом, если требуется задавать пароль в нижнем или в верхнем регистре, это можно сделать, задав для опции сервера *fold_pw* значение ″l″ или ″u″.

  Дополнительную информацию об опциях сервера смотрите в разделе “Использование опций сервера для определения характеристик источников данных и настройки процесса аутентификации” на стр. 161.
- Если вы заключаете регистрозависимый идентификатор или пароль в двойные кавычки в командной строке операционной системы, необходимо убедиться в правильности анализа системой этих двойных кавычек. Для этого:
  - В операционной системе на основе UNIX - заключите весь оператор в одинарные кавычки.
  - В операционной системе Windows NT - перед каждой кавычкой ставьте обратную косую черту.

  Например, многие идентификаторы с ограничителями на источниках данных DB2 регистрозависимы. Предположим, вы хотите создать псевдоним `NICK1` для производной таблицы DB2 for CS `"my_schema"."wkly_sal"`, которая находится на источнике данных под названием `NORBASE`.

  В командной строке системы на основе UNIX введите:

  ```
  db2 'create nickname nick1 for norbase."my_schema"."wkly_sal"'
  ```

  В командной строке Windows NT введите:

  ```
  db2 create nickname nick1 for norbase.\"my_schema\".\"wkly_sal\"
  ```

Если вы вводите оператор в командной строке DB2 (интерактивный режим) или задаете его в программе, одинарные кавычки или обратные косые черты не нужны. Например, в командной строке DB2, независимо от операционной системы, нужно ввести:

```
create nickname nick1 for norbase."my_schema"."wkly_sal"
```

Приложение B. Использование служб каталогов среды DCE

DCE поддерживает службу CDS (Cell Directory Service - служба каталога ячейки) и службу GDS (Global Directory Service - служба глобального каталога). Подробную информацию об этих службах и основных понятиях DCE смотрите в руководстве *Introduction to OSF DCE*. Функция DB2 для служб каталогов DCE поддерживает только CDS. Эта поддержка позволяет пользователю не повторять создание базы данных, узла или базы данных DCS на всех клиентах. Служба CDS DCE централизует эту информацию.

В следующих разделах описана настройка и доступ к базам данных с помощью служб каталогов DCE:

- Создание объектов каталогов
- Атрибуты каждого класса объектов
- Защита служб каталогов
- Параметры конфигурации и переменные реестра
- Команды CATALOG и ATTACH и оператор CONNECT
- Как клиент устанавливает соединение с базой данных
- Как происходит просмотр каталогов
- Временное переопределение информации о каталогах DCE
- Задачи служб каталогов
- Ограничения служб каталогов

Некоторые клиенты DB2 не поддерживают службы каталогов DCE. Если клиент DB2 поддерживает службы каталогов DCE, дополнительная информация приводится в соответствующем руководстве *Быстрый старт*.

Создание объектов каталогов

Администратор баз данных должен создавать объекты трех типов:

- “Объекты баз данных” на стр. 340
- “Объекты локаторов баз данных” на стр. 341
- “Объекты маршрутной информации” на стр. 342

У каждого объекта есть атрибуты. Их полное описание смотрите в разделе “Атрибуты каждого класса объектов” на стр. 344.

Чтобы администратор баз данных мог создавать объекты, администратор DCE должен добавить информацию о базе данных в таблицу CDS и дать администратору баз данных привилегии для создания. Подробности смотрите в разделе “Задачи администратора DCE” на стр. 360.

Объекты баз данных

Для каждой базы данных назначения требуется объект баз данных. Имя этого объекта составляется из имен ячейки, каталога и базы данных, например:

```
/.../cell_name/dir_name1/dir_name2/OBJ_NAME
```

Примечание: При задании имен базы данных следуйте приведенным ниже рекомендациям. Имя не должно быть длиннее 8 символов и должно состоять из символов верхнего регистра. Если имя длиннее 8 символов или содержит символы нижнего регистра, надо с помощью команды CATALOG GLOBAL DATABASE назначить алиас. Подробности об этой команде смотрите в разделе “команда CATALOG GLOBAL DATABASE” на стр. 352.

Ниже приводится пример объекта баз данных. Объект, расположенный в каталоге DCE, содержит также отметку времени и другую информацию. Буквы слева от атрибутов показывают, является ли атрибут обязательным - R, необязательным - O или комментарием - C.

```
 Object name:         /.../CELL_TORONTO/subsys/database/AIXDB1
R  DB_Object_Type:           D
C  DB_Product_Name:           DB2_for_AIX
C  DB_Product_Release:       V7R1M000
R  DB_Native_Database_Name:  AIXDBASE
R  DB_Database_Protocol:     DB2RA
R  DB_Authentication:        CLIENT
O  DB_Communication_Protocol:
O  DB_Database_Locator_Name: /.../CELL_TORONTO/subsys/database/AIX_INST
C  DB_Comment:               Test_database_on_AIX
```

Если с экземпляром менеджера баз данных связаны также другие базы данных, объект баз данных должен содержать имя объекта локатора баз данных; протокол связи должен быть пустым. Имя локатора баз данных - это полное имя экземпляра менеджера баз данных или экземпляра DB2 Connect.

Ниже приводится пример команд DCE, создающих этот объект. Предварительно администратор DCE должен выполнить шаги, описанные в разделе “Задачи администратора DCE” на стр. 360.

Сначала в файле *cdscp.inp* необходимо ввести:

```
create object /.:/subsys/database/AIXDB1

add object /.:/subsys/database/AIXDB1 DB_Object_Type            = D
add object /.:/subsys/database/AIXDB1 DB_Product_Name           = DB2_for_AIX
add object /.:/subsys/database/AIXDB1 DB_Product_Release        = V7R1M000
```

```
add object /.:/subsys/database/AIXDB1 DB_Native_Database_Name   = AIXDBASE
add object /.:/subsys/database/AIXDB1 DB_Database_Protocol      = DB2RA
add object /.:/subsys/database/AIXDB1 DB_Authentication         = CLIENT
add object /.:/subsys/database/AIXDB1 DB_Database_Locator_Name  = /...
/CELL_TORONTO/subsys/database/AIX_INST
add object /.:/subsys/database/AIXDB1 DB_Comment      = Test_database_on_AIX
```

Затем надо выполнить одну из следующих команд:

- dcelogin принципал пароль (в системе OS/2) или
- dce_login принципал пароль (в операционных системах UNIX и Windows).

После этого надо выполнить команду

- cdscp < cdscp.inp

Следующая команда выводит объект:

```
cdscp show object /.:/subsys/database/AIXDB1
```

Если с экземпляром менеджера баз данных не связаны другие базы данных, объект баз данных должен содержать имя протокола связи, а имя локатора баз данных должно быть пустым. Например:

```
  Object name:                 /.../CELL_TORONTO/subsys/database/MVSDB
R  DB_Object type:              D
C  DB_Product_Name:             DB2_for_MVS
C  DB_Product_Release:          V7R1M00
R  DB_Native_Database_Name:     MVSDBASE
R  DB_Database_Protocol:        DRDA
R  DB_Authentication:           SERVER
O  DB_Communication_Protocol:   APPC;NET1;TARGETLU1;DB2DRDA;MODE1;PROGRAM
O  DB_Database_Locator_Name:
C  DB_Comment:                  Test_database_on_MVS
```

Объекты локаторов баз данных

Эти объекты уточняют информацию обо всех протоколах связи, используемых экземпляром СУБД или DB2 Connect. Требуется по одному объекту локаторов баз данных:

- Для каждого экземпляра, связанного и с СУБД, и с DB2 Connect
- Для каждого экземпляра СУБД, связанного с несколькими базами данных, но не связанного с DB2 Connect
- Для каждого экземпляра DB2 Connect, не связанного с СУБД.

Имя объекта состоит из имен ячейки, каталога и короткого имени экземпляра базы данных, например:

```
/.../cell_name/dir_name1/dir_name2/AIX_INST
```

Примечание: Если для экземпляра выполняется команда ATTACH, короткое имя должно содержать не более 8 символов верхнего регистра.

Ниже приводится пример объекта локаторов баз данных. Объект, расположенный в каталоге DCE, содержит также отметку времени и другую информацию. Буквы слева от атрибутов показывают, является ли атрибут обязательным - R, необязательным - O или комментарием - C.

```
  Object name:               /.../CELL_TORONTO/subsys/database/AIX_INST
R  DB_Object_Type:            L
C  DB_Product_Name:           DB2_for_AIX
C  DB_Product_Release:        V7R1M00
R  DB_Communication_Protocol: TCPIP;HOSTNAME1;1234
R  DB_Communication_Protocol: APPC;NET1;TARGETLU1;TPN1;MODE;PROGRAM
C  DB_Comment:                Test_instance_on_AIX
```

Если атрибут задан и для объекта баз данных, и для объекта локаторов баз данных, используется значение, заданное в объекте баз данных.

Ниже приводится пример команд DCE, создающих этот объект. Предварительно администратор DCE должен выполнить шаги, описанные в разделе “Задачи администратора DCE” на стр. 360.

Сначала в файле *cdscp.inp* необходимо ввести:

```
create object /.:/subsys/database/AIX_INST

add object /.:/subsys/database/AIX_INST DB_Object_Type            = L
add object /.:/subsys/database/AIX_INST DB_Product_Name           = DB2_for_AIX
add object /.:/subsys/database/AIX_INST DB_Product_Release        = V7R1M00
add object /.:/subsys/database/AIX_INST DB_Communication_Protocol = TCPIP;
HOSTNAME1;1234
add object /.:/subsys/database/AIX_INST DB_Communication_Protocol = APPC;NET1;
TARGETLU;TPN1;MODE;PROGRAM
add object /.:/subsys/database/AIX_INST DB_Comment     = Test_instance_on_AIX
```

Затем надо выполнить одну из следующих команд:

- dcelogin принципал пароль (в системе OS/2) или
- dce_login принципал пароль (в операционных системах UNIX и Windows).

После этого надо выполнить команду

- cdscp < cdscp.inp

Следующая команда выводит объект:

```
cdscp show object /.:/subsys/database/AIX_INST
```

Объекты маршрутной информации

Объекты маршрутной информации требуются для доступа к хосту. Если протоколы баз данных, используемые клиентом и базой данных назначения, не согласуются, объект маршрутной информации указывает клиенту, какой экземпляр DB2 Connect использовать. Каждой базе данных назначения соответствует атрибут, включающий доступные протоколы и имя объекта

локаторов баз данных для экземпляра DB2 Connect. Имя объекта состоит из имени ячейки, имени каталога и короткого уникального имени, например:

```
/.../cell_name/dir_name1/dir_name2/ROUTE1
```

Ниже приводится пример объекта маршрутной информации. Объект, расположенный в каталоге DCE, содержит также отметку времени и другую информацию. Буквы слева от атрибутов показывают, является ли атрибут и все его элементы обязательными - R, необязательными - O или комментарием - C.

Группу клиентов 1 составляют Client_1, Client_2 и Client_3, как в разделе рис. 8 на стр. 354.

```
  Object name:    /.../CELL_TORONTO/subsys/database/ROUTE1
R  DB_Object_Type: R
C  DB_Comment:     Routing_for_client_group_1

R  DB_Target_Database_Info
   R  Database name                 = /.../CELL_TORONTO/subsys/database/MVSDB
   R  Outbound protocol from router = DRDA
   R  Inbound protocol to router    = DB2RA
   R  Authenticate at gateway       = 1
   O  Parameter string              = NOMAP,D,INTERRUPT_ENABLED
   R  DB_Database_Locator_Name      = /.../CELL_TORONTO/subsys/database/GW_INST

R  DB_Target_Database_Info
   R  Database name                 = *OTHERDBS
   R  Outbound protocol from router = DRDA
   R  Inbound protocol to router    = DB2RA
   R  Authenticate at gateway       = 0
   O  Parameter string              =
   R  DB_Database_Locator_Name      = /.../CELL_TORONTO/subsys/database/OTH_INST
```

Имя базы данных *OTHERDBS - специальное значение, означающее общий маршрутизатор, используемый для доступа к базам данных назначения, которые не определены в явном виде в объекте маршрутной информации.

Ниже приводится пример команд DCE, создающих этот объект. Обратная косая черта (\) означает продолжение строки.

Предварительно администратор DCE должен выполнить шаги, описанные в разделе “Задачи администратора DCE” на стр. 360.

Сначала в файле *cdscp.inp* необходимо ввести:

```
create object /.:/subsys/database/ROUTE1

add object /.:/subsys/database/ROUTE1 DB_Object_Type = R
add object /.:/subsys/database/ROUTE1 DB_Comment = Routing_for_client_group_1
add object /.:/subsys/database/ROUTE1 DB_Target_Database_Info = \
   /.../CELL_TORONTO/subsys/database/MVSDB;\
drda;db2ra;1;NOMAP,D,INTERRUPT_ENABLE;\
   /.../CELL_TORONTO/subsys/database/GW_INST
```

```
add object /.:/subsys/database/ROUTE1 DB_Target_Database_Info = \
*OTHERDBS;drda;db2ra;0;;\
/.../CELL_TORONTO/subsys/database/OTH_INST
```

Затем надо выполнить одну из следующих команд:

- dcelogin принципал пароль (в системе OS/2) или
- dce_login принципал пароль (в операционных системах UNIX и Windows).

После этого надо выполнить команду

- cdscp < cdscp.inp

Следующая команда выводит объект:

```
cdscp show object /.:/subsys/database/ROUTE1
```

Дополнительную информацию о командах DCE смотрите в следующих книгах по OSF DCE:

- *OSF DCE Administration Guide*
- *OSF DCE Administration Reference*

Атрибуты каждого класса объектов

В среде DCE каждый объект и атрибут объекта определяется с помощью ID объекта (OID). Все OID присваиваются иерархией организаций, высшей из которых является Международная организация по стандартизации (ISO).

В Табл. 19 показаны атрибуты каждого класса объектов, а в Табл. 20 - их характеристики.

Таблица 19. Атрибуты классов объектов

Класс объекта	ID объекта (OID)	Обязательные атрибуты	Необязательные атрибуты
(DB) Database_Object	1.3.18.0.2.6.12	DAU, DOT, DDP, DNN	DCO, DPN, DRL, DLN, DCP, DPR
(DL) Database_Locator_Object	1.3.18.0.2.6.13	DOT, DCP	DCO, DPN, DRL
(RI) Routing_Information_Object	1.3.18.0.2.6.14	DOT, DTI	DCO, DPN, DRL

Таблица 20. Характеристики атрибутов классов объектов

Имя атрибута	OID	Минимальная длина	Максимальная длина	Синтак-сис
(DAU) DB_Authentication	1.3.18.0.2.4.39	1	1024	Char

Таблица 20. Характеристики атрибутов классов объектов (продолжение)

Имя атрибута	OID	Минимальная длина	Максимальная длина	Синтаксис
(DCO) DB_Comment	1.3.18.0.2.4.30	1	1024	Char
(DCP) DB_Communication_Protocol	1.3.18.0.2.4.31	1	1024	Char
(DDP) DB_Database_Protocol	1.3.18.0.2.4.32	1	1024	Char
(DLN) DB_Database_Locator_Name	1.3.18.0.2.4.33	1	1024	Char
(DNN) DB_Native_Database_Name	1.3.18.0.2.4.34	1	1024	Char
(DOT) DB_Object_Type	1.3.18.0.2.4.35	1	1	Char
(DPN) DB_Product_Name	1.3.18.0.2.4.36	1	1024	Char
(DRL) DB_Product_Release	1.3.18.0.2.4.37	1	1024	Char
(DTI) DB_Target_Database_Info	1.3.18.0.2.4.38	1	1024	Char
(DPR) DB_Principal	1.3.18.0.2.4.63	1	1024	Char
Примечание: Атрибуты DCP, DDP и DTI могут иметь несколько значений. Все остальные должны иметь по одному значению.				

Подробное описание всех атрибутов

В данном разделе описаны все атрибуты.

Примечание: Служба каталогов DCE не проверяет правильность значений для DB2. Убедитесь в правильности значений и в том, что не пропущены обязательные атрибуты.

DB_Authentication (DAU)
Метод подтверждения, требуемый объектом. Этот атрибут является обязательным для объектов баз данных сервера DB2. Разрешенные значения - CLIENT, SERVER и DCE.

DB_Principal (DPR)
Если метод подтверждения - “DCE”, задайте в качестве значения этого атрибута принципал DCE.

DB_Comment (DCO)
Только для документирования.

DB_Communication_Protocol (DCP)
Каждое значение этого атрибута состоит из элементов, описывающих поддерживаемый сетевой протокол. Примеры сетевых протоколов - TCP/IP, APPC, IPX/SPX и NetBIOS. Элементы разделяются точками с запятыми. Не ставьте между ними пробелов.

- Элементы для TCP/IP:
  1. tcpip
  2. Имя хоста узла назначения

3. Номер порта, на котором объект ожидает входящие требования соединения TCP/IP
4. (Необязательно) Защита - NONE или SOCKS.

Например: `tcpip;HOSTNAME;1234`

- Элементы для APPC:
  1. appc
  2. Сетевой идентификатор места назначения, куда относится объект.
  3. Имя LU, где расположено место назначения.
  4. Имя программы транзакций (TPN), представляющее объект на LU (Для DB2 for MVS/ESA используйте DB2DRDA).
  5. Имя режима
  6. Тип защиты на месте назначения. Возможны следующие значения:
     - NONE
     - PROGRAM
     - SAME

  Например: `appc;NETID;TARGETLU;TPNAME;MODE;PROGRAM`

  Примечание: В протоколе APPC клиент должен использовать в качестве имени LU имя своей локальной точки управления (CP).
- (Только в OS/2 и поддерживаемых операционных системах Windows) Элементы для IPX/SPX:
  1. ipxspx
  2. Имя файл-сервера
  3. Имя объекта

  Например: `ipxspx;SVR_NAME;OBJ_NAME`
- (Только в OS/2 и поддерживаемых операционных системах Windows) Элементы для NetBIOS:
  1. netbios
  2. Имя узла сервера

  Например: `netbios;SVR_NNME`, где номером адаптера клиента служит либо значение реестра *db2clientadpt*, либо параметр конфигурации менеджера баз данных *dft_client_adpt*.
- (Только в поддерживаемых операционных системах Windows) Элементы для NPIPE:
  1. NPIPE
  2. Имя компьютера сервера

3. Имя экземпляра сервера

Например: `npipe;computername;instance`

DB_Database_Protocol (DDP)
Протокол или несколько протоколов баз данных, поддерживаемых базой данных назначения. Примеры значений - DB2RA и DRDA. Ниже приводятся команды *cdscp* для добавления этих протоколов:

```
add object /.:/subsys/database/AIXDB1 DB_Database_Protocol db2ra
add object /.:/subsys/database/AIXDB1 DB_Database_Protocol drda
```

DB_Database_Locator_Name (DLN)
Имя DCE объекта локатора базы данных. Для объекта базы данных - имя экземпляра СУБД. Для объектов маршрутной информации - имя экземпляра DB2 Connect.

Например, `/.../CELL_TORONTO/subsys/database/AIX_INST`

DB_Native_Database_Name (DNN)
Имя или алиас, под которым база данных определена в пределах своего экземпляра. Это имя используют для связи с базой данных в пределах экземпляра локальные программы.

Для баз данных DB2 UDB длина имени не превышает 8 символов. Для остальных баз данных длина имени может быть другой. Например, имена баз данных в DB2 for MVS/ESA могут содержать до 18 символов.

DB_Object_Type (DOT)
Тип объекта. Этот атрибут является обязательным для всех объектов; его возможные значения:

D Объект базы данных

L Объект локатора базы данных

R Объект маршрутной информации

DB_Product_Name (DPN)
Название продукта. Только для документирования.

DB_Product_Release (DRL)
Выпуск продукта. Только для документирования.

DB_Target_Database_Info (DTI)
Каждое значение этого атрибута состоит из определенного числа элементов, перечисленных через точку с запятой. Не ставьте между ними пробелов. Порядок элементов:

1. Имя базы данных. Имя DCE базы данных назначения, для которой производится маршрутизация. Значение *OTHERDBS означает шлюз по умолчанию для баз данных назначения, не определенных в объекте маршрутной информации.

2. Исходящий протокол маршрутизатора. Протокол баз данных, используемый базой данных назначения, или протокол баз данных, который использует для связи с ней экземпляр DB2 Connect, осуществляющий маршрутизацию. Например, DRDA.
3. Входящий протокол маршрутизатора. Протокол баз данных, который принимается экземпляром DB2 Connect, осуществляющим маршрутизацию.Например, DB2RA.
4. Подтверждение на шлюзе. Допустимые значения - 0 и 1. Подробности смотрите в разделе Табл. 21 на стр. 349.
5. Строка параметров, описывающая шлюз DB2 Connect. Элементы в строке должны следовать в описанном ниже порядке. Они разделяются запятыми. Если какой-либо элемент не указан, используется значение по умолчанию.
   - Имя файла отображения. Полное имя файла отображения SQLCODE, заменяет отображение SQLCODE по умолчанию. Чтобы отключить отображение SQLCODE, задайте значение NOMAP.
   - D. Программа разрывает соединение с базой данных сервера DRDA, если возвращены определенные коды SQL. Подробности о кодах SQL смотрите в разделе *DB2 Connect. Руководство пользователя*.
   - INTERRUPT_ENABLED. DB2 Connect разорвет соединение и выполнит откат единицы работы, если клиент, подключенный к серверу DRDA, пошлет прерывание.

   Ниже приводится несколько примеров:

   ```
   NOMAP
   /u/username/sqllib/map/dcs1new.map,D
   /u/username/sqllib/map/dcs1new.map,D,INTERRUPT_ENABLED
   ```

   Используя значения по умолчанию, ставьте запятую, чтобы сохранить порядок элементов, например:

   ```
   ,D
   ```

   или

   ```
   ,,INTERRUPT_ENABLED
   ```

   Подробности о строке параметров смотрите в разделе *DB2 Connect. Руководство пользователя*.
6. Имя DCE экземпляра DB2 Connect, осуществляющего маршрутизацию.

Ниже приведен пример атрибута DB_Target_Database_Info:

```
/.../CELL_TORONTO/subsys/database/MVSDB;\
drda;db2ra;0;;\
/.../CELL_TORONTO/subsys/database/GW_INST
```

Примечание: Обратная косая черта (\) - знак продолжения строки.

Защита служб каталогов

Если службы каталогов DCE используются в среде, где нет шлюза DB2 Connect, аутентификация выполняется так же, как и для других клиентов, обращающихся к серверам баз данных. Дополнительную информацию смотрите в разделе “Выбор метода аутентификации для сервера” на стр. 239.

Если службы каталогов DCE используются в среде, где есть шлюз DB2 Connect, администратор DB2 Connect определяет, где происходит проверка имен пользователей и паролей. Для каталогов DCE задайте:

- Тип защиты протокола связи в объекте локатора баз данных, представляющем рабочую станцию DB2 Connect. (Если удаленный клиент связывается со шлюзом DB2 Connect Extended Edition с использованием протокола APPC, в *объекте локатора DCE* этого шлюза следует указать тип защиты NONE).
- Тип аутентификации на объекте базы данных.
- Тип защиты протокола связи на объекте базы данных (или на связанном с ним объекте локатора).
- Элемент аутентификации на шлюзе для объекта маршрутной информации.

В Табл. 21 показаны возможные сочетания этих значений и соответствующие места аутентификации для соединений APPC. Сочетания, показанные в этой таблице, поддерживаются DB2 Connect со службами каталогов DCE.

Таблица 21. Возможные сценарии защиты при использовании DCE для соединений APPC

	Объект базы данных сервера		**Объект маршрутизатора**	**Проверка**
Вариант	**Аутентификация**	**Защита**	**Аутентификация на шлюзе**	
1	CLIENT	SAME	0	Удаленный клиент (или рабочая станция DB2 Connect)
2	CLIENT	SAME	1	Рабочая станция DB2 Connect
3	SERVER	PROGRAM	0	сервер DRDA
4	SERVER	PROGRAM	1	Рабочая станция DB2 Connect и сервер DRDA
5	DCE	NONE	Не применяется	DCE

В Табл. 22 показаны возможные сочетания этих значений и соответствующие места аутентификации для соединений TCP/IP. Сочетания, показанные в этой таблице, поддерживаются DB2 Connect со службами каталогов DCE.

Таблица 22. Возможные сценарии защиты при использовании DCE для соединений TCP/IP

Вариант	Аутентификация	Аутентификация на шлюзе	Проверка
1	CLIENT	0	Клиент
2	CLIENT	1	Рабочая станция DB2 Connect
3	SERVER	0	сервер DRDA
4	Не применяется	Не применяется	Нет
5	DCE	Не применяется	DCE

Все сочетания применимы и для APPC, и для TCP/IP и подробно описаны ниже:

1. Имя пользователя и пароль проверяются только на удаленном клиенте. (Для локальных клиентов пароль и имя пользователя проверяются только на рабочей станции DB2 Connect).

   Предполагается, что аутентификация пользователя проведена на месте его первой регистрации. По сети передается ID пользователя, но не пароль. Не используйте этот тип защиты, если не все рабочие станции клиентов достаточно защищены.
2. Имя пользователя и пароль проверяются только на рабочей станции DB2 Connect. Пароль посылается по сети от удаленного клиента на рабочую станцию DB2 Connect, но не на сервер DRDA.
3. Имя пользователя и пароль проверяются только на сервере DRDA. Пароль посылается по сети от удаленного клиента на рабочую станцию DB2 Connect, а оттуда - на сервер DRDA.
4. Имя пользователя и пароль проверяются и на рабочей станции DB2 Connect, и на сервере DRDA. Пароль посылается по сети от удаленного клиента на рабочую станцию DB2 Connect, а оттуда - на сервер DRDA.

   Поскольку проверка проводится в двух местах, на рабочей станции DB2 Connect и на сервере DRDA должен использоваться одинаковый набор имен пользователей и паролей.
5. Маркер DCE дает сервер защиты DCE.

Примечания:

1. В системах на базе AIX все пользователи, использующие тип защиты `SAME`, должны принадлежать к системной группе AIX.
2. В системах на базе AIX с удаленными клиентами экземпляр DB2 Connect на рабочей станции DB2 Connect должен принадлежать к системной группе AIX.
3. Доступ к серверу DRDA регулируется его собственными подсистемами и средствами защиты, такими как VTAM и RACF (Resource Access Control

Facility - управление доступом к ресурсам). Доступ к защищенным объектам баз данных регулируется операторами SQL **GRANT** и **REVOKE**.

Параметры конфигурации и переменные реестра

Ниже перечислены параметры конфигурации для каталогов DCE. Приводятся примеры значений. Подробности смотрите в разделе “Распределенные службы” главы “Конфигурирование DB2” в книге *Administration Guide: Performance*.

- *dir_obj_name* - имя экземпляра базы данных, к которому приписывается значение пути *dir_path_name*. Если для экземпляра выполняется команда ATTACH, имя должно содержать не более 8 символов, причем только в верхнем регистре, например:

  ```
  AIX_INST
  ```

- *dir_type* указывает, использовать ли службы каталогов DCE. Чтобы службы каталогов DCE использовались, этот параметр должен иметь значение

  ```
  DCE
  ```

  Обратите внимание на то, что для клиентов баз данных, которые не поддерживают службы каталогов DCE, параметр *dir_type* имеет значение NONE, и его нельзя изменить.

- *dir_path_name* - путь к каталогу, сообщаемый администратором DCE, например:

  ```
  /.:/subsys/database/
  ```

- *route_obj_name* - необязательный параметр, имя объекта маршрутной информации служб каталогов DCE. Имя может быть полным, например:

  ```
  /.:/subsys/database/ROUTE1
  ```

  или коротким, тогда к нему будет приписано значение пути *dir_path_name*, например:

  ```
  ROUTE1
  ```

- *dft_client_comm* - необязательный параметр DCE, указывающий, какой протокол связи используется клиентом, например:

  ```
  TCPIP
  ```

  Этот параметр может также задавать несколько протоколов, например:

  ```
  TCPIP,APPC  (на платформах на базе UNIX)
  TCPIP,APPC,IPXSPX,NETBIOS    (на платформах OS/2)
  TCPIP,APPC,IPXSPX,NETBIOS,NPIPE    (в поддерживаемых операционных системах
                                      Windows)
  ```

- *dft_client_adpt* - необязательный параметр DCE, задающий номер адаптера клиента по умолчанию для протокола NetBIOS в OS/2 и поддерживаемых операционных системах Windows. Допустимый диапазон значений - от 0 до 15. Если значение этого параметра не является числом, принимается значение

по умолчанию - ноль (0). Если значение выходит за пределы допустимого диапазона, принимается значение по умолчанию - ноль (0).

Значения следующих параметров могут переопределяться значениями переменных реестра.

Параметр конфигурации	Переменная реестра
dir_path_name	DB2DIRPATHNAME
route_obj_name	DB2ROUTE
dft_client_comm	DB2CLIENTCOMM
dft_client_adpt	DB2CLIENTADPT

Значения реестра задаются так же, как соответствующие параметры конфигурации. Так, DB2CLIENTCOMM, подобно параметру *dft_client_comm*, представляет собой строку символов и может состоять из нескольких значений, разделенных запятыми, например:

```
db2set DB2CLIENTCOMM=TCPIP,APPC
```

Команды CATALOG и ATTACH и оператор CONNECT

Информация DCE задается с помощью следующих команд:

- команда CATALOG GLOBAL DATABASE
- Оператор CONNECT
- Команда ATTACH

команда CATALOG GLOBAL DATABASE

Используйте команду CATALOG GLOBAL DATABASE, когда у клиента и сервера различные пути или когда имя базы данных длиннее 8 символов или содержит символы в разных регистрах. Администратор баз данных вводит имя DCE базы данных и тип каталога DCE.

Например:

- Если различаются пути, например *dir_path_name* = */.../CELL_TORONTO/subsys/database/*:

```
CATALOG GLOBAL DATABASE
/.../CELL_VANCOUVER/subsys/database/VMDB AS VANVMDB
USING DIRECTORY DCE WITH "комментарий"
```

- Если имя базы данных длиннее 8 символов, например, DB_LONGNAME:

```
CATALOG GLOBAL DATABASE
/.../CELL_VANCOUVER/subsys/database/DB_LONGNAME AS VANVMDB
USING DIRECTORY DCE WITH "комментарий"
```

Оператор CONNECT

Чтобы найти нужный объект каталогов DCE, клиент должен знать полное имя DCE базы данных или экземпляра СУБД. Ниже приводится несколько способов задать это имя в операторе CONNECT.

- Введите алиас, например:

  ```
  CONNECT TO VANVMDB
  ```

- Введите короткое имя, например:

  ```
  CONNECT TO VMDB
  ```

В этих случаях на клиенте и на сервере должен быть задан один и тот же путь. (Путь указывается как значение параметра конфигурации *dir_path_name* или соответствующей переменной реестра).

Команда ATTACH

Реальный путь на клиенте должен совпадать с путем экземпляра СУБД назначения.

Если значение пути *dir_path_name* для клиента и сервера совпадает (например, `/.../CELL_TORONTO/subsys/database/`), а имя *dir_obj_name* на сервере баз данных - `AIX_INST`, соединение устанавливается командой:

```
ATTACH TO AIX_INST
```

Как клиент устанавливает соединение с базой данных

На рис. 8 на стр. 354 показан пример конфигурации сети баз данных с двумя ячейками DCE. Названия этих ячеек - `/.../CELL_TORONTO` и `/.../CELL_VANCOUVER`. (Обе ячейки содержат каталог `/.:/subsys/database/`; это не показано на рисунке, но используется в других примерах).

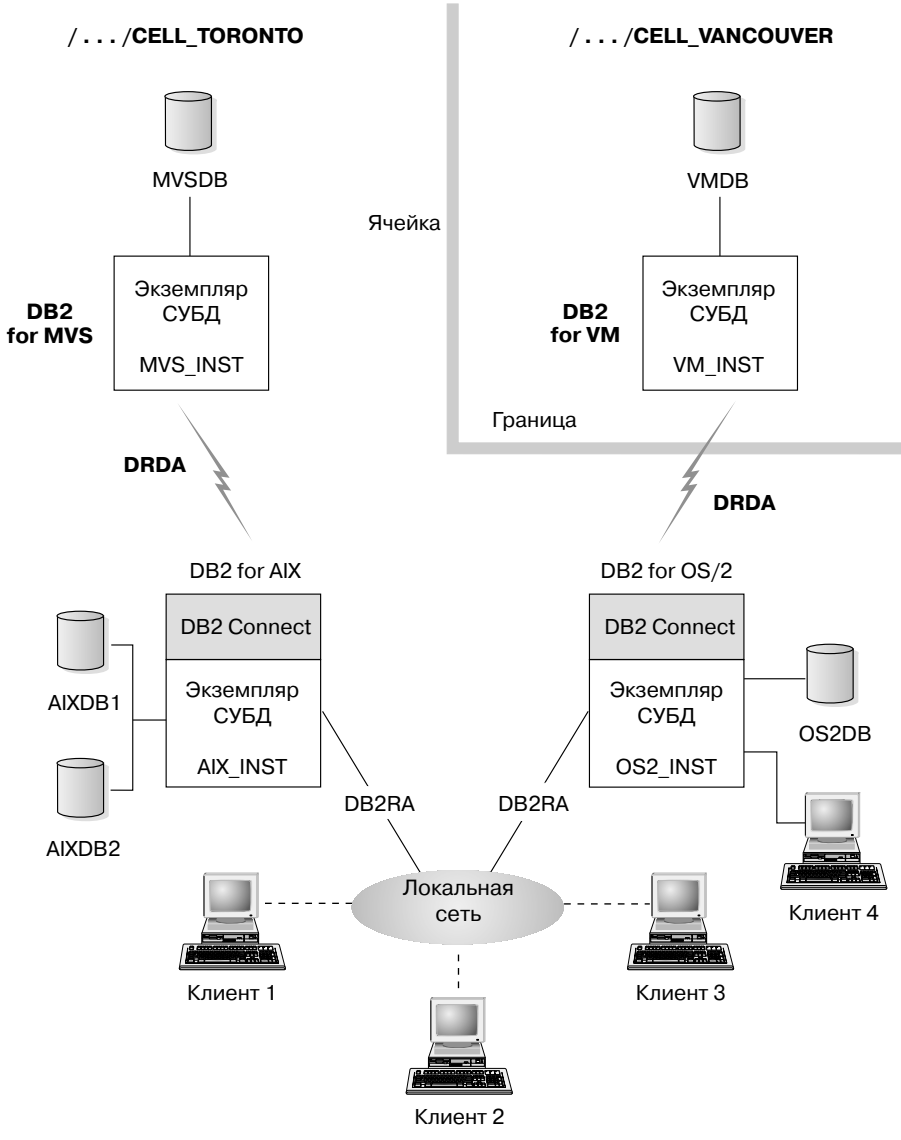


Рисунок 8. Конфигурация сетевой базы данных

Чтобы клиенты в ячейке TORONTO получили доступ ко всем базам данных в обеих ячейках, следует задать значения параметров конфигурации менеджера баз данных и создать следующие объекты:

- Объект базы данных для каждой базы данных.
- Объект локатора баз данных для баз данных назначения на двух серверах баз данных DB2 for AIX и DB2 for OS/2.

- Один объект маршрутной информации, известный всем клиентам. Его атрибуты определяют, какой узел DB2 Connect использовать для баз данных MVSDB и VMDB.

Ниже приводятся примеры соединения клиента с базой данных:

- Соединение с базами данных в той же ячейке
- Соединение с базой данных в другой ячейке.

Примеры включают необходимые параметры конфигурации менеджера баз данных.

Соединение с базами данных в той же ячейке

В этом разделе приводится несколько примеров того, как клиенты устанавливают соединение с базами данных в одной ячейке.

1. Клиент Client_1 соединяется с AIXDB2. У базы данных и клиента одинаковый путь.

   Администратор баз данных должен:

   - Указать этот путь в значении параметра *dir_path_name* (или значении реестра DB2DIRPATHNAME).
   - Указать тип служб каталогов DCE в параметре конфигурации *dir_type*.
   - Указать протокол связи как значение параметра конфигурации *dft_client_comm* (или значение реестра DB2CLIENTCOMM).

   Локальный системный каталог баз данных не содержит AIXDB2, поэтому в каталоге DCE ищется полное имя. Это имя получается приписываем к значению параметра конфигурации *dir_path_name* (или значению реестра DB2DIRPATHNAME) имени AIXDB2.

   Последовательность событий:

   a. Клиент Client_1 находит объект баз данных AIXDB2 с помощью имени DCE базы данных `/.../CELL_TORONTO/subsys/database/AIXDB2`.
   b. Из этого объекта клиент Client_1 получает информацию, что AIXDB2 использует протокол баз данных DB2RA, тот же, что и Client_1.
   c. Протоколы баз данных совпадают, поэтому Client_1 находит в объекте локатора СУБД для AIX_INST значение атрибута протокола связи, совпадающее с собственным протоколом, и начинает диалог с соответствующим экземпляром СУБД.

2. Клиент Client_3 подключается к MVSDB. У базы данных и клиента одинаковый путь, но они используют разные протоколы баз данных.

   Администратор баз данных должен:

   - Указать этот путь в значении параметра *dir_path_name* (или значении реестра DB2DIRPATHNAME).
   - Указать тип служб каталогов DCE в параметре конфигурации *dir_type*.

- Указать протокол связи как значение параметра конфигурации *dft_client_comm* (или значение реестра DB2CLIENTCOMM).
- Указать имя DCE объекта маршрутной информации по умолчанию как значение параметра конфигурации *route_obj_name* (или значение реестра DB2ROUTE).

Последовательность событий:

a. Клиент Client_3 находит объект баз данных MVSDB с помощью имени DCE базы данных `/.../CELL_TORONTO/subsys/database/MVSDB`.
b. Из этого объекта клиент Client_3 получает информацию, что MVSDB использует только протокол баз данных DRDA, отличающийся от протокола, который использует Client_3.
c. Затем Client_3 находит объект маршрутной информации по имени, определенному в параметре конфигурации *route_obj_name*, или по значению реестра DB2ROUTE. Клиент находит информацию о базе данных назначения для MVSDB.
d. Client_3 находит в объекте локатора, связанном с информацией о базе данных назначения MVSDB, протокол связи и посылает требование SQL CONNECT на маршрутизатор.
e. После этого маршрутизатор устанавливает соединение APPC с MVSDB.

Соединение с базой данных в другой ячейке

В этом разделе разбирается пример установления соединения клиента с базой данных в другой ячейке, причем используются разные протоколы баз данных.

1. Клиент Client_3 сконфигурирован так, чтобы использовать:
   - Службы каталогов DCE: параметр *dir_type* имеет значение DCE.
   - Ячейку, не совпадающую с CELL_VANCOUVER: параметр конфигурации *dir_path_name* имеет другое значение, например:

     ```
     /.../CELL_TORONTO/subsys/database/
     ```
2. Чтобы Client_3 мог установить соединение с VMDB, администратор баз данных должен:
   - В явном виде внести VMDB в локальный системный каталог баз данных. Связать с именем DCE для VMDB локально уникальный алиас и использовать его в операторе CONNECT. Например:

     ```
     CATALOG GLOBAL DATABASE
     /.../CELL_VANCOUVER/subsys/database/VMDB AS VANVMDB
     USING DIRECTORY DCE WITH "комментарий"
     ```

     , и далее:

     ```
     CONNECT TO VANVMDB
     ```
   - Указать протокол связи как значение параметра конфигурации *dft_client_comm* (или значение реестра DB2CLIENTCOMM).

- Указать имя DCE объекта маршрутной информации по умолчанию как значение параметра конфигурации *route_obj_name* (или значение реестра DB2ROUTE).

Последовательность событий:

a. Клиент Client_3 находит полное имя DCE VANVMDB в своем системном каталоге баз данных.
b. Клиент Client_3 находит объект баз данных VMDB с помощью ее имени DCE `/.../CELL_VANCOUVER/subsys/database/VMDB`.
c. Из этого объекта клиент Client_3 получает информацию, что VMDB использует только протокол баз данных DRDA, отличающийся от протокола, который использует Client_3.
d. Затем Client_3 находит объект маршрутной информации по имени, определенному в параметре конфигурации *route_obj_name*, или по значению реестра DB2ROUTE. Клиент находит информацию о базе данных назначения для VMDB.
e. Client_3 находит в объекте локатора, связанном с информацией о базе данных назначения VMDB, протокол связи и посылает требование SQL CONNECT на маршрутизатор.
f. После этого маршрутизатор устанавливает соединение APPC с VMDB.

Как происходит просмотр каталогов

Если каталог DCE используется в среде, где у всех баз данных назначения совпадает путь, на клиентах нет необходимости в локальных каталогах.

В этом разделе описан порядок просмотра каталогов для следующих команд:

- Команда ATTACH
- Оператор CONNECT

Команда ATTACH

На рис. 9 на стр. 358 показано, в каком порядке ведется просмотр каталогов, когда клиент подключается к экземпляру СУБД под названием ABC_INST.

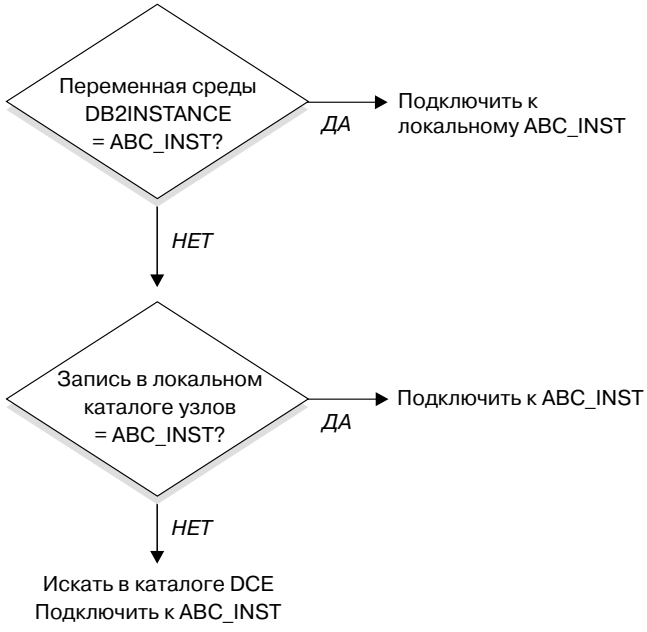


Рисунок 9. Как используются каталоги при подключении к базе данных

Оператор CONNECT

На рис. 10 на стр. 359 показано, в каком порядке идет просмотр каталогов, когда клиент устанавливает соединение с базой данных под названием DBTEST.

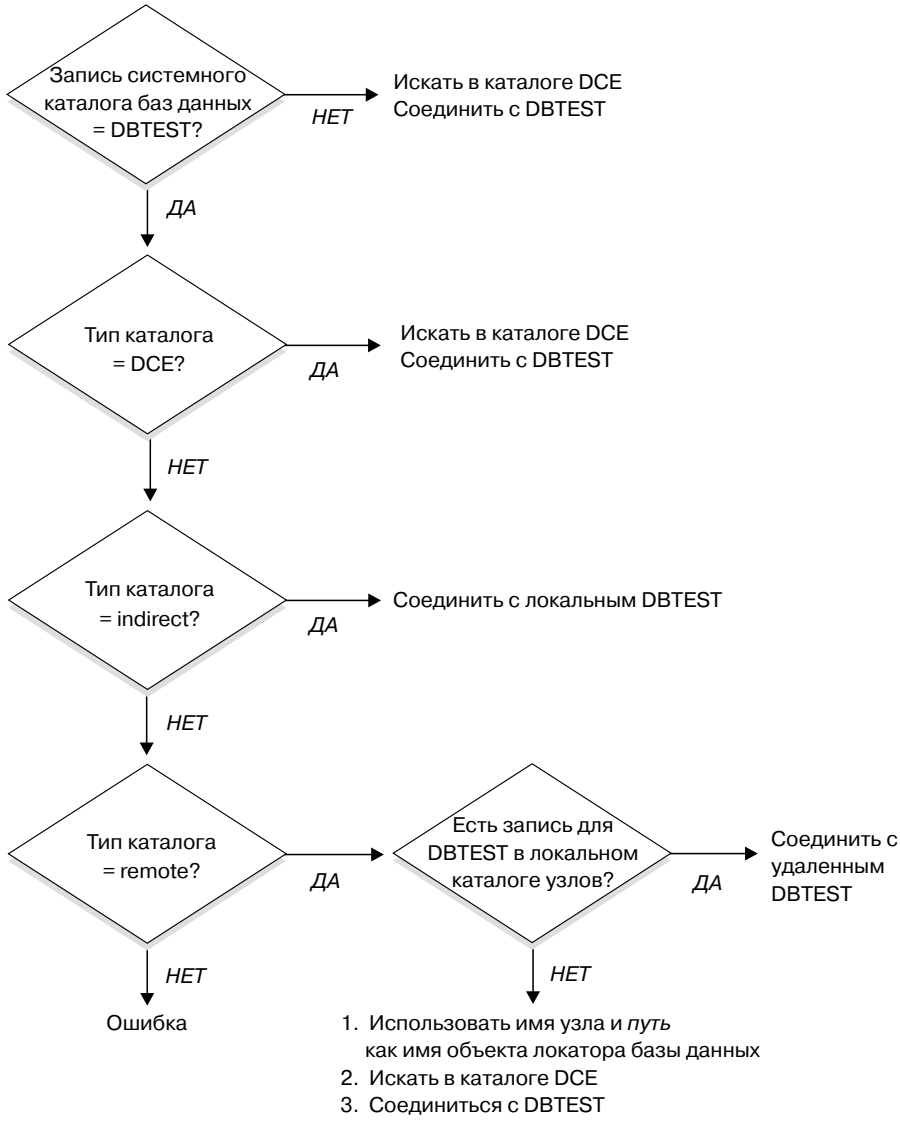


Рисунок 10. Как используются каталоги при соединении с базой данных

Временное переопределение информации о каталогах DCE

Значения в локальном каталоге баз данных имеют больший приоритет, чем информация о каталогах DCE. Например, посылая команду CONNECT TO DBTEST, когда база данных `/.:/subsys/database/DBTEST` определена в каталоге DCE как находящаяся на хосте JAGUAR, можно временно заменить DBTEST на

другую базу данных, находящуюся на хосте STORM. Внесите DBTEST в локальный каталог как удаленную базу данных, а в качестве каталога узла укажите STORM.

Для базы данных, в имени DCE которой путь не совпадает с путем клиента, можно создать алиас. Подробности об этой команде смотрите в разделе “команда CATALOG GLOBAL DATABASE” на стр. 352.

Задачи служб каталогов

Ниже перечислены задачи, которые необходимо выполнить для конфигурирования и использования служб каталогов DCE. Каждый раздел посвящен одной из задач.

- Задачи администратора DCE

  Администратор DCE должен изменить каталог DCE, чтобы можно было включить в него информацию о новых ресурсах баз данных.

- Задачи администратора баз данных

  Администратор баз данных должен изменить каталог DCE и внести информацию, необходимую для установки и конфигурирования DB2.

- Задачи пользователя базы данных

  Пользователь базы данных должен зарегистрироваться в DCE, зная имя базы данных назначения.

Кроме того, администратор сети настраивает доступ к сети для каждого пользовательского узла. Подробности смотрите в документации по сети.

Задачи администратора DCE

Чтобы обеспечить создание и чтение объектов каталогов, администратор DCE должен выполнить следующие задачи:

- Выделить для DB2 поддерево каталогов, например, /.:/subsys/database
- Дать администратору баз данных привилегию создавать объекты каталогов
- Дать пользователям баз данных привилегию читать объекты каталогов
- Добавить информацию об атрибутах нового объекта каталога DCE в *таблицу атрибутов DCE*.

  В файле атрибутов CDS (на платформах UNIX */etc/dce/cds_attributes*; в OS/2 *X:\opt\dcelocal\etc\cds_attr*, где "X" - буква диска) добавить:

```
 1.3.18.0.2.4.30 DB_Comment                char
1.3.18.0.2.4.31 DB_Communication_Protocol char
1.3.18.0.2.4.32 DB_Database_Protocol      char
1.3.18.0.2.4.33 DB_Database_Locator_Name  char
1.3.18.0.2.4.34 DB_Native_Database_Name   char
1.3.18.0.2.4.35 DB_Object_Type            char
1.3.18.0.2.4.36 DB_Product_Name           char
1.3.18.0.2.4.37 DB_Product_Release        char
```

```
1.3.18.0.2.4.38 DB_Target_Database_Info   char
1.3.18.0.2.4.39 DB_Authentication         char
1.3.18.0.2.4.63 DB_Principal              char
```

- Обеспечить работу DCE, когда пользователям нужен будет доступ к базам данных с помощью служб каталогов DCE.

Подробности смотрите в документации по DCE для вашей платформы.

Задачи администратора баз данных

Администратор баз данных должен выполнить следующие задачи:

- Получить от администратора DCE поддерево каталогов для ресурсов баз данных. Например, `/.:/subsys/database`
- Во время установки менеджера баз данных DB2 попросить администратора DCE добавить атрибуты нового объекта каталогов DCE, требуемые DB2.
- Каждому экземпляру СУБД задать уникальное имя в поддереве каталогов DCE. Например, `/.:/subsys/database/AIX_INST`
- Для каждого экземпляра СУБД задать параметры конфигурации менеджера баз данных DCE.
  - *dir_type*
  - *dir_obj_name*
  - *dir_path_name*
  - *route_obj_name*
  - *dft_client_comm*
  - *dft_client_adpt*

  Некоторые параметры конфигурации могут временно переопределяться переменными реестра клиента. Дополнительную информацию смотрите в разделе “Параметры конфигурации и переменные реестра” на стр. 351.
- Приписать каждой базе данных в поддереве каталогов DCE уникальное имя. Указать это имя в параметре *dir_obj_name* в файле конфигурации базы данных.
- Создать объекты для служб каталогов DCE с помощью команд DCE *cdscp* создания и просмотра объектов. Объекты создаются независимо от установки менеджера баз данных и запуска экземпляра менеджера баз данных.

  Существуют три типа объектов.
  - Для каждой базы данных назначения требуется объект баз данных.
  - Для каждого экземпляра DB2 Connect и для каждого экземпляра СУБД (без DB2 Connect), связанного с несколькими базами данных, требуется объект локатора базы данных.
  - Для доступа к хосту требуются объекты маршрутной информации.
- В зависимости от среды администратор баз данных должен решить:

– Как распределить клиенты по логическим группам в зависимости от баз данных, к которым они обращаются, и от используемых протоколов связи.
– Сколько требуется объектов маршрутной информации.
– Какие базы данных назначения записывать в каждом из объектов маршрутной информации.
– Какие объекты маршрутной информации сделать известными для каких групп клиентов.

Подробности об этих объектах смотрите в разделе “Создание объектов каталогов” на стр. 339.

Задачи пользователя базы данных

Пользователь базы данных должен выполнить следующие задачи:

- Узнать от администратора баз данных имя базы данных. Это может быть короткое имя или полное имя DCE.
- Если потребуется, задать переменные реестра, необходимые для служб каталогов DCE Directory Services. Переменные реестра, заданные на клиенте, могут временно переопределять параметры конфигурации.
  – При обращении к базе данных хоста - узнать от администратора базы данных полное имя DCE объекта маршрутной информации. Если оно не указано в параметре *route_obj_name* или не совпадает с его значением, задать это имя в переменной реестра DB2ROUTE, и только после этого пытаться установить соединение.
  – Если в параметре *dft_client_comm* не указан предпочтительный для вас протокол связи или указан другой, задать протокол связи клиента в переменной реестра DB2CLIENTCOMM. Ниже приводится **несколько** примеров для UNIX:

```
db2set DB2CLIENTCOMM=tcpip
db2set DB2CLIENTCOMM=appc
db2set DB2CLIENTCOMM=tcpip,appc
db2set DB2CLIENTCOMM=appc,tcpip
```

Несколько примеров для OS/2:

```
 db2set DB2CLIENTCOMM=ipxspx
db2set DB2CLIENTCOMM=netbios
db2set DB2CLIENTCOMM=tcpip,ipxspx,netbios
db2set DB2CLIENTCOMM=netbios,tcpip,ipxspx,appc
```

Несколько примеров для операционной системы Windows:

```
db2set DB2CLIENTCOMM=npipe
db2set DB2CLIENTCOMM=netbios
db2set DB2CLIENTCOMM=tcpip,ipxspx,netbios
db2set DB2CLIENTCOMM=netbios,tcpip,ipxspx,appc,npipe
```

Если протоколов связи несколько, используется указанный первым.

- Если путь DCE какой-нибудь из баз данных не совпадает с путем, заданным в параметре конфигурации *dir_path_name* или переменной реестра DB2DIRPATHNAME, его следует внести в каталог с помощью команды CATALOG GLOBAL DATABASE. Дополнительную информацию смотрите в разделе “команда CATALOG GLOBAL DATABASE” на стр. 352.
- Устанавливая соединение с базой данных назначения или экземпляром базы данных, следует предварительно зарегистрироваться в DCE. Дополнительную информацию о команде login смотрите в руководстве Refer to the *OSF DCE Administration Guide*.

Ограничения служб каталогов

В этом разделе перечислены неподдерживаемые возможности.

- Поддерживаются не все клиенты баз данных. Чтобы узнать, поддерживаются ли вашим клиентом DB2 службы каталогов DCE, посмотрите руководство *Быстрый старт*. В настоящее время поддерживаются только клиенты DB2 для всех операционных систем UNIX, OS/2 и поддерживаемых систем Windows.
- Клиент не может использовать службы каталогов DCE для связи с сервером DB2 for OS/2 версии 1.
- Клиенты поддерживаемых операционных систем Windows могут использовать следующие протоколы: TCP/IP, APPC, NetBIOS, IPX/SPX и NPIPE. Клиенты OS/2 могут использовать следующие протоколы: TCP/IP, APPC, NetBIOS и IPX/SPX. Все поддерживаемые клиенты UNIX могут использовать только протоколы TCP/IP и APPC.
- Команды LIST DATABASE (и NODE) DIRECTORY возвращают только записи из локальных каталогов, но не из каталога DCE. Просмотреть объекты в DCE можно с помощью команды *cdscp show object*.
- При соблюдении всех перечисленных ниже условий владелец экземпляра менеджера баз данных должен зарегистрироваться в DCE, прежде чем запустить менеджер баз данных (обычно с помощью команды `db2start`).
  - Экземпляр менеджера баз данных сконфигурирован для поддержки служб каталогов DCE с помощью параметра конфигурации *dir_type*
  - Объект служб каталогов ячейки доступен для чтения только после регистрации в DCE
  - Доступ к каталогу DCE необходим для поддержки:
    - Базы данных менеджера транзакций (заданной в параметре конфигурации *tm_database*), которая расположена на другом экземпляре
    - Клиента, не поддерживающего службы каталогов DCE или не сконфигурированного для их использования.

Примечание: При регистрации в DCE следует использовать принципал с большим сроком действия квитанции.

- При соединении клиента, использующего службы каталогов DCE, с сервером DRDA через шлюз DDCS версии 2.2 (или более ранней) необходимо внести алиас базы данных в локальный каталог шлюза. Этот алиас должен совпадать с алиасом на клиенте и обозначать ту же базу данных.
- Для клиентов поддерживаемых операционных систем Windows будет использоваться DB2DCE.DLL. Этот файл находится в подкаталоге *bin* подкаталога *sqllib*. Если DCE предоставляет Gradient**, по умолчанию файл DB2DCE.GRD эквивалентен файлу DB2DCE.DLL. Если DCE предоставляет IBM, файл DB2DCE.IBM следует скопировать в DB2DCE.DLL.

Приложение C. Обработчик пользователя для восстановления баз данных

Содержит обзорную информацию, аналогичную приводимой в главе Перемещение данных.

Примечание: Вся информация по данным темам и связанные с ней вопросы из книг *Command Reference* и *Administrative API Reference* собраны в книге *Data Recovery and High Availability Guide and Reference*.

Книга *Data Recovery and High Availability Guide and Reference* - это основной источник информации по данным темам.

Приложение D. Выполнение команд на нескольких разделах базы данных

В системе многораздельных баз данных может быть необходимо выполнить какие-либо команды на компьютерах экземпляра или на серверах разделов базы данных (узлах). Это можно сделать с помощью команды **rah** или команды **db2_all**. Команда **rah** позволяет посылать команды компьютерам экземпляра. Чтобы выполнить команды на серверах разделов базы данных, пользуйтесь командой **db2_all**. В этом разделе приводится обзор таких команд. Все нижеизложенное относится только к системам многораздельных баз данных.

Примечания:

1. На платформах на основе UNIX можно использовать оболочку Korn или любую другую; однако разные оболочки по-разному обрабатывают команды, содержащие спецсимволы.
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**.

Команды

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"
```

Описания команд

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 ваши 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.

Задание выполняемой команды

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" &
```

Параллельное выполнение команд на платформах на основе 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.

Мониторинг процессов rah на платформах на основе UNIX

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. Номер сигнала задавать не следует. 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 оболочкой 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.

Дополнительная информация о команде rah (Run All Hosts - запустить на всех хостах) (только 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.)

Префиксные последовательности

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. Последнее может произойти, если, например, продолжает выполняться дочерний (на платформах на основе UNIX) или фоновый (в Windows NT) процесс. В этом случае команда запускает отдельный фоновый процесс для получения любых удаленных выходных данных, сгенерированных после завершения команды и записывает его 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**. Подробности смотрите в руководстве *DB2 Enterprise - Extended Edition for Windows Quick Beginnings*.
2. Если задан символ ", повторы не удаляются из списка компьютеров. Если нужно удалить повторы, смотрите раздел “Удаление повторений из списка компьютеров” на стр. 376.

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.

Задание списка компьютеров

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**. Подробности смотрите в руководстве *DB2 Enterprise - Extended Edition for Windows Quick Beginnings*.

Удаление повторений из списка компьютеров

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. Информацию о префиксных последовательностях смотрите в разделе “Префиксные последовательности” на стр. 372.

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.

Управление командой rah

Для управления командой **rah** можно использовать перечисленные ниже переменные среды.

Таблица 23.

Имя	Значение	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 23. (продолжение)

Имя	Значение	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 значение, 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).

$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
```

Задание профиля среды по умолчанию в Windows NT

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**.

Диагностика ошибок при работе с 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>. Номер сигнала задавать не следует. 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.

Приложение E. Как DB2 for Windows NT работает с защитой Windows NT

При установке Windows NT разрешает создать два имени пользователя администратора:

- Один пользователь носит имя “Administrator”
- Другой носит выбранное вами имя. Он должен иметь полномочия администратора; имя должно соответствовать правилам именования DB2. Дополнительную информацию о правилах именования DB2 смотрите в разделе “Приложение A. Правила именования” на стр. 331.

Этот пользователь может зарегистрироваться на локальном компьютере или, если компьютер установлен в домене Windows NT - на домене. DB2 for Windows NT поддерживает оба варианта. Для аутентификации пользователя DB2 вначале проверяет локальный компьютер, затем контроллер доменов для текущего домена и, наконец, все доверенные домены, известные контроллеру доменов.

Покажем, как это происходит, в предположении, что экземпляр DB2 требует аутентификации типа Server. Используется следующая конфигурация:

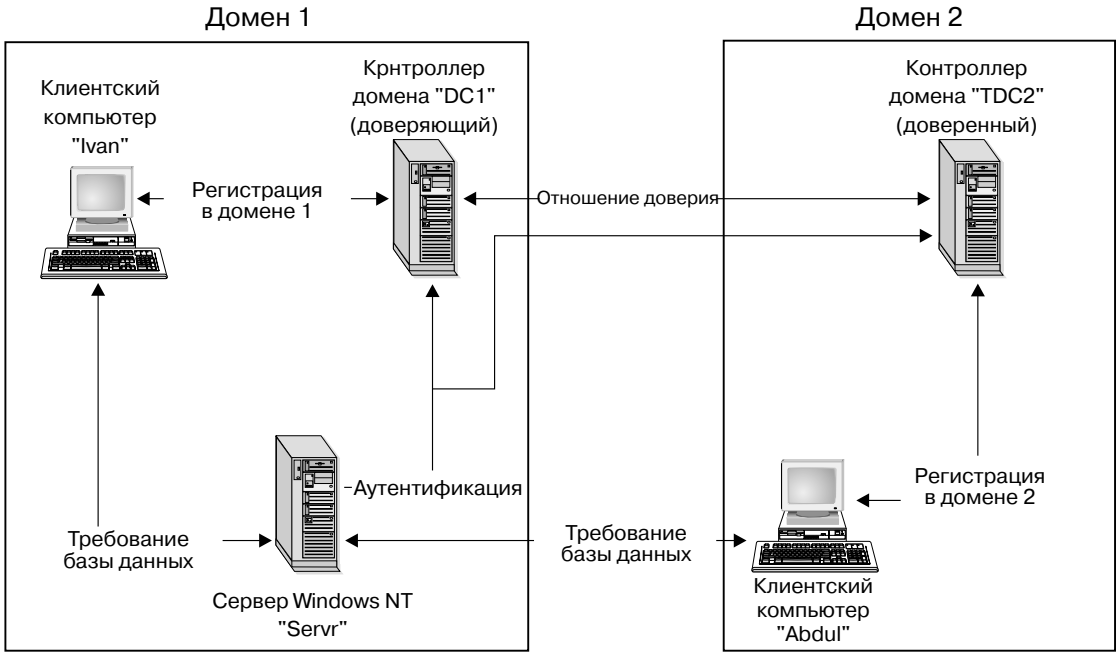


Рисунок 11. Аутентификация при помощи доменов Windows NT

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.

Пример сценария с аутентификацией типа Server:

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.

Пример сценария с аутентификацией типа Client и клиентским компьютером 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 myinst
db2start myinst
```

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 тем, как пытаться установить связь с базой данных DB2, убедитесь, что запущена служба защиты DB2. Служба защиты устанавливается DB2 и настраивается для запуска в качестве службы Windows NT, но она не запускается автоматически. Чтобы запустить службу защиты DB2, введите команду `NET START DB2NTSECSERVER`.

Пример сценария с аутентификацией типа Client и клиентским компьютером Windows 95:

1. Администратор Dale регистрируется на SRV3 и изменяет тип аутентификации для экземпляра базы данных на Client:

```
db2 update dbm cfg using authentication client
db2stop myinst
db2start myinst
```

2. Пользователь Ivan на компьютере клиента Windows 95 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 с Windows 95 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 95 не может проверить переданное имя пользователя и пароль, аутентификация типа client в Windows 95 в принципе не обеспечивает защиты. Но если компьютер с Windows 95 имеет доступ к провайдеру защиты Windows NT, некоторые меры защиты можно обеспечить, настроив систему Windows 95 для регистрации через проверенный промежуточный сервер. Подробности о том, как настроить для этого вашу систему Windows 95, смотрите в документации Microsoft по Windows 95.

DB2 также поддерживает глобальные группы. Чтобы использовать глобальные группы, нужно включить глобальные группы в какую-нибудь локальную группу на сервере защиты. 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).

Использование резервного контроллера домена с DB2

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 для DB2 задается 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 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.

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 имя пользователя и имя группы

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.)

Служба защиты DB2 for Windows NT

В DB2 Universal Database аутентификация имен пользователей и паролей интегрирована в системный контроллер DB2. Служба защиты требуется только когда клиента соединен с сервером, настроенном на аутентификацию типа CLIENT.

Установка DB2 на резервном контроллере домена

В среде Windows NT пользователь может пройти аутентификацию либо на первичном контроллере, либо на резервном контроллере. Эта возможность очень важна в больших распределенных локальных сетях с одним центральным первичным 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 аутентифкации первичный контроллер домена (PDC, primary domain controller).

В этом случае преимущество резервного контроллера домена в том, что что пользователи проходят аутентификацию быстрее, и локальная сеть не перегружается, как было бы без BDC.

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 первичном 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 локальных сетях служба Computer Browser могут не быть запущена, и информация менеджера сервера может устареть. 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 сконфигурирована как резервный контроллер домена для заданного домена. (Если компьютер настроен как резервный контроллер другого домена, это значение приведет к ошибке.)

Аутентификация при помощи групп и защиты домена

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 DB2_GRP_LOOKUP=local
    ```

  Чтобы просмотреть все заданные переменные реестра профилей DB2, введите

  ```
  db2set -all
  ```

Чтобы DB2 for Windows NT работала с защитой домена, нужно предоставить полномочия и привилегии локальной группе. Чтобы аутентификация происходила правильно, имена пользователей в локальных и глобальных группах НЕОБХОДИМО определять в том же домене, что и сами группы, .

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** станет искать пользователя на другой компьютере домена или на контроллерах доменов.

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.

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 24. 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`	

Таблица 24. Успешное соединение с использованием контроллера домена (продолжение)

Domain1	**Domain2**
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.	

Пример неудачной попытки соединения: В следующем сценарии соединения не происходит, поскольку имя пользователя определено не на том домене, на котором определена локальная или глобальная группа.

Таблица 25. Неудачная попытка соединения при помощи контроллера домена

Domain1	**Domain2**
Существует доверенное соединение с доменом Domain2.	• Существует доверенное соединение с доменом Domain1. • Определена локальная или глобальная группа `grp2`.
• Определена глобальная группа `grp1`. • Определено имя пользователя `id1`. • Имя пользователя `id1` входит в группу `grp1`.	
	Domain1\`grp1` 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`	

При поиске в локальном или глобальном домене обнаружен `id1`. DB2 получает информацию об этом имени пользователя (то есть о том, что имя пользователя `id1` входит в группу `grp1`, а группа `grp1` входит в Domain2\`grp2`).	
	Группа `grp2` существует в этом домене.

Таблица 25. Неудачная попытка соединения при помощи контроллера домена (продолжение)

Domain1	**Domain2**
Соединения не происходит, поскольку локальная или глобальная группа находится в Domain2, а актуальное имя пользователя определено в Domain1. Соединение было бы установлено, если бы вместо приведенной команды выполнялась такая: `GRANT CONNECT ON db TO GROUP grp1`	

Приложение F. Использование монитора производительности Windows NT

Существует два монитора производительности, доступных пользователям DB2 for Windows NT:

- **Монитор производительности DB2**

  Монитор производительности DB2 поддерживает снимки и данные событий, относящиеся только к 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 NT**

  Монитор производительности Windows позволяет следить за производительностью как базы данных, так и системы, получая информацию от всех поставщиков данных производительности, зарегистрированных в системе. Windows NT также предоставляет данные производительности по всем аспектам работы компьютера, включая:

  – Занятость процессора
  – Использование памяти
  – Активность дисков
  – Активность сети

Регистрация DB2 на мониторе производительности Windows NT

Программа установки автоматически регистрирует DB2 на мониторе производительности Windows NT.

Чтобы сделать информацию производительности DB2 и DB2 Connect доступной для монитора производительности Windows NT, нужно зарегистрировать DLL для счетчиков производительности DB2 for Windows NT. Одновременно данные производительности станут доступны для любых других прикладных программ Windows NT, использующих API производительности Win32.

Чтобы установить и зарегистрировать на мониторе производительности Windows NT модуль DLL счетчиков производительности DB2 for Windows NT (DB2Perf.DLL), 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.

Разрешение удаленного доступа к информации производительности DB2

Если ваша рабочая станция DB2 for Windows NT соединена сетью с другими компьютерами Windows NT, можно пользоваться описанными в этом разделе возможностями.

Чтобы посмотреть объекты производительности Windows NT из другого компьютера с DB2 for Windows NT, нужно зарегистрировать на DB2 имя пользователя и пароль администратора. (Имя пользователя монитора производительности Windows NT по умолчанию - **SYSTEM** - зарезервированное слово DB2, и использовать его нельзя.) 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. Сочетание имени пользователя и пароля должно поддерживаться совпадающими со значениями имени пользователя и пароля, хранящимися в базе данных защиты Windows NT. При изменении имени пользователя и

пароля в базе данных защиты Windows NT нужно выполнить сброс сочетания имени пользователя и пароля, используемых для удаленного мониторинга производительности.

3. Для отмены регистрации введите:

```
db2perfr -u <имя_пользователя> <пароль>
```

Вывод значений производительности DB2 и DB2 Connect

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:”.

Windows NT поддерживает следующие объекты производительности:

- **Менеджер баз данных DB2**

  Этот объект дает общую информацию об отдельном экземпляре Windows NT. 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 NT зависит от программного обеспечения, установленного на вашем компьютере Windows NT, и от того, какие программы активны сейчас. 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 вы используете систему Windows NT в качестве шлюза DB2 Connect и в настоящий момент активны какие-то базы данных и программы DCS, в списке будут объекты баз данных DCS DB2 и программ DCS DB2.

Доступ к удаленной информации производительности DB2

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

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. Можно также можете список сбрасываемых баз данных. 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 сейчас обращаются к информации производительности баз данных соответствующего экземпляра сервера DB2 (то есть указанного значение `db2instance` сеанса, в котором вы запустили **db2perfc**).

Вызов **db2perfc** также сбросит значения, которые увидит всякий удаленный получатель информации производительности DB2, выполнив команду **db2perfc**.

Примечание: Существует также API DB2 `sqlmrset`, который позволяет прикладной программе сбросить значения, которые она видит, не глобально, а локально, для конкретных баз данных. Дополнительную информацию смотрите в разделе *Administrative API Reference*.

Приложение G. Работа с серверами разделов баз данных Windows NT или Windows 2000

Задачи по изменению характеристик конфигурации в среде Windows NT или Windows 2000 решаются при помощи специальных утилит. В среде других операционных систем применяются методы, описанные в главе "Масштабирование конфигурации при добавлении процессоров" руководства *Administration Guide: Performance*.

Ниже описаны следующие утилиты:

- "Вывод списка серверов разделов баз данных в экземпляре"
- "Добавление сервера разделов баз данных к экземпляру"
- "Изменение раздела базы данных" на стр. 401
- "Отбрасывание раздела базы данных из экземпляра" на стр. 402

Вывод списка серверов разделов баз данных в экземпляре

В Windows NT или Windows 2000 команда **db2nlist** служит для получения списка серверов разделов баз данных, входящих в экземпляр.

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.

Добавление сервера разделов баз данных к экземпляру

В Windows NT или Windows 2000 для добавления сервера разделов баз данных (узла) к экземпляру служит команда **db2ncrt**.

Примечание: Не используйте команду **db2ncrt**, если этот экземпляр уже содержит базы данных. Вместо этого используйте 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 диапазон, отведенный для соединений FCM в файле служб в каталоге `x:\winnt\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 NT, на которой находится узел; по умолчанию - 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
```

Изменение раздела базы данных

В системах Windows NT или Windows 2000 команда **db2nchg** позволяет:

- 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:имя_сети

  Изменяет имя сети 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. Вы можете ввести `имя_сети`, используя имя сети 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
```

Отбрасывание раздела базы данных из экземпляра

В Windows NT или Windows 2000 для отбрасывания сервера разделов баз данных (узла) из экземпляра 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.

Если вы хотите отбросить узел, назначенный логическому порту 0, из компьютера, на которой запущено несколько логических узлов, нужно отбросить все остальные узлы, назначенные другим логическим портам, прежде чем вы сможете отбросить узел, назначенный логическому порту 0. У каждого сервера разделов баз данных должен быть узел, назначенный логическому порту 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).

Приложение H. Конфигурирование нескольких логических узлов

Обычно DB2 Enterprise - Extended Edition конфигурируется так, что каждому компьютеру назначается один сервер разделов баз 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.

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.

Приложение I. Высокоскоростная межузловая связь

DB2 Universal Database Enterprise - Extended Edition позволяет работать в коммуникационно насыщенной среде, где общая производительность системы жизненно важна для вашего бизнеса.

Есть два типа сетей, которые можно использовать для многораздельной среды. Первый использует TCP/IP в общедоступных сетях. Другой тип использует TCP/IP или архитектуру VI (Virtual Interface - виртуальный интерфейс) на выделенном соединении.

Соединение общего доступа использует установленный протокол TCP/IP. TCP/IP как протокол связи доступен практически везде. При этом используется среда локальных сетей. Преимущество этой среды - возможность немедленного присоединения вашего кластера без применения дополнительных особых программных и аппаратных средств. Недостаток этой среды в том, что трафик дополнительного кластера ухудшает качество обслуживания во всей сети. Например, эффект коммуникационного “взрыва” при активности базы данных внутри кластера способен нарушить связь во всей сети. Кроме того, информационный обмен в остальной части сети затрудняет поддержание производительности работы с базами данных внутри кластера.

Выделенное соединение работает как отдельная сеть. Эта сеть может быть единственной доступной внутри кластера или же использоваться в дополнение к сетевой среде. Эта сеть выделяется для обеспечения связи между членами данного кластера. Такую сеть называют системной сетью (System Area Network, SAN). На производительность базы данных не влияет трафик внешних линий связи (в отличие от среды локальной сети), и наоборот. Недостаток этой среды в том, что могут потребоваться раздельное управление обеими сетями и дополнительные затраты на оборудование, программное обеспечение и протоколы для локальной сети и системной сети. Пример выделенного соединения - 100 Мб/с Ethernet.

Вы можете захотеть сохранить обеспечить поддержку уже имеющейся общедоступной локальной сети, но при этом получить возможность передавать большие объемы через системную сеть (внутри вашего кластера). Такая распределение удобно, если вы хотите иметь коммуникационный доступ за пределы кластера. В операционной среде Windows NT сохранение общедоступной локальной сети для доступа к контроллеру доменов NT. (Информацию о контроллере доменов смотрите в разделе “Приложение E. Как DB2 for Windows NT работает с защитой Windows NT” на стр. 383.)

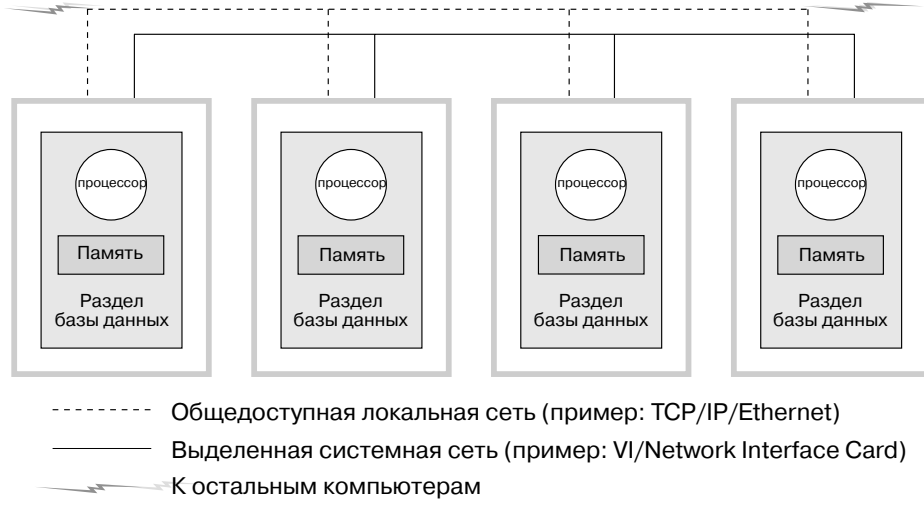


Рисунок 12. Объединение выделенной системной сети с общедоступной локальной сетью

В остальных разделах обсуждаются:

- “Высокоскоростное соединение с использованием TCP/IP”
- “Высокоскоростное соединение с использованием VI” на стр. 409

Высокоскоростное соединение с использованием TCP/IP

Примеры необходимых требований к установке сетевого оборудования с использованием TCP/IP:

- Стандартная сеть Ethernet.

  Особых требований к аппаратному и программному обеспечению, а также протоколам связи нет.
- IBM Netfinity SP Switch.

  Требования изложены в следующем разделе.

Необходимые требования для использования IBM Netfinity SP Switch

Информацию об Netfinity смотрите на странице:
http://www.ibm.com/pc/us/netfinity

Дополнительную документацию и обновления программ смотрите на сайте поддержки IBM: http://www.ibm.com/pc/support

1. Выберите **Servers**
2. В разделе family выберите **Clustering**
3. В разделе Technical Information выберите **Downloadable files**, если требуется обновление программ, или **Online publications**, если нужна документация

Найдите раздел IBM Netfinity SP Switch и сгрузите нужные файлы.

Процедура установки IBM Netfinity SP Switch

Указания по установке IBM Netfinity SP Switch смотрите в руководстве *IBM Netfinity SP Switch Installation and User's Guide*.

Для установки, конфигурирования и тестирования этих компонентов (массивов серверов, адаптера хоста или коммутатора SP) используйте руководства по установке устройств и программ, поставляемые с различными программными и аппаратными средствами .

Установленная система DB2 без дополнительных модификаций будет использовать IBM Netfinity SP Switch.

Высокоскоростное соединение с использованием VI

Архитектура виртуального интерфейса (VI) - это альтернативный TCP/IP протокол межузловой связи в среде массово-параллельной обработки (MPP) Windows NT. VI - новая архитектура связи, разработанная совместно Intel, Microsoft и Compaq для повышения производительности системных сетей. Дополнительную информацию об этой архитектуре смотрите на сайте http://www.viarch.org

Есть продукты, приобретаемые отдельно от DB2 Universal Database, в состав которых входят сетевая плата с поддержкой VIA, коммутатор и программный драйвер. Ряд независимых поставщиков аппаратуры выпускает или планирует выпуск таких продуктов.

Архитектура VI по сравнению с TCP/IP обладает низкой латентностью, высокой пропускной способностью и низкими требованиями к процессорам. В коммуникационно-насыщенной среде использование архитектуры VI повышает общую производительность системы. Чем больше количество узлов в кластере и объем передаваемой информации, тем больше выигрыш от применения архитектуры VI.

DB2 Universal Database поддерживает реализации архитектуры VI, удовлетворяющие спецификациям *Virtual Interface Architecture Specification, Version 1.0*, *Intel Virtual Interface (VI) Architecture Developers' Guide, Version 1.0* и прошедшие тест "Virtual Interface Architecture Conformance Suite". Эти спецификации можно найти по адресу http://www.intel.com/design/servers/vi/the_spec/specification.htm. Руководство разработчика смотрите на сайте http://www.intel.com/design/servers/vi/developer/ia_imp_guide.htm. Там же находится информация о тестах соответствия.

Поддержка архитектуры VI была объявлена IBM в DB2 Universal Database EEE V5.2.

Чтобы узнать о других продуктах архитектуры VI, поддерживаемых DB2 Universal Database EEE, обратитесь в центр поддержки DB2 Universal Database по адресу http://www.software.ibm.com/data или звоните 1-800-237-5511 (только из США и Канады).

Следующие продукты были протестированы с DB2 Universal Database:

- GigaNet Interconnect, подробности смотрите в разделе “Процедура установки для GigaNet Interconnect” на стр. 411.
- Compaq ServerNet Interconnect, подробности смотрите в разделе “Процедура установки ServerNet Interconnect” на стр. 413.
- Fujitsu Synfinity Interconnect, подробности смотрите в разделе “Процедура установки Synfinity Interconnect” на стр. 416.

Возможно, с DB2 Universal Database будут работать и другие продукты. Проверить, совместим ли с DB2 Universal Database другой продукт, можно, связавшись с его поставщиком и со службой поддержки IBM.

Установка оборудования виртуального интерфейса (VI)

Примеры необходимых требований к установке сетевого оборудования с использованием VI:

- GigaNet Interconnect.

  Раздел “Процедура установки для GigaNet Interconnect” на стр. 411 содержит обзор информации по аппаратному и программному обеспечению и протоколам связи, необходимым для этой конфигурации.

  Узнать о продуктах GigaNet или связаться со службой поддержки GigaNet можно через сайт: http://www.giganet.com/

- Compaq ServerNet Interconnect.

  Раздел “Процедура установки ServerNet Interconnect” на стр. 413 содержит обзор информации по аппаратному и программному обеспечению и протоколам связи, необходимым для этой конфигурации.

  Узнать о продуктах ServerNet или связаться со службой поддержки ServerNet можно через сайт: http://www.servernet.com/

- Fujitsu Synfinity Interconnect.

  Раздел “Процедура установки Synfinity Interconnect” на стр. 416 содержит обзор информации по аппаратному и программному обеспечению и протоколам связи, необходимым для этой конфигурации.

  Узнать о продуктах Synfinity или связаться со службой поддержки Synfinity в Fujitsu System Technologies можно через сайт: http://www.fjst.com/

Надо сконфигурировать DB2 для использования VI. Раздел “Подготовка DB2 к работе с использованием VI” на стр. 418 содержит необходимую информацию по использованию VI.

Процедура установки для GigaNet Interconnect

Ниже приводится список оборудования и программ, необходимых для установки этой среды:

- Сетевая плата GigaNet GNN1000 Network Interface Card
- GigaNet GNX5000 Switch
- Медные соединительные кабели GigaNet GNCxx11
- Программное обеспечение GigaNet cLAN, Версия 2.0.

Ниже описаны действия, которые надо выполнить, что GigaNet Interconnect мог работать с DB2 Universal Database. Каждый шаг описан кратко, подробности здесь не приводятся. Подробные инструкция и указания смотрите в справочной документации по каждому этапу.

К GigaNet GNN1000 прилагается компакт-диск с программным обеспечением GigaNet cLAN. Этот диск содержит все необходимые программы для установки GigaNet Interconnect. Кроме того, этот компакт-диск содержит комплект разработки программ VI Architecture SDK и Adobe Acrobat Reader. VI Architecture SDK применяется для разработки и тестирования программ, использующих VI. Adobe Acrobat Reader используется для просмотра документации на этом компакт-диске, в которой объясняется, как разрабатывать программы, использующие VI.

Сводка этапов:

1. Установите сетевые платы
2. Установите коммутаторы и кабели
3. Установка драйверов адаптера
4. Установка консоли управления cLAN
5. Проверьте соединение

Описание этапов:

1. Установите сетевые платы GigaNet GNN1000. Инструкции по установке смотрите в руководстве *GigaNet GNN1000 User Guide*.
2. Установите GigaNet GNX5000 Switch и кабели. Инструкции по установке смотрите в руководстве *GigaNet GNX5000 User Guide*.
3. Установите драйверы адаптера GigaNet GNN1000 на каждый узел, соединенный с GNX5000 Switch. Инструкции по установке смотрите в руководстве *GigaNet GNN1000 User Guide*. Если вы устанавливаете драйверы, поставляемые GigaNet:

a. Удалите предыдущую версию драйвера GNN1000. После удаления драйвера перезагрузите узел.

b. Для установки нового драйвера выберите `Пуск->Настройка->Панель управления->Сеть->Адаптеры->Добавить`.

c. Нажмите кнопку **Установить с диска...** и укажите каталог Driver на компакт-диске. Например, если дисковод компакт-дисков - F:, укажите `F:\Driver`

d. Выберите “GNN1000 NDIS Adapter” и нажмите кнопку **OK**.

e. Последний этап установки - настройка сетевых протоколов.

Драйверы адаптера GigaNet можно также сгрузить с сайта GigaNet: http://www.giganet.com.Следуйте инструкциям на странице поддержки сайта GigaNet.

При установке драйвера адаптера GNN1000 производится перезагрузка узла.

4. Для проверки GigaNet Interconnect можно использовать консоль управления GigaNet cLAN (GMC). Консоль управления GigaNet cLAN содержит две части: консоль и агент. Агент должен быть установлен на всех узлах кластера. Консоль может быть установлена на любом сетевом узле, имеющем доступ к узлам кластера. При наиболее гибкой и рекомендуемой установке и консоль, и агент устанавливаются на каждом узле кластера.

   Установите консоль управления GigaNet cLAN. Инструкции по установке и дополнительную информацию о консоли управления cLAN смотрите в руководстве *GigaNet GNN1000 User Guide*. Некоторые подробности процедуры установки:

   a. Вставьте компакт-диск с программным обеспечением cLAN в дисковод.

   b. Дождитесь появления меню автоматической установки с компакт-диска.

   c. Выберите “Install cLAN Management Console”.

   d. Повторите процедуру установки для каждого из оставшихся узлов кластера.

   Программное обеспечение консоли управления GigaNet cLAN можно также сгрузить с сайта GigaNet: http://www.giganet.com. Следуйте инструкциям на странице поддержки сайта GigaNet.

   Установка консоли управления cLAN может вызвать перезагрузку узла.

5. Проверьте работоспособность аппаратных модулей GigaNet. Это можно сделать так:

   a. Откройте GMC. (`Программы->GigaNet->cLAN Management Console`)

   b. Появится диалоговое окно, показывающее все доступные компьютеры в сети. Нажмите клавишу **ESC**.

   c. В полосе меню выберите **Console—>Local**.

d. Подтвердите, что показаны все члены кластера, и все они “активны”.

e. В полосе меню выберите **Utilities—>VI Throughput**. При этом запускается тест производительности, позволяющий убедиться, что данные действительно передаются через оборудование.

f. Введите в верхнем регистре имена компьютеров для двух узлов, используемых в тестировании. Укажите локальный узел и узел источника.

g. Нажмите кнопку **Start Measuring**. Скорость передачи данных должна быть не меньше 65 Мб/с.

h. Нажмите кнопку **Stop Measuring**, чтобы остановить тестирование соединения.

i. Повторите тест для других узлов кластера, измеряя скорость обмена данными между локальным узлом (Source) и другими узлами (Sink).

Если тест соединения не работает, обратитесь разделам по устранению неисправностей руководств *GigaNet GNN1000 User Guide* и *GigaNet GNX5000 User Guide*.

Информацию по установке и запуску DB2 Universal Database for Windows NT смотрите в книге *DB2 Enterprise - Extended Edition for Windows Quick Beginnings*.

Процедура установки ServerNet Interconnect

Ниже приводится список оборудования и программ, необходимых для установки этой среды:

- ServerNet PCI Adapter Driver (SPAD), (ID продукта T0089), версия 1.3.5 или более новая
- ServerNet Switch 1
- ServerNet Area Network Manager (SANMan), (ID продукта T0087), версия 1.1.3 или более новая.

Ниже описаны действия, которые надо выполнить, что ServerNet Interconnect мог работать с DB2 Universal Database. Каждый шаг описан кратко, подробности здесь не приводятся. Подробные инструкция и указания смотрите в справочной документации по каждому этапу.

Этапы, приведенные ниже, подразумевают, что в кластере не более шести (6) узлов. Если требуется использовать больше шести узлов, обратитесь к ServerNet.

Описание этапов:

1. Установите сетевую плату ServerNet. Инструкции по установке смотрите в книге *ServerNet-I Virtual Interface Software Release Document, (product ID N0031)*.
2. Установите ServerNet Switch 1. Инструкции по установке смотрите в книге *ServerNet-I Virtual Interface Software Release Document, (product ID N0031)*.

3. Деинсталлируйте ранее установленные драйверы ServerNet. (Пропустите этот этап, если вы устанавливаете ServerNet впервые.)
   a. Откройте панель управления Сеть. (`Пуск->Настройка->Панель управления->Сеть`)
   b. Выберите закладку **Адаптеры**.
   c. Удалите драйвер адаптера Tandem ServerNet PCI.
   d. Выберите закладку **Службы**.
   e. Удалите SANMan.
   f. Выберите закладку **Протоколы**.
   g. Удалите протокол Tandem ServerNet-I VI.
4. Установите драйвер адаптера Tandem ServerNet PCI. Если вы устанавливаете программное обеспечение с компакт-диска, поставляемого ServerNet:
   a. Откройте панель управления Сеть. (`Пуск->Настройка->Панель управления->Сеть`)
   b. Выберите закладку **Адаптеры**. (Появится панель с адаптерами).
   c. Убедитесь, что новый драйвер ServerNet находится на отдельном диске и/или в отдельном каталоге. После этого в командной строке, указав правильный диск и/или каталог, введите “ernnn.exe -d”, чтобы запустить самораспаковывающуюся программу. (Имя файла “ernnn.exe” означает Engineering Release с числом nnn, указывающим номер устанавливаемой версии драйвера ServerNet.)
   d. Перейдите к диску и/или каталогу, где находятся извлеченные файлы. Перейдите в подкаталог “Spad n.n.n \ Free” (где “n.n.n” номер версии программного продукта). (Если вы работаете в среде отладки или разработки, перейдите в подкаталог “Spad n.n.n \ Checked”, а не в подкаталог “Spad n.n.n \ Free”.)
   e. Измените имя файла “oemsetup.multi_node” на “oemsetup.inf”.
   f. В закладке Адаптеры выберите **Добавить**. (Появится панель Выбрать адаптер.)
   g. Выберите **Установить с диска...**. (Появится экран Вставьте диск).
   h. Введите диск и/или каталог, где находится файл oemsetup.inf.
   i. Убедитесь, что в диалоговом окне показан “драйвер адаптера Tandem ServerNet PCI”, и нажмите кнопку **OK**. Убедитесь, что в списке адаптеров теперь показан адаптер ServerNet. Нажмите кнопку **Закрыть**.
   j. Нажмите кнопку **Да**, чтобы перезапустить компьютер. Можно нажать кнопку **Нет**, чтобы продолжить установку SANMan и VI Software Developer’s Kit (SDK).
5. Установите SANMan. Если вы устанавливаете программное обеспечение с компакт-диска, поставляемого ServerNet:
   a. Откройте панель управления Сеть. (`Пуск->Настройка->Панель управления->Сеть`)

b. Выберите закладку **Службы**. (Появится панель Службы).

c. Убедитесь, что новый драйвер ServerNet находится на отдельном диске и/или в отдельном каталоге. После этого в командной строке, указав правильный диск и/или каталог, введите “ernnn.exe -d”, чтобы запустить самораспаковывающуюся программу. (Имя файла “ernnn.exe” означает Engineering Release с числом nnn, указывающим номер устанавливаемой версии драйвера ServerNet.)

d. На странице Службы выберите **Добавить**. (Появится экран Выберите службы).

e. Перейдите к диску и/или каталогу, где находятся извлеченные файлы. Перейдите в подкаталог “SANMan n.n.n \ Free” (где “n.n.n” номер версии программного продукта). (Если вы работаете в среде отладки или разработки, перейдите в подкаталог “SANMan n.n.n \ Checked”, а не в подкаталог “SANMan n.n.n \ Free”.)

f. Определите состояние коммутатора - X или Y, посмотрев на световой индикатор коммутатора. На одной лампочке написано “X”, а на другой - “Y”.

g. Для состояния X выберите X=1 и Y=0. Убедитесь, что все кабели подсоединены к порту X сетевых плат.

h. Для состояния Y выберите X=0 и Y=1. Убедитесь, что все кабели подсоединены к порту Y сетевых плат.

i. Задайте номер порта коммутатора, к которому подключена сетевая плата данного компьютера.

j. Выберите “PC” для всех шести (6) портов.

6. Установите протокол виртуального интерфейса. Если вы устанавливаете программное обеспечение с компакт-диска, поставляемого ServerNet:

   a. Откройте панель управления Сеть. (`Пуск->Настройка->Панель управления->Сеть`)

   b. Выберите закладку **Протоколы**. (Появится экран Сетевые протоколы).

   c. Убедитесь, что новый драйвер ServerNet находится на отдельном диске и/или в отдельном каталоге. После этого в командной строке, указав правильный диск и/или каталог, введите “ernnn.exe -d”, чтобы запустить самораспаковывающуюся программу. (Имя файла “ernnn.exe” означает Engineering Release с числом nnn, указывающим номер устанавливаемой версии драйвера ServerNet.)

   d. На странице Протоколы выберите **Добавить**. (Появится панель Выберите сетевые протоколы).

   e. Выберите **Установить с диска...**. (Появится экран Вставьте диск).

   f. Введите имя диска и/или каталога, где находятся извлеченные файлы.

7. Проверьте работоспособность оборудования ServerNet. Специальных программ тестирования нет. Поэтому для тестирования оборудования ServerNet используйте DB2.

Если аппаратные модули не работают, обратитесь за помощью к руководству *ServerNet-I Virtual Interface Software Release Document, (product ID N0031)*.

Информацию по установке и запуску DB2 Universal Database for Windows NT смотрите в книге *DB2 Enterprise - Extended Edition for Windows Quick Beginnings*.

Процедура установки Synfinity Interconnect

Ниже приводится список оборудования и программ, необходимых для установки этой среды:

- Сетевая плата Synfinity PCI
- Шестипортовый коммутатор Synfinity
- Соединительные кабели Synfinity
- Программное обеспечение диспетчера кластеров Synfinity, версия 1.10.

Ниже описаны действия, которые надо выполнить, что Synfinity Interconnect мог работать с DB2 Universal Database. Каждый шаг описан кратко, подробности здесь не приводятся. Подробные инструкция и указания смотрите в справочной документации по каждому этапу.

В комплект Synfinity входит компакт-диск программным обеспечением диспетчера кластеров Synfinity, Версия 1.10. Этот диск содержит все необходимые программы и документацию для установки Synfinity Interconnect. Кроме того, он содержит руководство *Synfinity Cluster Manager Software User Guide*.

При наличии других установленных аппаратных и программных средств и протоколов VI перед установкой Synfinity Synfinity Interconnect может потребоваться их удалить.

Установленное оборудование Synfinity может быть неизвестен Windows NT и не отображаться на панели управления Windows NT.

Сводка этапов:

1. Установите сетевые платы
2. Установите программное обеспечение Диспетчера кластеров Synfinity
3. Установите коммутаторы и кабели
4. Проверьте соединение

Описание этапов:

1. Установите сетевую плату Synfinity PCI. Инструкции по установке смотрите в руководстве *Synfinity Cluster Manager Software User Guide*.

2. Установите программное обеспечение диспетчера кластеров Synfinity на узле, соединенном с коммутатором. Инструкции по установке смотрите в руководстве *Synfinity Cluster User Guide*.

   Выбранный вами узел будет диспетчером кластера. Это единственный узел, на котором вы должны установить программное обеспечение с компакт-диска.

   После установки надо произвести запуск программного обеспечения диспетчера кластеров Synfinity. Диспетчер кластеров выдаст вам план кластера и поможет поэтапно пройти настроить сеть, а также предложит лучшие опции маршрутизации и кабельной разводки.Этот этап следует выполнить до подключения кабелей к коммутаторам и сетевым платам Synfinity. В качестве составной части процесса планирования Диспетчер кластеров использует план кластера для создания установочных дискет для других узлов. На них будут помещены драйверы для сетевых плат, устанавливаемых на других узлах. Подробную информацию смотрите в руководстве *Synfinity Cluster Manager Software User Guide*.
3. Установите коммутатор и кабели Synfinity. Инструкции по установке смотрите в руководстве *Synfinity Cluster User Guide*.
4. Проверьте работоспособность оборудования Synfinity. Это можно сделать так:
   a. На любой системе кластера откройте окно командной строки в Windows NT.
   b. Перейдите в подкаталог ″utils″ каталога, куда загружено программное обеспечение диспетчера кластеров Synfinity.
   c. Введите ″vitest″ и запишите номер узла, который будет показан.
   d. Перейдите на любую другую систему кластера и откройте окно командной строки.
   e. Перейдите в подкаталог ″utils″ каталога, куда загружено программное обеспечение диспетчера кластеров Synfinity на этой другой системе.
   f. Введите ″vitest x″, где x - номер узла, который вы записали ранее на этапе c.
   g. Должно появиться сообщение ″CONNECTION GOOD″.
   h. Если появилось сообщение ″NO CONNECTION″, проверьте установку кабелей и аппаратных модулей; дополнительную информацию по устранению неисправностей смотрите в руководстве *Synfinity Cluster Manager Software User Guide*. Посмотрите также страницы сайта технических советов (″Tech-tips″): http://www.fjst.com/

Информацию по установке и запуску DB2 Universal Database for Windows NT смотрите в книге *DB2 Enterprise - Extended Edition for Windows Quick Beginnings*.

Подготовка DB2 к работе с использованием VI

Подробную информацию по установке смотрите в руководстве *DB2 Enterprise - Extended Edition for Windows Quick Beginnings*.

По завершении установки DB2 согласно описанию *DB2 Enterprise - Extended Edition for Windows Quick Beginnings* задайте следующие переменные реестра DB2 и выполните следующие задачи на каждом разделе базы данных в этом экземпляре:

1. Задайте DB2_VI_ENABLE=ON

   Используйте команду **db2set**, чтобы изменить значение этой переменной реестра. Используйте команду **db2_all**, чтобы выполнить команду **db2set** на всех серверах разделов базы данных в этом экземпляре. Для выполнения команды **db2_all** вы должны зарегистрироваться с именем пользователя, входящим в группу администраторов.

   В следующем примере внутрь двойных кавычек помещен символ ;, чтобы этот запрос мог выполняться одновременно на всех серверах разделов базы данных в этом экземпляре:

   ```
   db2_all ";db2set DB2_VI_ENABLE=ON"
   ```

   Дополнительную информацию о команде **db2_all** смотрите в разделе ″Выполнение команд на нескольких серверах разделов базы данных″ руководства *Administration Guide: Implementation*.

2. Задайте DB2_VI_DEVICE=nic0

   Например:

   ```
   db2_all ";db2set DB2_VI_DEVICE=nic0"
   ```

   Примечание: Для соединения Synfinity значение этой переменной должно быть `DB2_VI_DEVICE=VINIC`. Имя устройства (VINIC) должно быть введено в верхнем регистре.

3. Задайте DB2_VI_VIPL=vipl.dll

   Например:

   ```
   db2_all ";db2set DB2_VI_VIPL=vipl.dll"
   ```

   Примечание: В этом примере используется значение по умолчанию для этой переменной реестра. Дополнительную информацию об этих переменных реестра смотрите в книге *Administration Guide: Performance*.

4. Введите `db2start` на многораздельном экземпляре.
5. Просмотрите файл db2diag.log. Для каждого раздела должно быть одно сообщение, что “VI включен.”
6. Может потребоваться обновление параметров конфигурации FCM (Fast Communications Manager - менеджер быстрой связи). Если проблема заключается в ограниченности ресурсов, касающихся FCM, следует

увеличить значения параметров настройки FCM. Если вы переходите из другой среды высокоскоростного соединения, для которой вы увеличили значения параметров настройки FCM, может понадобиться уменьшить эти значения. Кроме того, в Windows NT может потребоваться перезадать переменную реестра DB2NTMEMSIZE, изменив заданное DB2 значение по умолчанию. Дополнительную информацию об этих переменных реестра смотрите в книге *Administration Guide: Performance*.

Приложение J. Службы каталогов протокола LDAP

Протокол LDAP (Lightweight Directory Access Protocol - протокол упрощенного доступа к каталогам) - это промышленный стандарт для метода доступа к службам каталогов. Служба каталогов хранит информацию о ресурсах для множества систем и служб в распределенной среде и обеспечивает клиенту и серверу доступ к этим ресурсам. 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.

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, 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: Значение реестра DB2LDAPCACHE применимо только к базам данных и каталогам узлов.

Поддержка конфигураций клиента и сервера LDAP

В следующей таблице перечислены поддерживаемые конфигурации клиента и сервера LDAP:

Таблица 26. Поддерживаемые конфигурации клиента и сервера LDAP

	IBM SecureWay Directory Версия 3.1 и 3.1.1	**Microsoft Active Directory**
Клиент LDAP IBM	Поддерживается	Не поддерживается
Клиент LDAP/ADSI Microsoft	Поддерживается	Поддерживается

IBM SecureWay Directory, Версия 3.1 - сервер LDAP версии 3 для систем Windows NT, AIX и Solaris. SecureWay Directory поставляется как часть базовой операционной системы в AIX и 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.

Поддержка клиента LDAP Microsoft включена в следующие продукты Microsoft:

- Outlook 98, Outlook 2000 и Outlook Express

  Примечание: Outlook Express устанавливается как часть программы Microsoft Internet Explorer.
- Exchange Server
- Windows NT Server, Service Pack 4
- Windows 98 Second Edition
- Windows 2000

Поддержка клиента LDAP Microsoft включена также в компонент Active Directory Service Interface (ADSI). Последнюю версию ADSI можно загрузить с сайта Microsoft:

http://www.microsoft.com/windows2000/techinfo/howitworks/activedirectory/adsilinks.asp

В операционных системах Windows 98, Windows NT и Windows 2000 DB2 поддерживает использование клиента LDAP IBM или LDAP Microsoft для доступа к серверу IBM SecureWay Directory. Если клиента LDAP Microsoft нет, DB2 пытается использовать клиент LDAP IBM. Чтобы явно указать выбор клиента LDAP IBM, с помощью команды **db2set** задайте для переменной реестра DB2LDAP_CLIENT_PROVIDER значение “IBM”.

Поддержка Active Directory в Windows 2000

DB2 использует 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, используемом для связи с сервером баз данных (информация связывания), получается от объекта ibm_db2Node, по которым создан объект ibm_db2Database.

Конфигурирование DB2 для работы с Active Directory

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.

Конфигурирование DB2 в среде LDAP IBM

Чтобы использовать DB2 в среде LDAP IBM, необходимо на каждом компьютере сконфигурировать:

- 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**.
- 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 содержит объекты DB2.

- Имя (DN) и пароль пользователя LDAP. Они требуются только для хранения с помощью LDAP информации DB2, относящейся к отдельным пользователям.

Создание пользователя LDAP

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.

Конфигурирование пользователя LDAP для прикладных программ DB2

При работе с клиентом LDAP IBM перед запуском DB2 следует создать имя (DN) пользователя LDAP и пароль для текущего зарегистрированного пользователя. Это можно сделать с помощью утилиты db2ldcfg:

```
db2ldcfg -u <DNпользователя> -w <пароль> -> задать DN и пароль пользователя
         -r                              -> удалить DN и пароль пользователя
```

Например:

```
db2ldcfg -u "cn=Mary Burnnet,ou=DB2 UDB Development,ou=Toronto,o=ibm,c=ca"
         -w пароль
```

Регистрация серверов DB2 после установки

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. Регистрируя экземпляр сервера баз данных, необходимо указать *имя узла*. Имя узла используется клиентскими программами, когда они подключаются к серверу или устанавливают с ним соединение. Можно внести 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 имеет следующий вид:

```
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 для DB2 Universal Database EEE, включающего несколько физических компьютеров, команду REGISTER следует выполнить по одному разу для каждого компьютера. Чтобы выполнить команду REGISTER на всех компьютерах, используйте команду *rah*.

Примечание: Нельзя использовать одно `имя_узла_ldap` на всех компьютерах, поскольку оно должно быть уникально в LDAP. Необходимо будет заменить в команде REGISTER `имя_узла_ldap` на имя хоста каждого компьютера. Например:

```
rah ">DB2 REGISTER DB2 SERVER IN LDAP AS <> PROTOCOL TCPIP"
```

На каждом компьютере, где выполняется команда `rah`, символ "<>" заменяется на имя хоста. В редких случаях, когда имеется несколько экземпляров DB2 Universal Database EEE, в качестве имени узла в команде *rah* можно использовать сочетание индексов экземпляра и хоста.

Команду REGISTER можно выполнить с 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.

Дополнительную информацию о команде REGISTER смотрите в книге *Command Reference*.

Изменение протокола для сервера DB2

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.
- Адрес IPX.
- 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>
```

Дополнительную информацию о команде UPDATE LDAP смотрите в книге *Command Reference*.

Создание алиаса узла для команды ATTACH

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>
```

Отмена регистрации сервера DB2

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.

Регистрация баз данных

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"
```

Подключение к удаленному серверу

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.

Отмена регистрации базы данных

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>
```

Обновление записей LDAP в локальных каталогах баз данных и узлов

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.

Поиск

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.

Регистрация баз данных хоста

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”.

Задание значений реестра DB2 на уровне пользователя

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
  ```

Включение поддержки LDAP после установки

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.

Отключение поддержки LDAP

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”.

Поддержка LDAP и DB2 Connect

Если на шлюзе DB2 Connect поддерживается LDAP, а база данных отсутствует в каталоге баз данных шлюза, DB2 выполнит поиск в LDAP и постарается сохранить полученные сведения.

Особенности защиты

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 к него нет полномочий администратора каталогов.

Примечание: Проверка авторизации всегда осуществляется сервером LDAP, а не DB2. Проверка авторизации LDAP не связана 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. Дополнительную информацию о командах DB2 смотрите в книге *Command Reference*, а об API DB2 - в книге *Administrative API Reference*.

Особенности защиты в Active Directory Windows 2000

Объекты баз данных и узлов DB2 создаются как подчиненные объекту компьютера, где сервер DB2 установлен в Active Directory. 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 Users and Computer* так:

1. Запустите средство управления *Active Directory Users and Computer*

(Пуск —> Программы —> Средства управления —> Active Directory Users and Computer)

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 Users and Computer* так:

1. Запустите средство управления *Active Directory Users and Computer*

   (Пуск —> Программы —> Средства управления —> Active Directory Users and Computer)
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”

Дополнение схемы каталога 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.

Дополнение схемы каталога для IBM eNetwork Directory версии 2.1

При использовании IBM eNetwork Directory версии 2.1 необходимо дополнить базовую схему классами объектов и атрибутами, которые используются DB2.

Чтобы дополнить базовую схему IBM eNetwork Directory версии 2.1, выполните следующие шаги:

1. Скопируйте файл определений атрибутов DB2 `db2.at` и файл определений классов объектов `db2.oc` в каталог, содержащий файлы системных атрибутов и определений классов объектов (`slapd.at.conf` и `slapd.oc.conf`). Файлы определений атрибутов и классов объектов DB2 находятся в подкаталоге `cfg` подкаталога `sqllib`. Файлы системных атрибутов и определений классов объектов расположены в подкаталоге `etc` подкаталога `%LDAPHome%`.
2. Просмотрите файлы определений атрибутов и классов объектов DB2. Закомментируйте все классы и атрибуты, определенные в текущей схеме каталога LDAP.
3. В конце файла `slapd.oc.conf` добавьте следующую строку:
   ```
   include db2.oc
   ```
4. В конце файла `slapd.at.conf` добавьте следующую строку:
   ```
   include db2.at
   ```
5. Перезапустите сервер LDAP.

Дополнение схемы каталога для Active Directory Windows 2000

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.

Эта команда используется так:

```
db2schex
```

С этой командой используются также необязательные условия:

- –b DNпользователя

  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. Значением условия DNпользователя может быть 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
  ```

  Или

  ```
  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.

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.

Классы объектов и атрибуты, используемые DB2

Ниже в таблицах описаны классы объектов, используемые DB2:

Таблица 27. 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
Тип	abstract
OID (идентификатор объекта)	1.3.18.0.2.6.132
GUID (глобальный уникальный идентификатор)	b3afd63f-5c5b-11d3-b818-002035559151

Таблица 28. 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

Таблица 28. cimSetting (продолжение)

Класс	**cimSetting**
Обязательные атрибуты	
Необязательные атрибуты	settingID
Тип	abstract
OID (идентификатор объекта)	1.3.18.0.2.6.131
GUID (глобальный уникальный идентификатор)	b3afd64d-5c5b-11d3-b818-002035559151

Таблица 29. eProperty

Класс	**eProperty**
Имя дисплея LDAP Active Directory	ibm-eProperty
Общее имя (cn) Active Directory	ibm-eProperty
Описание	Используется для задания значений пользовательских свойств для конкретной программы
Надкласс	cimSetting
Обязательные атрибуты	
Необязательные атрибуты	propertyType cisPropertyType cisProperty cesPropertyType cesProperty binPropertyType binProperty
Тип	structural
OID (идентификатор объекта)	1.3.18.0.2.6.90
GUID (глобальный уникальный идентификатор)	b3afd69c-5c5b-11d3-b818-002035559151

Таблица 30. DB2Node

Класс	**DB2Node**
Имя дисплея LDAP Active Directory	ibm-db2Node
Общее имя (cn) Active Directory	ibm-db2Node
Описание	Представляет сервер DB2
Надкласс	eSap / ServiceConnectionPoint

Таблица 30. DB2Node (продолжение)

Класс	**DB2Node**
Обязательные атрибуты	db2nodeName
Необязательные атрибуты	db2nodeAlias db2instanceName db2Type host / dNSHostName (см. Примечание 2) protocolInformation/ServiceBindingInformation
Тип	structural
OID (идентификатор объекта)	1.3.18.0.2.6.116
GUID (глобальный уникальный идентификатор)	b3afd65a-5c5b-11d3-b818-002035559151
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 IPX/SPX:

- “IPXSPX”
- Адрес IPX

Прием IPX/SPX доступен на серверах (но не клиентах) DB2 for AIX и Solaris. NetBIOS и NPIPE не поддерживаются в AIX и Solaris.

Элементы для NetBIOS:

- “NETBIOS”
- Имя рабочей станции сервера NetBIOS

Элементы для именованных конвейеров:

- “NPIPE”
- Имя компьютера сервера
- Имя экземпляра сервера

Таблица 31. 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

Таблица 31. DB2Database (продолжение)

Класс	**DB2Database**
Необязательные атрибуты	db2databaseAlias db2additionalParameter db2ARLibrary db2authenticationLocation db2gwPtr db2databaseRelease DCEPrincipalName
Тип	structural
OID (идентификатор объекта)	1.3.18.0.2.6.117
GUID (глобальный уникальный идентификатор)	b3afd659-5c5b-11d3-b818-002035559151

Таблица 32. db2additionalParameters

Атрибут	**db2additionalParameters**
Имя дисплея LDAP Active Directory	ibm-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 33. 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

Таблица 33. db2authenticationLocation (продолжение)

Атрибут	**db2authenticationLocation**
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 34. 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 35. 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 36. 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

Таблица 36. db2databaseName (продолжение)

Атрибут	**db2databaseName**
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 37. 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 38. 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 39. 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

Таблица 39. db2nodeName (продолжение)

Атрибут	**db2nodeName**
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 40. 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
Особые замечания	Это отношение позволяет клиенту находить информацию о протоколе связи для соединения с базой данных.

Таблица 41. 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 42. 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 43. 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
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 44. 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 45. 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 46. 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 47. cisProperty

Атрибут	**cisProperty**
Имя дисплея LDAP Active Directory	ibm-cisProperty
Общее имя (cn) Active Directory	ibm-cisProperty

Таблица 47. cisProperty (продолжение)

Атрибут	**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 48. cisPropertyType

Атрибут	**cisPropertyType**
Имя дисплея LDAP Active Directory	ibm-cisPropertyType
Общее имя (cn) Active Directory	ibm-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 49. binProperty

Атрибут	**binProperty**
Имя дисплея LDAP Active Directory	ibm-binProperty
Общее имя (cn) Active Directory	ibm-binProperty

Таблица 49. binProperty (продолжение)

Атрибут	**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 50. binPropertyType

Атрибут	**binPropertyType**
Имя дисплея LDAP Active Directory	ibm-binPropertyType
Общее имя (cn) Active Directory	ibm-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 51. 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

Таблица 51. PropertyType (продолжение)

Атрибут	**PropertyType**
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 52. 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
GUID (глобальный уникальный идентификатор)	b3afd596-5c5b-11d3-b818-002035559151

Приложение K. Расширение Центра управления

В Версии 7 функциональность Центра управления DB2 Universal Database можно расширить при помощи новой архитектуры *расширений* (plug-in).

Идея архитектуры *расширений* заключается в возможности добавления новых пунктов всплывающее меню Центра управления для данного объекта и новых кнопок 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.

Вопросы производительности

Модули расширения (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.

Вопросы упаковки

Файлы класса расширения необходимо упаковывать архиватором ZIP в соответствии с правилами файла архива Java. Чтобы инструменты Центра управления можно было запустить как прикладную программу, ZIP-файл (db2plug.zip) должен находиться в пути *classpath*. Чтобы инструменты Центра управления можно было запустить как апплет, ZIP-файл должен располагаться по адресу, записанному в тэге <codebase> в файле html Центра управления.

ZIP-файл должен быть построен без сжатия и поддерживать относительные пути для всех файлов класса (`zip -r0 db2plug.zip *.class`).

Описания интерфейсов

Поставляются следующие интерфейсы:

- CCExtension
- CCObject
- CCMenuAction
- CCToolbarAction.

Эти интерфейсы описаны в следующих разделах с примерами.

CCExtension

Интерфейс CCExtension позволяет расширить пользовательский интерфейс Центра управления, добавляя к нему новые кнопок панели инструментов, новые пунктов меню и переопределяя существующие действия меню.

Внешний интерфейс определяется так:

```
public interface CCExtension
{
   /**
    * Получить массив объектов подкласса CCObject, определяющих
    * список объектов, которые надо вставить или переопределить в
    * Центре управления
    * @return CCObject[] массив объектов подкласса CCObject
    */
   public CCObject[] getObjects();

   /**
    * Получить массив объектов подкласса CCToolbarAction, представляющих
    * список кнопок, которые надо добавить на главную панель инструментов
    * Центра управления.
    * @return CCToolbarAction[] массив объектов подкласса CCToolbarAction
    */
   public CCToolbarAction[] getToolbarActions();
}
```

Чтобы воспользоваться интерфейсом CCExtension, создайте класс Java, импортирующий пакет ″com.ibm.db2.tools.cc.navigator″ и реализующий данный интерфейс. Новый класс должен обеспечивать реализацию методов getObjects() и getToolbarActions().

Метод getObjects() возвращает массив CCObject, определяющий существующие объекты, для которых пользователь хотел бы добавить новые действия меню или удалить предопределенный набор действий меню.

Метод getToolbarActions() возвращает массив CCToolbarAction, который будет добавлен на главную панель меню Центра управления.

Для определения расширений Центра управления можно использовать один файл подкласса CCExtension или же несколько таких файлов. Чтобы Центр управления мог использовать эти расширения, используйте следующую процедуру установки:

1. Создайте файл ″db2plug.zip″, содержащий все файлы подкласса CCExtension. Файлы не должны быть сжатыми. Например, если файлы CCExtension содержатся в пакете модуля расширения и находятся в каталоге модуля расширения, используйте такую команду:

   ```
   zip -r0 db2plug.zip plugin\*.class
   ```

Эта команда помещает все файлы класса пакета модуля расширения в файл db2plug.zip с сохранением информации об их относительных путях.

2. Чтобы Центр управления можно было запустить как апплет, поместите файл db2plug.zip по адресу, записанному в тэге <codebase> файла html Центра управления. Чтобы запустить Центр управления как прикладную программу, поместите файл db2plug.zip в каталог, заданный переменной среды CLASSPATH.

Если браузер поддерживает несколько архивов, достаточно добавить ″db2plug.zip″ в список архивов страницы html Центра управления. Иначе все файлы подклассов CCExtension, CCObject, CCToolbarAction и CCMenuAction должны находиться в своих относительных каталогах, в зависимости от того, к какому пакету они принадлежат.

CCObject

Интерфейс CCObject позволяет изменить действия меню для существующего объекта.

Внешний интерфейс определяется так:

```
public interface CCObject
{
   /**
    * Следующие статические константы определяют список типов объектов,
    * которые можно добавить в дерево Центра управления.
    */
   public static final int UDB_SYSTEMS_FOLDER                          = 0;
   public static final int UDB_SYSTEM                                  = 1;
   public static final int UDB_INSTANCES_FOLDER                        = 2;
   public static final int UDB_INSTANCE                                = 3;
   public static final int UDB_DATABASES_FOLDER                        = 4;
   public static final int UDB_DATABASE                                = 5;
   public static final int UDB_TABLES_FOLDER                           = 6;
   public static final int UDB_TABLE                                   = 7;
   public static final int UDB_TABLESPACES_FOLDER                      = 8;
   public static final int UDB_TABLESPACE                              = 9;
   public static final int UDB_VIEWS_FOLDER                            = 10;
   public static final int UDB_VIEW                                    = 11;
   public static final int UDB_ALIASES_FOLDER                          = 12;
   public static final int UDB_ALIAS                                   = 13;
   public static final int UDB_TRIGGERS_FOLDER                         = 14;
   public static final int UDB_TRIGGER                                 = 15;
   public static final int UDB_SCHEMAS_FOLDER                          = 16;
   public static final int UDB_SCHEMA                                  = 17;
   public static final int UDB_INDEXES_FOLDER                          = 18;
   public static final int UDB_INDEX                                   = 19;
   public static final int UDB_CONNECTIONS_FOLDER                      = 20;
   public static final int UDB_CONNECTION                              = 21;
   public static final int UDB_REPLICATION_SOURCES_FOLDER              = 22;
   public static final int UDB_REPLICATION_SOURCE                      = 23;
   public static final int UDB_REPLICATION_SUBSCRIPTIONS_FOLDER        = 24;
   public static final int UDB_REPLICATION_SUBSCRIPTION                = 25;
   public static final int UDB_BUFFERPOOLS_FOLDER                      = 26;
```

```
public static final int UDB_BUFFERPOOL                                = 27;
public static final int UDB_APPLICATION_OBJECTS_FOLDER                 = 28;
public static final int UDB_USER_DEFINED_DISTINCT_DATATYPES_FOLDER     = 29;
public static final int UDB_USER_DEFINED_DISTINCT_DATATYPE             = 30;
public static final int UDB_USER_DEFINED_DISTINCT_FUNCTIONS_FOLDER     = 31;
public static final int UDB_USER_DEFINED_DISTINCT_FUNCTION             = 32;
public static final int UDB_PACKAGES_FOLDER                            = 33;
public static final int UDB_PACKAGE                                    = 34;
public static final int UDB_STORE_PROCEDURES_FOLDER                    = 35;
public static final int UDB_STORE_PROCEDURE                            = 36;
public static final int UDB_USER_AND_GROUP_OBJECTS_FOLDER              = 37;
public static final int UDB_DB_USERS_FOLDER                            = 38;
public static final int UDB_DB_USER                                    = 39;
public static final int UDB_DB_GROUPS_FOLDER                           = 40;
public static final int UDB_DB_GROUP                                   = 41;
public static final int UDB_DRDA_TABLE                                 = 42;

public static final int S390_SUBSYSTEMS_FOLDER                         = 43;
public static final int S390_SUBSYSTEM                                 = 44;
public static final int S390_BUFFERPOOLS_FOLDER                        = 45;
public static final int S390_BUFFERPOOL                                = 46;
public static final int S390_VIEWS_FOLDER                              = 47;
public static final int S390_VIEW                                      = 48;
public static final int S390_DATABASES_FOLDER                          = 49;
public static final int S390_DATABASE                                  = 50;
public static final int S390_TABLESPACES_FOLDER                        = 51;
public static final int S390_TABLESPACE                                = 52;
public static final int S390_TABLES_FOLDER                             = 53;
public static final int S390_TABLE                                     = 54;
public static final int S390_INDEXS_FOLDER                             = 55;
public static final int S390_INDEX                                     = 56;
public static final int S390_STORAGE_GROUPS_FOLDER                     = 57;
public static final int S390_STORAGE_GROUP                             = 58;
public static final int S390_ALIASES_FOLDER                            = 59;
public static final int S390_ALIAS                                     = 60;
public static final int S390_SYNONYMS_FOLDER                           = 61;
public static final int S390_SYNONYM                                   = 62;
public static final int S390_APPLICATION_OBJECTS_FOLDER                = 63;
public static final int S390_COLLECTIONS_FOLDER                        = 64;
public static final int S390_COLLECTION                                = 65;
public static final int S390_PACKAGES_FOLDER                           = 66;
public static final int S390_PACKAGE                                   = 67;
public static final int S390_PLANS_FOLDER                              = 68;
public static final int S390_PLAN                                      = 69;
public static final int S390_PROCEDURES_FOLDER                         = 70;
public static final int S390_PROCEDURE                                 = 71;
public static final int S390_DB_USERS_FOLDER                           = 72;
public static final int S390_DB_USER                                   = 73;
public static final int S390_LOCATIONS_FOLDER                          = 74;
public static final int S390_LOCATION                                  = 75;
public static final int S390_DISTINCT_TYPES_FOLDER                     = 76;
public static final int S390_DISTINCT_TYPE                             = 77;
public static final int S390_USER_DEFINED_FUNCTIONS_FOLDER             = 78;
public static final int S390_USER_DEFINED_FUNCTION                     = 79;
public static final int S390_TRIGGERS_FOLDER                           = 80;
```

```
public static final int S390_TRIGGER                                = 81;
public static final int S390_SCHEMAS_FOLDER                         = 82;
public static final int S390_SCHEMA                                 = 83;
public static final int S390_CATALOG_TABLES_FOLDER                  = 84;
public static final int S390_CATALOG_TABLE                          = 85;

public static final int DCS_GATEWAY_CONNECTIONS_FOLDER              = 86;
public static final int DCS_GATEWAY_CONNECTION                      = 87;

/**
 * Общее число типов объектов
 */
public static final int NUM_OBJECT_TYPES                            = 88;

/**
 * Получить имя этого объекта
 * Эта функция возвращает имя данного объекта. Имя объекта
 * может быть трех типов:
 * (1) Полное имя
 *     Синтаксис: xxxxx-yyyyy-zzzzz
 *             где xxxxx-yyyyy - полное имя
 *             родительского объекта, а zzzzz - имя нового объекта.
 *     Примечание: Имена родительского и дочернего объектов разделены
 *     символом '-'. Если для идентификации объекта требуется имя схемы,
 *     полное имя представляется в виде xxxxx-yyyyy-wwwww.zzzzz,
 *     где wwwww - имя схемы.
 *     Будет изменено только поведение объекта, который соответствует
 *     этому полному имени.
 * (2) Родительское полное имя
 *     Синтаксис: xxxxx-yyyyy
 *             где xxxxx-yyyyy - полное имя
 *             родительского объекта.
 *     Если тип объекта - папка (например, DATABASES_FOLDER),
 *     getName() должна вернуть лишь полное имя
 *     родительского объекта папки.
 *     Будет изменено только поведение объекта, который соответствует этому
 *     имени и конкретному типу, возвращаемому функцией getType().
 * (3) null (пустое значение)
 *     Синтаксис: null
 *     Если возвращен null, возвращаемые при вызове getActions() значения
 *     CCActions будут применены ко всем объектам типа, возвращаемого при
 *     вызове getType().
 * @return строка с именем объекта
 */
public String getName();

/**
 * Получить тип этого объекта
 * @return int одна из констант статического типа, определенная
 * в этом интерфейсе
 */
public int getType();

/**
 * Получить массив CCMenu Action, определяющий список действий меню,
```

```
     * которые надо создать для выбранного объекта
     * return CCMenuAction[] массив CCMenuAction
     */
    public CCMenuAction[] getMenuActions();

    /**
     * Проверяет возможность редактирования этого объекта.
     * Если объект нельзя редактировать, пункты меню, связанные с изменением,
     * будут удалены из всплывающего меню объекта.
     * return boolean Если false, пункт меню Изменить
     * будет удален из всплывающего меню объекта
     */
    public boolean isEditable();

    /**
     * Проверяет возможность конфигурирования этого объекта.
     * Если выдается значение "нет", пункты меню, связанные с конфигурированием,
     * будут удалены из всплывающего меню объекта
     * return boolean Если false, пункт меню, связанный с конфигурированием, будет
     * удален из всплывающего меню объекта
     */
    public boolean isConfigurable();
}
```

Примечание: На данный момент два последних метода в `CCObject`: `isEditable()` и `isConfigurable()` должны всегда возвращать значение **true**.

CCMenuAction

Интерфейс CCMenuAction позволяет определить новое действие, которое будет использовать объект Центра управления.

Внешний интерфейс определяется так:

```
public interface CCMenuAction
{
    /**
     * Получить имя этого действия
     * @return String Name текст в пункте меню
     */
    public String getMenuText();

    /**
     * Вызывается, когда происходит действие. Используйте метод getActionCommand()
     * из ActionEvent для получения полного имени
     * вызываемого объекта Центра управления.
     * @param e Событие действия
     */
    public void actionPerformed(ActionEvent e);
}
```

CCToolBarAction

Интерфейс CCToolbarAction позволяет определить новое действие на панели инструментов Центра управления.

Внешний интерфейс определяется так:

```
public interface CCToolbarAction
{
   /**
    * Получить имя этого действия
    * @return String Name текст в пункте меню или на панели инструментов
    * всплывающая справка о кнопках
    */
   public String getHoverHelpText();

   /**
    * Получить значок для кнопки полосы инструментов
    * Для любой CCAction эта функция должна возвращать
    * допустимый объект ImageIcon. В противном случае у кнопки не будет своего
    * значка. @return ImageIcon Значок для вывода на экран
    */
   public ImageIcon getIcon();

   /**
    * Вызывается, когда происходит действие.
    * @param e Событие действия
    */
   public void actionPerformed(ActionEvent e);
}
```

Сценарий использования

Код в следующем примере позволяет:

1. Изменить действия для базы данных SAMPLE (смотрите раздел “MySample.java” на стр. 460)
2. Изменить действия для всех объектов базы данных (смотрите раздел “MyDatabaseActions.java” на стр. 461)
3. Добавить новый объект экземпляра (смотрите раздел “MyInstance.java” на стр. 462)
4. Изменить действия для экземпляра DB2 (смотрите раздел “MyDB2.java” на стр. 462)
5. Изменить действия для папки Базы данных (смотрите раздел “MyDatabases.java” на стр. 463)
6. Изменить действия для таблицы SYSIBM.SYSPLAN (смотрите раздел “MySYSPLAN.java” на стр. 464)
7. Добавить новый объект таблицы (смотрите раздел “MyTable.java” на стр. 464)
8. Изменить действия для объекта DB_User под объектом Программы (смотрите раздел “MyDBUser.java” на стр. 465)
9. Добавить кнопку на панель инструментов Центра управления (смотрите раздел “MyToolbarAction.java” на стр. 466).

Главный файл расширения - MyExtension.java. Все файлы класса хранятся в каталоге модулей расширения и запакованы с помощью команды:

```
zip -r0 db2plug.zip plugin
```

Выходной файл db2plug.zip помещается затем в каталог CLASSPATH или codebase в зависимости от того, запущен ли Центр управления как прикладная программа или апплет.

MyExtension.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyExtension implements CCExtension
{
   public CCObject[] getObjects()
   {
      CCObject[] objs = new CCObject[10];
      objs[0] = new MySample();
      objs[1] = new MyDatabaseActions();
      objs[2] = new MyInstance();
      objs[3] = new MyDB2();
      objs[4] = new MyDatabases();
      objs[5] = new MySYSPLAN();
      objs[6] = new MyTable();
      objs[7] = new MyDBUser();
      return objs;
   }

   public CCAction[] getActions()
   {
      CCAction[] actions = new CCAction[1];
      actions[0] = new MyToolbarAction();
      return actions;
   }
}
```

MySample.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MySample implements CCObject
{
   public String getName()
   {
      return "LOCAL - DB2 - SAMPLE";
   }

   public int getType()
   {
      return DATABASE;
   }

   public javax.swing.ImageIcon getIcon()
```

```
    {
       return null;
    }

    public boolean isNew()
    {
       return false;
    }

    public CCAction[] getActions()
    {
       CCAction[] acts = new CCAction[2];
       acts[0] = new MyAlterAction();
       acts[1] = new MyAction();
       return acts;
    }

}
```

MyDatabaseActions.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyDatabaseActions implements CCObject
{
   public String getName()
   {
      return null;
   }

   public int getType()
   {
      return DATABASE;
   }

   public javax.swing.ImageIcon getIcon()
   {
      return null;
   }

   public boolean isNew()
   {
      return false;
   }

   public CCAction[] getActions()
   {
      CCAction[] acts = new CCAction[2];
      acts[0] = new MyDropAction();
      acts[1] = new MyAction();
      return acts;
   }

}
```

MyInstance.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyInstance implements CCObject
{
   public String getName()
   {
      return "LOCAL - MyInstance";
   }

   public int getType()
   {
      return INSTANCE;
   }

   public javax.swing.ImageIcon getIcon()
   {
      return null;
   }

   public boolean isNew()
   {
      return true;
   }

   public CCAction[] getActions()
   {
      CCAction[] acts = new CCAction[2];
      acts[0] = new MyAlterAction();
      acts[1] = new MyAction();
      return null;
   }

}
```

MyDB2.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyDB2 implements CCObject
{
   public String getName()
   {
      return "LOCAL - DB2";
   }

   public int getType()
   {
      return INSTANCE;
   }

   public javax.swing.ImageIcon getIcon()
   {
```

```
      return null;
   }

   public boolean isNew()
   {
      return false;
   }

   public CCAction[] getActions()
   {
      CCAction[] acts = new CCAction[3];
      acts[0] = new MyAlterAction();
      acts[1] = new MyAction();
      acts[2] = new MyCascadeAction();
      return acts;
   }
}
```

MyDatabases.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyDatabases implements CCObject
{
   public String getName()
   {
      return "LOCAL - DB2 - Databases";
   }

   public int getType()
   {
      return DATABASE;
   }

   public javax.swing.ImageIcon getIcon()
   {
      return null;
   }

   public boolean isNew()
   {
      return false;
   }

   public CCAction[] getActions()
   {
      CCAction[] acts = new CCAction[1];
      acts[0] = new MyCreateAction();
      return acts;
   }

}
```

MySYSPLAN.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MySYSPLAN implements CCObject
{
   public String getName()
   {
      return "LOCAL - DB2 - SAMPLE - SYSIBM - SYSPLAN";
   }

   public int getType()
   {
      return TABLE;
   }

   public javax.swing.ImageIcon getIcon()
   {
      return null;
   }

   public boolean isNew()
   {
      return false;
   }

   public CCAction[] getActions()
   {
      CCAction[] acts = new CCAction[2];
      acts[0] = new MyAlterAction();
      acts[1] = new MyAction();
      return acts;
   }

}
```

MyTable.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyTable implements CCObject
{
   public String getName()
   {
      return "LOCAL - DB2 - SAMPLE - SYSIBM - MyTable";
   }

   public int getType()
   {
      return TABLE;
   }

   public javax.swing.ImageIcon getIcon()
   {
```

```
        return null;
    }

    public boolean isNew()
    {
        return true;
    }

    public CCAction[] getActions()
    {
        CCAction[] acts = new CCAction[2];
        acts[0] = new MyAlterAction();
        acts[1] = new MyAction();
        return acts;
    }

}
```

MyDBUser.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyDBUser implements CCObject
{
    public String getName()
    {
        return "LOCAL - DB2 - TEST-DB Users";
    }

    public int getType()
    {
        return DB_USER;
    }

    public javax.swing.ImageIcon getIcon()
    {
        return null;
    }

    public boolean isNew()
    {
        return false;
    }

    public CCAction[] getActions()
    {
        CCAction[] acts = new CCAction[2];
        acts[0] = new MyAlterAction();
        acts[1] = new MyAction();
        return acts;
    }

}
```

MyToolbarAction.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;
import javax.swing.*;

public class MyToolbarAction extends CCAction
{
   public MyToolbarAction()
   {
      super("MyToolbarAction");
   }

   public ImageIcon getIcon()
   {
      return <Your icon>;
   }

   public boolean actionPerformed(String objectName)
   {
      System.out.println( "Действие выполнено, имя объекта = " +
                          objectName );
      return true;
   }
}
```

MyAlterAction.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyAlterAction extends CCAction
{
   public MyAlterAction()
   {
      super(0);
   }

   public boolean actionPerformed(String objectName)
   {
      System.out.println( "Действие изменения выполнено, имя объекта = " +
                          objectName );
      return true;
   }
}
```

MyAction.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyAction extends CCAction
{
   public MyAction()
   {
      super("MyAction");
   }
```

```
   public boolean actionPerformed(String objectName)
   {
      System.out.println( "Действие выполнено, имя объекта = " +
                          objectName );
      return true;
   }
}
```

MyDropAction.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyDropAction extends CCAction
{
   public MyDropAction()
   {
      super(1);
   }

   public boolean actionPerformed(String objectName)
   {
      System.out.println( "Действие отбрасывания выполнено, имя объекта = " +
                          objectName );
      return true;
   }
}
```

MyCascadeAction.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyCascadeAction extends CCAction
{
   public MyCascadeAction()
   {
      super(11,2);
   }

   public boolean actionPerformed(String objectName)
   {
      System.out.println( "Каскадное действие выполнено, имя объекта = " +
                          objectName );
      return true;
   }
}
```

MyCreateAction.java

```
package plugin;
import com.ibm.db2.tools.cc.navigator.*;

public class MyCreateAction extends CCAction
{
   public MyCreateAction()
```

```
    {
       super(0);
    }

    public boolean actionPerformed(String objectName)
    {
       System.out.println( "Действие создания выполнено, имя объекта = " +
                           objectName );
       return true;
    }
}
```

Приложение L. Использование библиотеки DB2

Библиотека DB2 Universal Database состоит из электронной справки, книг (в формате PDF и HTML) и примеров программ в формате HTML. В этом разделе объясняется, какая информация содержится в ней и как ее получить.

Для оперативного доступа к этой информации можно использовать Информационный центр. Дополнительную информацию смотрите в разделе “Доступ к информации через Информационный центр” на стр. 484. Вы можете просматривать сведения о задачах, книги DB2, информацию по устранению неисправностей, программы примеров и информацию по DB2 в Web.

Файлы PDF и печатные книги DB2

Информация DB2

В следующей таблице книги DB2 разделены на 4 категории:

Руководства и справочники по DB2
В этих книгах содержится информация по DB2, общая для всех платформ.

Информация по установке и конфигурированию DB2
Эти книги применимы к DB2 для конкретной платформы. Например, есть отдельные книги *Быстрый старт* для DB2 на OS/2, Windows и на платформах на основе UNIX.

Кроссплатформенные программы примеров в формате HTML
Эти примеры - HTML-версии программ примеров, которые устанавливаются с клиентом разработки программ. Примеры используются для справок и не заменяют самих программ.

Замечания по выпуску
Эти файлы содержат самую свежую информацию, которую не успели включить в книги по DB2.

Руководства по установке, замечания по выпуску и обучающие книги в формате HTML можно просматривать прямо на компакт-диске. Большинство книг доступны в формате HTML на компакт-диске данного продукта (для просмотра) и в формате Adobe Acrobat (PDF) на компакт-диске публикаций DB2 (для просмотра и печати). Можно также заказать печатные копии в IBM; смотрите раздел “Заказ печатных копий” на стр. 480. Ниже в таблице перечислены книги, которые можно заказать.

На платформах OS/2 и Windows файлы в формате HTML можно установить в каталог `sqllib\doc\html`. Информация о DB2 переведена на различные языки, однако не на каждом языке доступна вся информация. Если информация на конкретном языке недоступна, приводится информация на английском языке.

На платформах UNIX вы можете установить версии файлов в формате HTML на нескольких языках в подкаталоги `doc/%L/html`, где *%L* - обозначение вашей национальной версии. Дополнительную информацию смотрите в соответствующей книге *Quick Beginnings* (Быстрый старт).

Вызвать книги DB2 и обратиться к информации в них можно разными способами:

- “Просмотр информации на экране” на стр. 483
- “Поиск электронной информации” на стр. 488
- “Заказ печатных копий” на стр. 480
- “Печать книг PDF” на стр. 479

Таблица 53. Информация DB2

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
	Руководства и справочники по DB2		
Administration Guide	*Administration Guide: Planning* содержит обзор понятий баз данных, информацию по вопросам разработки (в частности, по логическому и физическому проектированию баз данных) и обсуждение доступности баз данных. *Administration Guide: Implementation* содержит информацию о реализации ваших проектов, доступе к базам данных, аудите, резервном копировании и восстановлении. *Administration Guide: Performance* содержит информацию о среде баз данных, оценке и настройке производительности программ. Эти три тома *Administration Guide* можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8934.	SC09-2946 db2d1x70 SH43-0144 db2d2x70 SC09-2945 db2d3x70	db2d0

Таблица 53. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
Administrative API Reference	Описывает интерфейсы прикладного программирования (API) DB2 и структуры данных, которые можно использовать при работе с вашими базами данных. Эта книга также объясняет, как вызывать API из ваших программ.	SC09-2947 db2b0x70	db2b0
Application Building Guide	Содержит информацию о настройке среды и пошаговые инструкции для компиляции, компоновки и запуска программ DB2 в системах Windows, OS/2 и на платформах на базе UNIX.	SC09-2948 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	Содержит общие сведения о смысловых кодах APPC, CPI-C и SNA, которые могут встретиться вам при работе с продуктами DB2 Universal Database. Существует только в формате HTML.	Номера формы нет db2apx70	db2ap
Application Development Guide	Объясняет, как разрабатывать программы, обращающиеся к базам данных DB2 с использованием встроенного SQL или Java (JDBC и SQLJ). Эта книга содержит обсуждение программирования хранимых процедур, пользовательских функций, создания пользовательских типов, использования триггеров и разработки прикладных программ для работы в многораздельной среде и в системах объединения.	SC09-2949 db2a0x70	db2a0
CLI Guide and Reference	Объясняет, как разрабатывать программы, обращающиеся к базам данных DB2 при помощи интерфейса уровня вызовов (CLI) DB2 - интерфейса SQL, совместимого со спецификациями Microsoft ODBC.	SC09-2950 db2l0x70	db2l0
Command Reference	Объясняет, как использовать процессор командной строки, и описывает команды DB2, которые можно использовать для управления вашей базой данных.	SC09-2951 db2n0x70	db2n0

Таблица 53. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
Connectivity Supplement	Содержит установочную и справочную информацию по использованию DB2 for AS/400, DB2 for OS/390, DB2 for MVS, или DB2 for VM как реквестеров прикладных программ DRDA с серверами DB2 Universal Database. В этой книге описано также использование серверов прикладных программ DRDA с реквестерами прикладных программ DB2 Connect. Эта книга доступна только в форматах HTML и PDF.	Номера формы нет db2h1x70	db2h1
Data Movement Utilities Guide and Reference	Объясняет, как использовать утилиты DB2, в частности, import, export, load, AutoLoader и DPROP, которые упрощают перемещение данных.	SC09-2955 db2dmx70	db2dm
Data Warehouse Center Administration Guide	Содержит сведения о том, как построить и обслуживать хранилище данных при помощи Центра хранилища данных.	SC26-9993 db2ddx70	db2dd
Data Warehouse Center Application Integration Guide	Содержит информацию, которая поможет программистам интегрировать прикладные программы с Центром хранилища данных и Менеджером каталога данных.	SC26-9994 db2adx70	db2ad
DB2 Connect. Руководство пользователя	Содержит информацию по основным понятиям, программированию и общим вопросам использования продуктов DB2 Connect.	SC09-2954 db2c0x70	db2c0
DB2 Query Patroller Administration Guide	Содержит обзор системы DB2 Query Patroller, информацию по использованию и управлению, а также сведения по выполнению заданий при помощи утилит управления с графическим интерфейсом.	SC09-2958 db2dwx70	db2dw
DB2 Query Patroller User's Guide	Объясняет, как использовать средства и функции DB2 Query Patroller.	SC09-2960 db2wwx70	db2ww
Глоссарий	Содержит определения терминов, используемых в DB2 и его компонентах. Доступен в формате HTML, а также в книге *SQL Reference*.	Номера формы нет db2t0x70	db2t0

Таблица 53. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
Image, Audio, and Video Extenders Administration and Programming	Содержит общую информацию о модулях расширения DB2, о конфигурировании модулей расширения для работы с изображениями, звуком и видео (IAV), об управлении ими и о программировании с использованием модулей расширения IAV. Включает в себя справочную информацию, диагностическую информацию (с сообщениями) и примеры.	SC26-9929 dmbu7x70	dmbu7
Information Catalog Manager Administration Guide	Руководство по управлению каталогами данных.	SC26-9995 db2dix70	db2di
Information Catalog Manager Programming Guide and Reference	Содержит определения для проектирования интерфейсов менеджера каталогов данных.	SC26-9997 db2bix70	db2bi
Information Catalog Manager User's Guide	Содержит информацию об использовании пользовательского интерфейса менеджера каталогов данных.	SC26-9996 db2aix70	db2ai
Дополнение по установке и конфигурированию	Помогает планировать, устанавливать и конфигурировать клиенты DB2 для конкретных платформ. Это дополнение содержит также информацию по связыванию, конфигурированию связей клиента и сервера, инструментам DB2 с графическим интерфейсом, DRDA AS, распределенной установке, конфигурации распределенных запросов и доступу к неоднородным источникам данных.	GC09-2957 db2iyx70	db2iy
Message Reference	Содержит список сообщений и кодов, выдаваемых DB2, Information Catalog Manager, и Data Warehouse Center, и описывает для них рекомендуемые действия. Оба тома Message Reference можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8932.	Том 1 SC09-2978 db2m1x70 Том 2 SC09-2979 db2m2x70	db2m0
OLAP Integration Server Administration Guide	Объясняет, как использовать менеджер управления сервером OLAP Integration Server.	SC27-0782 db2dpx70	нет

Таблица 53. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
OLAP Integration Server Metaoutline User's Guide	Объясняет, как создавать и заполнять метамакеты OLAP при помощи стандартного интерфейса метамакетов OLAP (а не при помощи Metaoutline Assistant).	SC27-0784 db2upx70	нет
OLAP Integration Server Model User's Guide	Объясняет, как создавать и заполнять метамакеты OLAP при помощи стандартного интерфейса моделей OLAP (а не при помощи Model Assistant).	SC27-0783 db2lpx70	нет
Руководство по установке и использованию OLAP	Содержит информацию о конфигурировании и установке для Начального комплекта OLAP.	SH43-0137 db2ipx70	db2ip
Руководство пользователя надстройки электронных таблиц для Excel	Описывает, как использовать программу электронных таблиц Excel для анализа данных OLAP.	SH43-0141 db2epx70	db2ep
Руководство пользователя надстройки электронных таблиц для Lotus 1-2-3	Описывает, как использовать программу электронных таблиц Lotus 1-2-3 для анализа данных OLAP.	SH43-0140 db2tpx70	db2tp
Replication Guide and Reference	Содержит информацию по планированию, конфигурированию, управлению и использованию инструментов IBM Replication, поставляемых с DB2.	SC26-9920 db2e0x70	db2e0
Spatial Extender User's Guide and Reference	Содержит информацию по установке, конфигурированию, управлению, программированию и устранению неисправностей для DB2 Spatial Extender. Кроме того, содержит содержательное описание понятий пространственных данных и справочную информацию (сообщения и SQL) по модулю Spatial Extender.	SC27-0701 db2sbx70	db2sb
SQL Getting Started	Введение в основные понятия SQL и примеры для многих конструкций и задач.	SC09-2973 db2y0x70	db2y0

Таблица 53. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
SQL Reference, Том 1 и Том 2	Описывает синтаксис SQL, его семантику и правила языка. Эта книга включает также информацию о совместимости версий, ограничения продукта и обзор каталогов. Оба тома *SQL Reference* можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8933.	Том 1 SC09-2974 db2s1x70 Том 2 SC09-2975 db2s2x70	db2s0
System Monitor Guide and Reference	Описывает сбор различной информации о базах данных и менеджере баз данных. Эта книга объясняет, как использовать информацию, чтобы понять работу с базой данных, улучшить производительность и найти причины ошибок.	SC09-2956 db2f0x70	db2f0
Text Extender Administration and Programming	Содержит общую информацию о модулях расширения DB2, о конфигурировании модуля расширения для работы с текстом, об управлении им и о программировании с использованием модулей расширения для работы с текстом. Включает в себя справочную информацию, диагностическую информацию (с сообщениями) и примеры.	SC26-9930 desu9x70	desu9
Troubleshooting Guide	Помогает определить причины ошибок, выполнить восстановительные операции, и использовать средства диагностики, консультируясь со Службой заказчиков DB2.	GC09-2850 db2p0x70	db2p0
Что нового	Описывает новые возможности, функции и усовершенствования в DB2 Universal Database Версии 7.	SC09-2976 db2q0x70	db2q0
Информация по установке и конфигурированию DB2			
DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings	Содержит информацию по планированию, установке и конфигурированию DB2 Connect Enterprise Edition в OS/2 и 32-битных системах Windows. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2953 db2c6x70	db2c6

Таблица 53. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
DB2 Connect Enterprise Edition for UNIX Quick Beginnings	Содержит информацию по планированию, установке, конфигурированию и выполнению заданий для DB2 Connect Enterprise Edition на платформах на основе UNIX. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2952 db2cyx70	db2cy
DB2 Connect Personal Edition Quick Beginnings	Содержит информацию по планированию, установке, конфигурированию и выполнению заданий для DB2 Connect Personal Edition в OS/2 и 32-битных средах Windows. Эта книга содержит также информацию по установке и настройке для всех поддерживаемых клиентов.	GC09-2967 db2c1x70	db2c1
DB2 Connect Personal Edition Quick Beginnings for Linux	Содержит информацию по планированию, установке, перенастройке и конфигурированию DB2 Connect Personal Edition во всех поддерживаемых версиях Linux.	GC09-2962 db2c4x70	db2c4
DB2 Data Links Manager Quick Beginnings	Содержит информацию по планированию, установке и конфигурированию DB2 Data Links Manager в AIX и 32-битных операционных системах Windows.	GC09-2966 db2z6x70	db2z6
DB2 Enterprise - Extended Edition for UNIX Quick Beginnings	Содержит информацию по планированию, установке и конфигурированию DB2 Enterprise - Extended Edition на платформах на основе UNIX. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2964 db2v3x70	db2v3
DB2 Enterprise - Extended Edition for Windows Quick Beginnings	Содержит информацию по планированию, установке и конфигурированию DB2 Enterprise - Extended Edition в 32-битных системах Windows. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2963 db2v6x70	db2v6

Таблица 53. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
DB2 for OS/2 Быстрый старт	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database Personal Edition в операционной системе OS/2. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2968 db2i2x70	db2i2
DB2 for UNIX Быстрый старт	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database на платформах на основе UNIX. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2970 db2ixx70	db2ix
DB2 for Windows Быстрый старт	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database в 32-битных системах Windows. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2971 db2i6x70	db2i6
DB2 Personal Edition Quick Beginnings	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database Personal Edition в OS/2 и в 32-битных системах Windows.	GC09-2969 db2i1x70	db2i1
DB2 Personal Edition Quick Beginnings for Linux	Содержит информацию по планированию, установке, перенастройке и конфигурированию DB2 Universal Database Personal Edition во всех поддерживаемых версиях Linux.	GC09-2972 db2i4x70	db2i4
DB2 Query Patroller Installation Guide	Содержит информацию по установке DB2 Query Patroller.	GC09-2959 db2iwx70	db2iw
DB2 Warehouse Manager Installation Guide	Содержит информацию по установке агентов хранилища, преобразователей хранилища и менеджера каталога данных.	GC26-9998 db2idx70	db2id
Кроссплатформенные программы примеров в формате HTML			

Таблица 53. Информация DB2 (продолжение)

Название	Описание	Номер формы Имя файла PDF	Каталог HTML
Программы примеров в виде HTML	Содержит для справок программы примеров в виде HTML для языков программирования на всех платформах, поддерживаемых DB2. Эти программы примеров приводятся только в информационных целях. Не все из них доступны на всех языках программирования. Примеры HTML доступны, только если установлен клиент разработки программ DB2. Дополнительную информацию об этих программах смотрите в книге *Application Building Guide*.	Номера формы нет	db2hs
	Замечания по выпуску		
DB2 Connect Release Notes	Содержит самую свежую информацию, которую не успели включить в книги по DB2 Connect.	Смотрите примечание 2.	db2cr
DB2 Installation Notes	Содержит самую свежую информацию по установке, которую не успели включить в книги по DB2.	Доступна только на компакт-диске продукта.	
DB2 Release Notes	Содержит самую свежую информацию о всех продуктах DB2 и их возможностях, которую не успели включить в книги по DB2.	Смотрите примечание 2.	db2ir

Примечания:

1. Символ *x* на шестой позиции в имени файла указывает язык книги. Например, имя файла `db2d0e70` говорит о том, что это английская версия книги *Administration Guide*, а имя файла `db2d0f70` соответствует французской версии этой же книги. Для обозначений языков используются на шестой позиции имени файла используются следующие буквы:

Язык	Обозначение
Английский	e
Болгарский	u
Бразильский португальский	b
Венгерский	h
Голландский	q
Греческий	a
Датский	d

Испанский	z
Итальянский	i
Корейский	k
Немецкий	g
Норвежский	n
Польский	p
Португальский	v
Русский	r
Словенский	l
Традиционный китайский	t
Турецкий	m
Упрощенный китайский	c
Финский	y
Французский	f
Чешский	x
Шведский	s
Японский	j

2. Последнюю информацию, которую не успели включить в книги по DB2, смотрите в Замечаниях по выпуску в формате HTML и в виде ASCII-файла. HTML-версию можно вызвать через Информационный центр или с компакт-диска продукта. Чтобы посмотреть ASCII-файл:
   - На платформах на базе UNIX смотрите файл `Release.Notes`. Он расположен в каталоге `DB2DIR/Readme/%L`, где *%L* - национальная версия, а `DB2DIR`:
     - `/usr/lpp/db2_07_01` в AIX
     - `/opt/IBMdb2/V7.1` в HP-UX, PTX, Solaris, и Silicon Graphics IRIX
     - `/usr/IBMdb2/V7.1` в Linux.
   - На других платформах смотрите файл `RELEASE.TXT`. Он находится в каталоге, где установлен продукт. На платформах OS/2 можно также дважды щелкнуть по папке **IBM DB2**, а затем дважды щелкнуть по значку **Release Notes**.

Печать книг PDF

Если вы предпочитаете использовать печатные версии книг, можно напечатать файлы .pdf с компакт-диска публикаций по DB2. При помощи Adobe Acrobat Reader можно напечатать книгу целиком или же определенный диапазон страниц. Имена файлов для каждой книги в библиотеке приводятся в Табл. 53 на стр. 470.

Последнюю версию Adobe Acrobat Reader можно получить с Web-сайта фирмы Adobe, http://www.adobe.com.

Файлы PDF (расширения файлов - .PDF) входят в состав компакт-диска публикаций DB2. Для доступа к этим файлам:

1. Вставьте в устройство CD-ROM компакт-диск с публикациями DB2. На платформах на основе UNIX смонтируйте компакт-диск с публикациями DB2. Процедуру монтирования посмотрите в книге *Быстрый старт*.
2. Запустите Acrobat Reader.
3. Откройте требуемый файл PDF из одного из следующих мест:
   - На платформах OS/2 и Windows:

     Из каталога *x*:\doc*язык*, где *x* - буква компакт-диска, а *язык* двухсимвольный код страны, соответствующий вашему языку (например, RU для русского).
   - На платформах на основе UNIX:

     Из каталога /*cdrom*/doc/*%L* на компакт-диске, где */cdrom* - точка установки компакт-диска, а *%L* - имя требуемой национальной версии.

Можно также скопировать файлы PDF с компакт-диска на локальный или сетевой диск и читать их оттуда.

Заказ печатных копий

Печатные копии книг DB2 можно заказать по отдельности или в комплекте (только в Северной Америке) по номеру SBOF. Чтобы заказать книги, обратитесь к вашему авторизованному дилеру или торговому представителю IBM, или позвоните по телефону 1-800-879-2755 в Соединенных Штатах или 1-800-IBM-4YOU в Канаде. Можно также заказать книги на Web-странице Publications по адресу http://www.elink.ibmlink.ibm.com/pbl/pbl.

Есть два комплекта книг. SBOF-8935 содержит справочную и пользовательскую информацию для DB2 Warehouse Manager. SBOF-8931 содержит справочную и пользовательскую информацию для всех остальных продуктов и возможностей DB2 Universal Database. Содержимое каждого комплекта SBOF приводится в следующей таблице:

Таблица 54. Заказ печатных книг

Номер SBOF	Содержит книги	
SBOF-8931	• Administration Guide: Planning • Administration Guide: Implementation • Administration Guide: Performance • Administrative API Reference • Application Building Guide • Application Development Guide • CLI Guide and Reference • Command Reference • Data Movement Utilities Guide and Reference • Data Warehouse Center Administration Guide • Data Warehouse Center Application Integration Guide • DB2 Connect User's Guide • Installation and Configuration Supplement • Image, Audio, and Video Extenders Administration and Programming • Справочник по сообщениям, Тома 1 и 2	• OLAP Integration Server Administration Guide • OLAP Integration Server Metaoutline User's Guide • OLAP Integration Server Model User's Guide • OLAP Integration Server User's Guide • OLAP Setup and User's Guide • OLAP Spreadsheet Add-in User's Guide for Excel • OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3 • Replication Guide and Reference • Spatial Extender Administration and Programming Guide • SQL Getting Started • SQL Reference, Volumes 1 and 2 • System Monitor Guide and Reference • Text Extender Administration and Programming • Troubleshooting Guide • What's New
SBOF-8935	• Information Catalog Manager Administration Guide • Information Catalog Manager User's Guide • Information Catalog Manager Programming Guide and Reference	• Query Patroller Administration Guide • Query Patroller User's Guide

Электронная документация DB2

Обращение к электронной справке

Для всех компонентов DB2 доступна электронная справка. Различные типы справки перечислены в следующей таблице.

Тип справки	Содержание	Как вызвать...
Справка по командам	Объясняет синтаксис команд процессора командной строки.	В процессоре командной строки в интерактивном режиме введите: `? команда` где *команда* - ключевое слово для команды целиком. Например, `? catalog` выводит справку по всем командам CATALOG, а `? catalog database` выводит справку по команде CATALOG DATABASE.
Справка по Ассистенту конфигурирования клиента *Справка по Командному центру* *Справка по Центр управления* *Справка по Data Warehouse Center* *Справка по анализатору событий* *Справка по менеджеру каталога данных* *Справка по центру управления сателлитами* *Справка по центру сценариев*	Объясняет задания, которые можно выполнить в окне или в записной книжке. Справка содержит обзор и предварительную информацию, которую надо знать, и описывает, как использовать управляющие элементы окна или записной книжки.	В окне или в записной книжке нажмите кнопку **Справка** или клавишу **F1**.

Тип справки	Содержание	Как вызвать...

Справка по сообщениям	Описывает для сообщения причину и действия, которые следует предпринять.	В процессоре командной строки в интерактивном режиме введите:
		`? XXXnnnnn`
		где *XXXnnnnn* - идентификатор допустимого сообщения.
		Например, `? SQL30081` выводит справку по сообщению SQL30081.
		Чтобы смотреть справку по сообщению поэкранно, введите:
		`? XXXnnnnn | more`
		Чтобы записать справку по сообщению в файл, введите:
		`? XXXnnnnn > имяфайла.рсш`
		где *имяфайла.рсш* - имя файла, где вы хотите сохранить справку.

Справка по SQL	Объясняет синтаксис операторов SQL.	В процессоре командной строки в интерактивном режиме введите: `help оператор` где *оператор* - оператор SQL. Например, `help` SELECT выводит справку по оператору SELECT. **Примечание:** Справка по SQL недоступна на платформах на основе UNIX.
Справка по SQLSTATE	Объясняет состояния SQL и коды классов.	В процессоре командной строки в интерактивном режиме введите: `? sqlstate или ? код класса` где *sqlstate* - допустимый пятизначный код SQL, а *код класса* - первые две цифры sqlstate. Например, `? 08003` выводит справку по состоянию SQL 08003, а `? 08` выводит справку по коду класса 08.

Просмотр информации на экране

Книги, поставляемые с этим продуктом, записаны в формате HTML. Этот формат позволяет искать и просматривать информацию и поддерживает гипертекстовые ссылки. Он упрощает также совместное использование библиотеки на сайте.

Электронные книги и примеры программ можно просматривать в любом браузере, который поддерживает спецификации HTML Версии 3.2.

Чтобы просмотреть книги или примеры программ:

- Если вы работаете с инструментами администратора DB2, используйте Информационный центр.
- В браузере выберите **Файл —>Открыть страницу**. На открытой странице приводятся описания и ссылки на информацию по DB2:
  - На платформах на базе UNIX откройте страницу:

    INSTHOME/sqllib/doc/*%L*/html/index.htm

    где *%L* - имя национальной версии.
  - На других платформах откройте страницу:

    sqllib\doc\html\index.htm

    на диске, где установлена DB2.

    Если вы не установили Информационный центр, эту страницу можно открыть, щелкнув дважды по значку **Информация DB2**. В зависимости от того, в какой системе вы работаете, этот значок может находиться в основной папке продукта или в меню Windows Пуск.

Установка браузера Netscape

Если у вас еще не установлен браузер Web, можно установить Netscape с компакт-диска Netscape, включенного в состав продукта. Чтобы получить подробные указания по установке, выполните следующие действия:

1. Вставьте в устройство CD-ROM компакт-диск Netscape.
2. На платформах на основе UNIX смонтируйте компакт-диск. Процедуру монтирования посмотрите в книге *Быстрый старт*.
3. Прочтите инструкции по установке в файле CDNAV*nn*.txt, где *nn* - двухсимвольный идентификатор языка. Этот файл находится в корневом каталоге компакт-диска.

Доступ к информации через Информационный центр

Информационный центр обеспечивает быстрый доступ к информации о продуктах DB2. Информационный центр доступен на всех платформах, где есть инструменты администратора DB2.

Чтобы открыть Информационный центр, щелкните дважды по значку Информационный центр. В зависимости от того, в какой системе вы работаете, этот значок может находиться в основной папке продукта или в меню **Пуск**.

На платформах Windows можно также вызвать Информационный центр через панель задач и через меню **Справка** DB2.

Информационный центр дает шесть типов информации. Для обращения к информации одного из этих типов выберите соответствующую закладку.

Задания Основные задания, которые вы можете выполнить в DB2.

Справочник Справочная информацию по таким элементам DB2, как ключевые слова, команды и API.

Книги Книги DB2.

Устранение неисправностей
Список сообщений об ошибках и рекомендуемых действий по категориям.

Программы примеров
Программы примеров, поставляемые с клиентом разработки программ DB2. Если вы не установили клиент разработки программ DB2, эта закладка не выводится.

Web Информация по DB2 в WWW. Чтобы посмотреть эту информацию, ваша система должна быть подключена к Web.

Когда вы выбираете пункт в одном из списков, информационный центр запускает программу просмотра для вывода информации. Этой программой может быть программа просмотра системной справки, редактор или браузер Web в зависимости от того, какую информацию вы выбрали.

Информационный центр поддерживает возможность поиска, и вы можете искать определенную тему, не просматривая книги целиком.

Для полнотекстового поиска выберите гипертекстовую ссылку в Информационном центре и откройте поисковую форму **Поиск электронной информации DB2**.

Обычно сервер поиска HTML запускается автоматически. Если поиск информации HTML не работает, вам, возможно надо запустить сервер поиска одним из следующих способов:

В Windows
Выберите **Пуск**, затем **Программы—> IBM DB2 —> Информация —> Запустить сервер поиска HTML**.

В OS/2 Щелкните дважды по папке **DB2 for OS/2**, а затем щелкните дважды по значку **Запустить сервер поиска HTML**.

Если у вас есть проблемы с использованием поиска информации HTML, посмотрите замечания по выпуску.

Примечание: Функция поиска недоступна в средах Linux, PTX и Silicon Graphics IRIX.

Использование мастеров DB2

Мастера помогают вам выполнять конкретные задачи управления, ведя последовательно по шагам необходимых действий. Мастера доступны в Центре управления и в Ассистенте конфигурирования клиента. Список мастеров с соответствующими задачами приведен в следующей таблице.

Примечание: Мастера по созданию баз данных, индексов, конфигурированию многоузлового изменения и производительности доступны в среде многораздельных баз данных.

Мастер	Помогает вам...	Как вызвать...
по добавлению баз данных	Каталогизировать базу данных на клиентской рабочей станции	В Ассистенте конфигурирования клиента нажмите кнопку **Добавить**.
по резервному копированию базы данных	Определить, создать и включить в расписание резервное копирование.	В Центре управления щелкните правой кнопкой мыши по базе данных, для которой вам нужна резервная копия, и выберите **Резервное копирование —> Базы данных при помощи мастера**.
по конфигурированию многоузлового изменения	Конфигурировать многоузловые изменения, распределенные транзакции или двухфазное принятие.	В Центре управления щелкните правой кнопкой мыши по папке **Базы данных** и выберите **Многоузловое изменение**.
по созданию баз данных	Создать базу данных и выполнить основные задачи конфигурирования.	В Центре управления щелкните правой кнопкой мыши по папке **Базы данных** и выберите **Создать —> Базу данных при помощи мастера**.
по созданию таблиц	Выбрать типы основных данных и создать первичные ключи для таблицы.	В Центре управления щелкните правой кнопкой мыши по значку **Таблицы** и выберите **Создать —> Таблицу при помощи мастера**.
по созданию табличных пространств	Создать новое табличное пространство.	В Центре управления щелкните правой кнопкой мыши по значку **Табличные пространства** и выберите **Создать —> Табличное пространство при помощи мастера**.
Создать индекс	Выбрать, какие индексы создать или отбросить для всех ваших запросов.	В Центре управления щелкните правой кнопкой мыши по значку **Индекс** и выберите **Создать —> Индекс при помощи мастера**.

Мастер	Помогает вам...	Как вызвать...
по настройке производительности	Настроить производительность базы данных, изменив параметры конфигурации в соответствии с вашими требованиями.	В Центре управления щелкните правой кнопкой мыши по базе данных, которую вы хотите настроить, и выберите **Конфигурировать производительность при помощи мастера**. Для многораздельной среды баз данных в окне Разделы баз данных щелкните правой кнопкой мыши по первому разделу баз данных, который вы хотите настроить, и выберите **Конфигурировать производительность при помощи мастера**.
по восстановлению баз данных	Восстановить базу данных после сбоя. Он поможет понять, какую резервную копию использовать, и какие журналы использовать при повторе.	В Центре управления щелкните правой кнопкой мыши по базе данных, которую вы хотите восстановить, и выберите **Восстановить —> Базу данных при помощи мастера**.

Установка сервера документации

По умолчанию информация по DB2 устанавливается в вашей локальной системе. Это значит, что каждый, кому требуется доступ к информации по DB2, должен устанавливать одни и те же файлы. Чтобы держать информацию по DB2 в едином месте, выполните следующие действия:

1. Скопируйте все файлы и подкаталоги каталога `\sqllib\doc\html` вашей локальной системы на сервер Web. Каждая книга находится в своем собственном подкаталоге, где записаны все необходимые для нее файлы HTML и GIF. Структура подкаталогов должна остаться без изменений.
2. Сконфигурируйте сервер Web на поиск файлов на новом месте. Дополнительную информацию смотрите в приложении NetQuestion руководства *Дополнение по установке и конфигурированию*.
3. Если вы используете Java-версию Информационного центра, можно задать базовый URL для всех файлов HTML. Этот URL надо использовать для списка книг.
4. Когда вы сможете просматривать файлы книг, можно пометить закладками часто используемые темы. Вероятно, вы захотите пометить закладками следующие страницы:
   - Список книг
   - Содержания часто используемых книг

- Часто требуемые статьи, например, тему ALTER TABLE
- Форму поиска

Информацию о том, как работать с файлами электронной документации на центральном компьютере, смотрите в приложении NetQuestion руководства *Дополнение по установке и конфигурированию*.

Поиск электронной информации

Для поиска информации в файлах HTML используйте один из следующих способов:

- Нажмите кнопку **Поиск** в верхнем фрейме. При помощи формы поиска найдите нужную тему. Эта функция недоступна в средах Linux, PTX и Silicon Graphics IRIX.
- Нажмите кнопку **Индекс** в верхнем фрейме. При помощи индекса найдите в книге нужную тему.
- Выведите содержание или индекс справки или книги HTML, затем при помощи функции поиска браузера Web найдите в книге нужную тему.
- При помощи функции закладок браузера Web можно быстро вернуться к определенной теме.
- Используйте для поиска определенных тем функцию поиска информационного центра. Подробности смотрите в разделе “Доступ к информации через Информационный центр” на стр. 484.

Приложение M. Замечания

IBM может предлагать описанные продукты, услуги и возможности не во всех странах. Сведения о продуктах и услугах, доступных в настоящее время в вашей стране, можно получить в местном представительстве IBM. Любые ссылки на продукты, программы или услуги IBM не означают явным или неявным образом, что можно использовать только продукты, программы или услуги IBM. Разрешается использовать любые функционально эквивалентные продукты, программы или услуги, если при этом не нарушаются права IBM на интеллектуальную собственность. Однако ответственность за оценку и проверку работы любых продуктов, программ и услуг других фирм лежит на пользователе.

Фирма IBM может располагать патентами или рассматриваемыми заявками на патенты, относящимися к предмету данного документа. Получение этого документа не означает предоставления каких-либо лицензий на эти патенты. Запросы по поводу лицензий следует направлять в письменной форме по адресу:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

По поводу лицензий, связанных с использованием наборов двухбайтных символов (DBCS), обращайтесь в отдел интеллектуальной собственности IBM в вашей стране или направьте запрос в письменной форме по адресу:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

Следующий абзац не применяется в Великобритании или в любой другой стране, где подобные заявления противоречат местным законам: КОРПОРАЦИЯ INTERNATIONAL BUSINESS MACHINES ПРЕДСТАВЛЯЕТ ДАННУЮ ПУБЛИКАЦИЮ “КАК ЕСТЬ” БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ПРЕДПОЛАГАЕМЫЕ ГАРАНТИИ СОВМЕСТИМОСТИ, РЫНОЧНОЙ ПРИГОДНОСТИ И СООТВЕТСТВИЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. В некоторых странах для определенных сделок подобные оговорки не допускаются, таким образом, это утверждение может не относиться к вам.

Данная информация может содержать технические неточности и типографские опечатки. Периодически в информацию вносятся изменения, они будут включены в новые издания этой публикации. Фирма IBM может в любое время без уведомления вносить изменения и усовершенствования в продукты и программы, описанные в этой публикации.

Любые ссылки в данной информации на Web-сайты, не принадлежащие IBM, приводятся только для удобства и никоим образом не означают поддержки IBM этих Web-сайтов. Материалы этих Web-сайтов не являются частью данного продукта IBM и вы можете использовать их только на собственную ответственность.

IBM может использовать или распространять присланную вами информацию любым способом, как фирма сочтет нужным, без каких-либо обязательств перед вами.

Если обладателю лицензии на данную программу понадобятся сведения о возможности: (i) обмена данными между независимо разработанными программами и другими программами (включая данную) и (ii) совместного использования таких данных, он может обратиться по адресу:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

Такая информация может быть предоставлена на определенных условиях (в некоторых случаях к таким условиям может относиться оплата).

Лицензированная программа, описанная в данной публикации, и все лицензированные материалы, доступные с ней, предоставляются IBM на условиях IBM Customer Agreement (Соглашения IBM с заказчиком), Международного соглашения о лицензиях на программы IBM или эквивалентного соглашения.

Приведенные данные о производительности измерены в контролируемой среде. Таким образом, результаты, полученные в других операционных средах, могут существенно отличаться от них. Некоторые показатели измерены в системах разработки и нет никаких гарантий, что в общедоступных системах эти показатели будут теми же. Более того, некоторые результаты могут быть получены путем экстраполяции. Реальные результаты могут отличаться от них. Пользователи должны проверить данные для своих конкретных сред.

Информация о продуктах других фирм получена от поставщиков этих продуктов, из их опубликованных объявлений или из других общедоступных

источников. Фирма IBM не проверяла эти продукты и не может подтвердить точность измерений, совместимость или прочие утверждения о продуктах других фирм. Вопросы о возможностях продуктов других фирм следует направлять поставщикам этих продуктов.

Все утверждения о будущих планах и намерениях IBM могут быть изменены или отменены без уведомлений, и описывают исключительно цели фирмы.

Эта информация может содержать примеры данных и отчетов, иллюстрирующие типичные деловые операции. Чтобы эти примеры были правдоподобны, в них включены имена лиц, названия компаний и товаров. Все эти имена и названия вымышлены и любое их сходство с реальными именами и адресами полностью случайно.

ЛИЦЕНЗИЯ НА КОПИРОВАНИЕ:

Эта информация может содержать примеры прикладных программ на языках программирования, иллюстрирующих приемы программирования для различных операционных платформ. Разрешается копировать, изменять и распространять эти примеры программ в любой форме без оплаты фирме IBM для целей разработки, использования, сбыта или распространения прикладных программ, соответствующих интерфейсу прикладного программирования операционных платформ, для которых эти примера программ написаны. Эти примеры не были всесторонне проверены во всех возможных условиях. Поэтому IBM не может гарантировать их надежность, пригодность и функционирование.

Каждая копия программ примеров или программ, созданных на их основе, должна содержать следующее замечание об авторских правах:

© (название вашей фирмы) (год). Части этого кода построены на основе примеров программ IBM Corp. © Copyright IBM Corp. _введите год или годы_. Все права защищены.

Товарные знаки

Следующие термины (они могут быть помечены звездочкой - *) являются товарными знаками корпорации International Business Machines в Соединенных Штатах и/или в других странах:

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

Следующие термины являются товарными знаками или зарегистрированными товарными знаками других компаний:

Microsoft, Windows и Windows NT - товарные знаки или зарегистрированные товарные знаки Microsoft Corporation.

Java, все товарные знаки и логотипы на основе Java и Solaris - товарные знаки Sun Microsystems, Inc. в Соединенных Штатах и/или в других странах.

Tivoli и NetView - товарные знаки Tivoli Systems Inc. в Соединенных Штатах и/или других странах.

UNIX - зарегистрированный товарный знак в Соединенных Штатах и в других странах, его использование лицензируется исключительно фирмой X/Open Company Limited.

Названия других компаний, продуктов и услуг (они могут быть отмечены двойной звездочкой - **) могут быть товарными знаками или марками сервиса других фирм.

Индекс

Спец. символы

A

C

D

F

G

H

I

K

L

N

P

К

Л

М

H

O

П

P

C

T

У

Ф

Х

Ц

Ш

Э

Я

Как связаться с IBM

Если у вас имеется техническая проблема, пожалуйста, перед обращением к службе поддержки пользователей DB2 просмотрите еще раз и выполните действия, рекомендуемые в руководстве *Troubleshooting Guide*. В этом руководстве описано, какую информацию надо собрать, чтобы служба поддержки пользователей DB2 могла лучше помочь вам.

Чтобы получить информацию или заказать любой из продуктов DB2 Universal Database, обратитесь к представителю IBM в местном отделении или к авторизованному продавцу программных продуктов IBM.

Если вы находитесь в США, позвоните по одному из следующих номеров:

- 1-800-237-5511, чтобы обратиться в службу поддержки
- 1-888-426-4343, чтобы узнать о доступных формах обслуживания.

Информация о продукте

Если вы находитесь в США, позвоните по одному из следующих номеров:

- 1-800-IBM-CALL (1-800-426-2255) или 1-800-3IBM-OS2 (1-800-342-6672), чтобы заказать продукты или получить общую информацию.
- 1-800-879-2755, чтобы заказать публикации.

http://www.ibm.com/software/data/
На страницах DB2 в WWW содержится текущая информация DB2: новости, описания продуктов, учебные планы и т.д.

http://www.ibm.com/software/data/db2/library/
DB2 Product and Service Technical Library содержит ответы на часто задаваемые вопросы, исправления, книги и свежую техническую информацию по DB2.

Примечание: Эта информация может быть только в английском варианте.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
На сайте заказов International Publications приводится информация о том, как заказывать книги.

http://www.ibm.com/education/certify/
На этом сайте представлена программа Professional Certification Program IBM и приводится информация о сертификационных испытаниях для многих продуктов IBM, в том числе DB2.

ftp.software.ibm.com

Зарегистрируйтесь как аноним. В каталоге `/ps/products/db2` можно найти демо-версии, исправления, информацию и инструменты для DB2 и многих других продуктов.

comp.databases.ibm-db2, bit.listserv.db2-l

В этих группах новостей пользователи обмениваются опытом работы с продуктами DB2.

В Compuserve: GO IBMDB2

Введите эту команду, чтобы попасть на форумы IBM DB2 Family. Через эти форумы поддерживаются все продукты DB2.

Информацию о том, как связаться с IBM из других стран, смотрите в Приложении A книги *IBM Software Support Handbook*. Этот документ можно найти в Web, обратившись по адресу: http://www.ibm.com/support/ и выбрав ссылку на IBM Software Support Handbook у нижнего края страницы.

Примечание: В некоторых странах авторизованные дилеры IBM должны обращаться не в центр поддержки IBM, а в структуры поддержки дилеров.

Напечатано в Дании

SH43-0144-01