IBM® DB2 Universal Database™

Руководство администратора: Планирование

Версия 8

SH43-0200-00

IBM® DB2 Universal Database™

Руководство администратора: Планирование

Версия 8

SH43-0200-00

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

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

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

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

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

Содержание

Об этой книге

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

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

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

- **Процессор командной строки**, позволяющий обращаться к базам данных и работать с ними через графический интерфейс. При помощи этого интерфейса можно также выполнять операторы SQL и утилиты DB2. Большинство примеров в этой книге иллюстрируют использование данного интерфейса. Дополнительную информацию об использовании процессора командной строки смотрите в руководстве *Command Reference*.
- **Интерфейс прикладного программирования**, позволяющий вызывать утилиты DB2 из прикладной программы. Дополнительную информацию об использовании интерфейса прикладного программирования смотрите в книге *Administrative API Reference*.
- **Центр управления**, предоставляющий графический пользовательский интерфейс для выполнения задач управления, например, конфигурирования системы, управления каталогами, резервного копирования и восстановления системы, составления расписаний заданий и управления носителями. Центр управления позволяет выполнять также Управление репликацией для графического задания репликации данных между системами. Кроме того, Центр управления позволяет выполнять утилиты DB2 при помощи графического пользовательского интерфейса. В зависимости от вашей платформы есть различные методы вызова Центра управления. Например, можно ввести команду `db2cc` в командной строке, выбрать значок Центра управления в папке DB2 или использовать меню Пуск на платформах Windows. Начальные справочные сведения 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.
- Центр работоспособности для устранения неполадок производительности и размещения ресурсов.
- Параметры инструментов для изменения параметров Центра управления, Центра предупреждений и Репликации.
- Журнал для планирования автоматического запуска заданий.
- Центр хранилищ данных для управления объектами хранилищ.

Для кого предназначена эта книга

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

Структура книги

Эта книга содержит сведения на следующие основные темы:

Концепции баз данных

- В разделе Глава 1, “Основные принципы реляционных баз данных” приводится обзор объектов и понятий баз данных.
- В разделе Глава 2, “Параллельные системы баз данных” излагаются начальные сведения о типах параллелизма, доступных при работе с DB2.
- В разделе Глава 3, “О хранилищах данных” дается обзор работы с хранилищами данных и возникающих при этом задач.

Проектирование баз данных

- В разделе Глава 4, “Логическая структура базы данных” обсуждаются концепции логической структуры базы данных и даются указания по разработке баз данных.
- В разделе Глава 5, “Физическая структура базы данных” содержатся указания по разработке физической структуры базы данных, включая структуру табличных пространств и требования к памяти.

- В разделе Глава 6, “Проектирование распределенных баз данных” обсуждается, как обращаться к нескольким базам данных в ходе одной транзакции.
- В разделе Глава 7, “Проектирование базы данных для применения XA-совместимых менеджеров транзакций” обсуждается, как использовать ваши базы данных в среде обработки распределенных транзакций.

Приложения

- Приложение A, “Несовместимости выпусков” содержит информацию о несоответствиях между версиями 7 и 8, а также возможных несоответствиях в будущем, которые необходимо учитывать.
- В разделе Приложение B, “Поддержка национальных языков (NLS)” дается понятие о поддержке национальных языков в DB2, включая информацию о странах, языках и кодовых страницах.

Краткий обзор других томов Руководства администратора

Руководство администратора: Реализация

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

Реализация вашего проекта

- В разделе ″Перед созданием базы данных″ описываются предварительные требования для создания базы данных.
- В разделе ″Создание базы данных″ рассматриваются задачи, связанные с созданием базы данных и связанных с ней объектов базы данных.
- В разделе ″Изменение базы данных″ обсуждается, что необходимо сделать перед изменением базы данных и задачи, связанные с модификацией или отбрасыванием базы данных или связанных с ней объектов базы данных.

Защита базы данных

- В разделе ″Управление доступом к базе данных″ описывается, как управлять доступом к ресурсам вашей базы данных.
- В разделе ″Аудит активности DB2″ описывается, как обнаруживать и отслеживать нежелательные или непредвиденные попытки обращения к данным.

Приложения

- В разделе ″Правила именования″ приводятся правила именования баз данных и объектов.
- В разделе ″Использование служб каталога протокола LDAP″ приводится информация о том, как пользоваться службами каталога LDAP.

- В разделе ″Выдача команд нескольким разделам баз данных″ обсуждается использование сценариев оболочки *db2_all* и *rah* для отправки команд всем разделам в среде многораздельных баз данных.
- В разделе ″Поддержка Windows Management Instrumentation (WMI)″ описана поддержка DB2 указанного стандарта инфраструктуры управления для объединения различных систем управления аппаратным и программным обеспечением. Также обсуждается взаимодействие DB2 и WMI.
- В разделе ″Как DB2 for Windows NT работает с защитой Windows NT″ описывается, как DB2 работает с защитой Windows NT.
- В книге ″Использование монитора производительности Windows″ приводится информация о регистрации DB2 с монитором производительности Windows NT и об использовании сведений о производительности.
- В разделе ″Работа с серверами разделов баз данных Windows″ приводится информация об имеющихся утилитах для работы с серверами разделов баз данных Windows NT или Windows 2000.
- В разделе ″Конфигурирование нескольких логических узлов″ описывается, как конфигурировать несколько логических узлов в среде многораздельной базы данных.
- В разделе ″Расширение Центра управления″ приводится информация о том, как расширить Центр управления, добавив новые кнопки на панель инструментов, новые действия, новые определения объектов и действий.

Примечание: Из этой книги были удалены две главы.

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

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

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

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

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

Руководство администратора: производительность

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

Производительность -- Введение

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

Настройка производительности прикладных программ

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

Настройка и конфигурирование вашей системы

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

- В разделе ″Измерение производительности″ представлен обзор вопросов и способов измерения производительности.
- В разделе ″Настройка DB2″ обсуждаются файлы конфигурации менеджера баз данных и базы данных и значения параметров конфигурации DAS.

Приложения

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

Часть 1. Концепции баз данных

Глава 1. Основные принципы реляционных баз данных

Объекты базы данных

Экземпляры:

Экземпляр (другое название - *менеджер баз данных*) - это программа DB2®, предназначенная для управления данными. Он управляет возможными действиями над данными и назначенными для него системными ресурсами. Каждый экземпляр является полнофункциональной средой. Он содержит все разделы базы данных, определенные для данной параллельной системы баз данных. У экземпляра есть собственные базы данных (к ним не могут обращаться другие экземпляры), а все его разделы баз данных совместно используют одни и те же системные каталоги. Кроме того, у экземпляра есть своя защита, отдельная от остальных экземпляров, установленных на этом же компьютере (или в системе).

Базы данных:

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

Группы разделов базы данных:

Группа разделов базы данных - это набор из одного или нескольких разделов базы данных. При создании таблиц для базы данных сначала создается группа разделов, в которой будут храниться табличные пространства, а затем создается табличное пространство, в котором будут храниться таблицы.

В предыдущих версиях DB2 *группы разделов базы данных* назывались *группами узлов*.

Таблицы:

В реляционной базе данных данные представлены как собрание таблиц. *Таблица* состоит из данных, логически упорядоченных по столбцам и строкам. Все данные базы данных и таблиц собраны в табличных пространствах. Данные в

таблице логически связаны, а между таблицами могут быть заданы отношения. Данные можно просматривать и работать с ними на основе математических правил и операций.

Для доступа к табличным данным применяется Язык структурированных запросов (SQL) - стандартный язык для определения данных и работы с ними в реляционных базах данных. *Запрос* применяется в программах или пользователями для получения данных из базы данных. Запрос использует SQL для создания оператора следующей формы:

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

Производные таблицы:

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

Производная таблица может включать все или некоторые столбцы или строки из таблиц, на которых она определена. Например, в производной таблице можно объединить таблицу отделов и таблицу сотрудников, чтобы было можно вывести список всех сотрудников по заданному отделу.

На рис. 1 на стр. 5 показаны отношения между таблицами и производными таблицами.

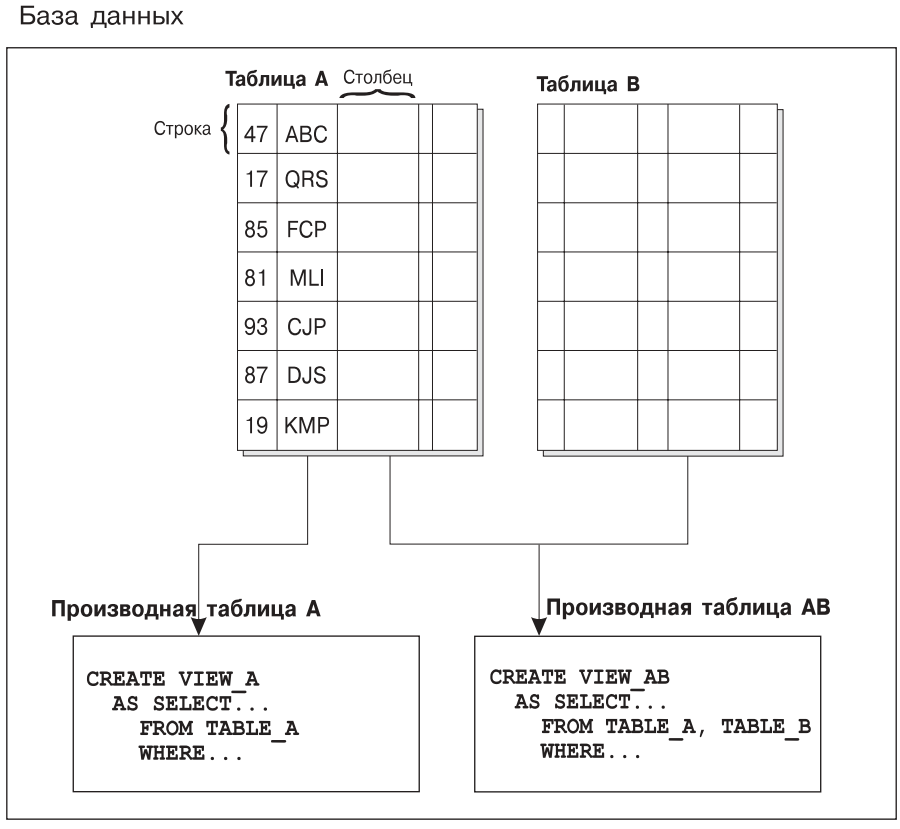

Рисунок 1. Отношение между таблицами и производными таблицами

Индексы:

Индекс - это набор ключей, каждый из которых указывает на строки в таблице. Например, таблица A на рис. 2 на стр. 6 имеет индекс по номерам сотрудников этой таблицы. По значению ключа можно получить указатель на строки в таблице: так, по номеру сотрудника 19 можно найти сотрудника KMP. Индекс позволяет эффективнее получать доступ к строкам в таблице, создавая прямой путь к данным через указатели.

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

Для гарантии уникальности индексного ключа могут быть созданы индексы уникальности. *Индексный ключ* - это столбец или упорядоченное собрание

столбцов, на которых определяется индекс. Использование индекса уникальности гарантирует, что значение каждого индексного ключа в индексном столбце или столбцах уникально.

На рис. 2 показано отношение между индексом и таблицей.

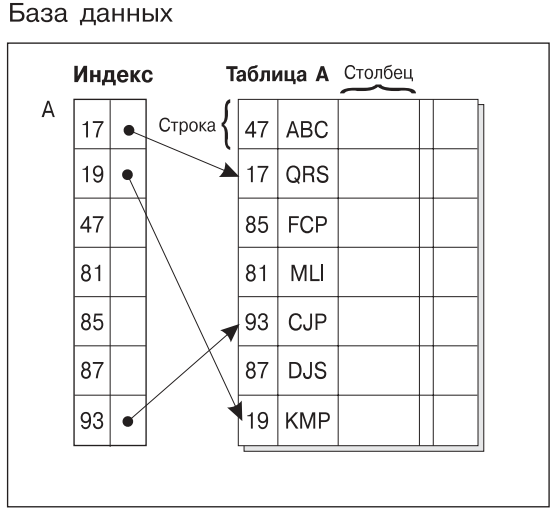

Рисунок 2. Отношение между индексом и таблицей

На рис. 3 на стр. 7 показана взаимосвязь между различными объектами базы данных. Здесь также показано, что таблицы, индексы и длинные данные хранятся в табличных пространствах.

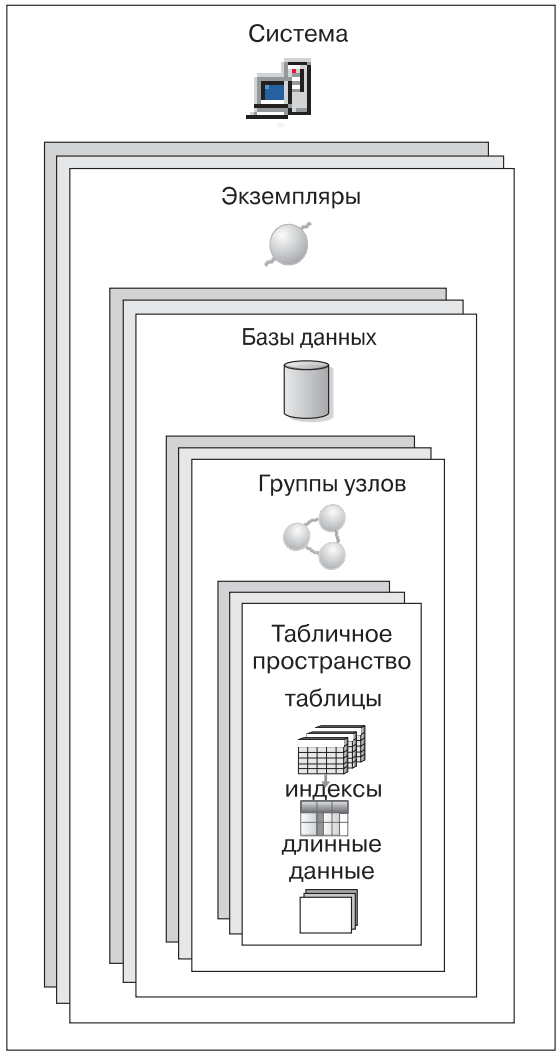

Рисунок 3. Отношения между некоторыми объектами баз данных

Схемы:

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

Схема также является объектом баз данных. Она может быть создана автоматически, когда для нее создается первый объект. Этим объектом может быть любой объект, который можно специфицировать по имени схемы,

например, таблица, индекс, производная таблица, пакет, особый тип, функция или триггер. Чтобы схема создавалась автоматически, у вас должны быть полномочия IMPLICIT_SCHEMA; можно также создать схему явно.

Имя схемы используется как первая часть двухчастного имени объекта. При создании объекта его можно назначить в заданную схему. Если не задать схему, объект назначается в схему по умолчанию; обычно это схема ID пользователя, создавшего данный объект. Вторая часть такого имени - это собственно имя объекта. Например, у пользователя по имени Smith может быть таблица с именем SMITH.PAYROLL.

Таблицы системного каталога:

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

Табличные пространства:

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

Рисунок 4. Табличные пространства

Табличные пространства расположены в группе разделов базы данных. Определения и атрибуты табличных пространств хранятся в системном каталоге баз данных.

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

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

На рис. 5 показано отношение между таблицами, табличными пространствами и этими двумя типами пространств. Здесь также показано, что таблицы, индексы и длинные данные хранятся в табличных пространствах.

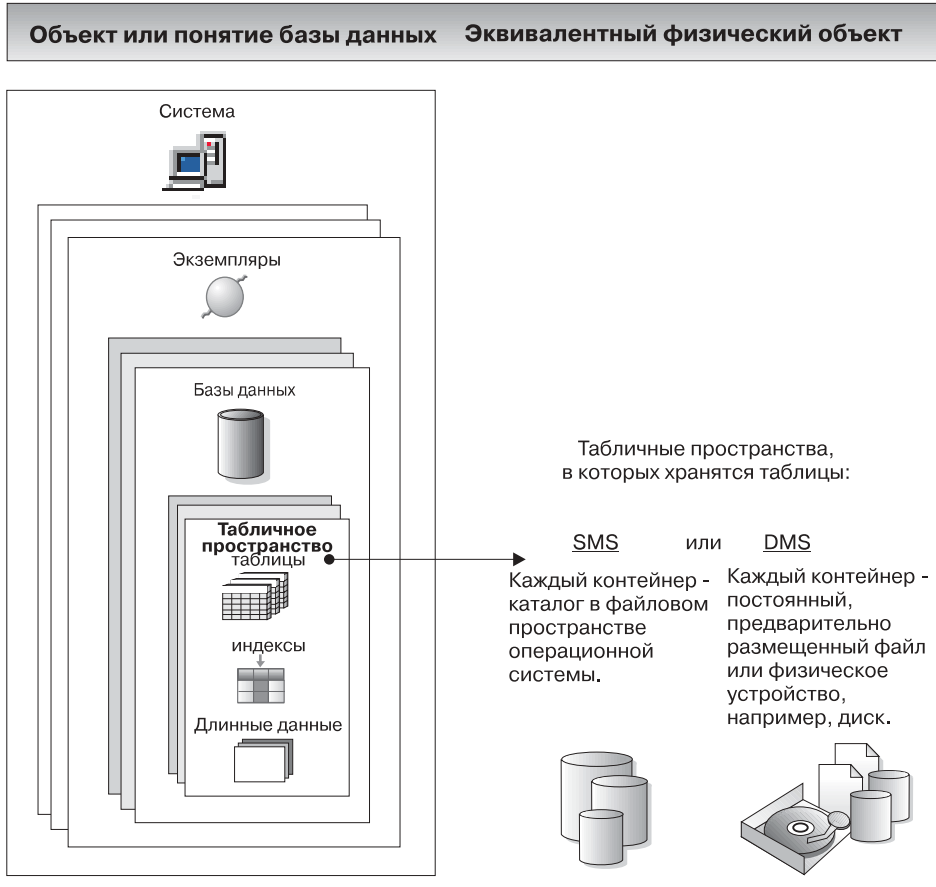

Рисунок 5. Табличные пространства и таблицы

На рис. 6 на стр. 12 показаны табличные пространства трех типов: *обычное*, *временное* и *большое*.

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

Таблицы, содержащие длинные поля данных или данные больших объектов, например, объектов мультимедиа, размещаются в больших табличных пространствах или обычных табличных пространствах. Основные данные из

таких столбцов хранятся в обычном табличном пространстве, а данные длинных полей и больших объектов хранятся либо в том же обычном табличном пространстве, либо в указанном большом табличном пространстве.

Индексы хранятся в обычных табличных пространствах или больших табличных пространствах.

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

Рисунок 6. Три типа табличных пространств

Контейнеры:

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

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

На рис. 7 на стр. 13 показано отношение между таблицами и табличным пространством в базе данных, а также приведены связанные с ними контейнеры и диски.

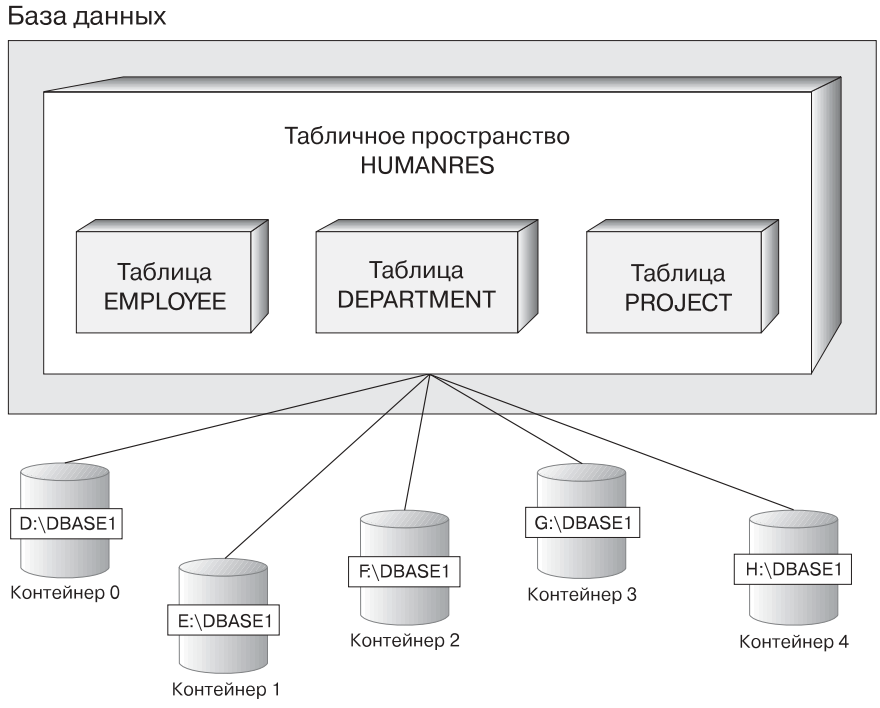

Рисунок 7. Табличные пространства и таблицы в базе данных

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

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

Пулы буферов:

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

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

На рис. 8 показана взаимосвязь между пулом буферов и контейнерами.

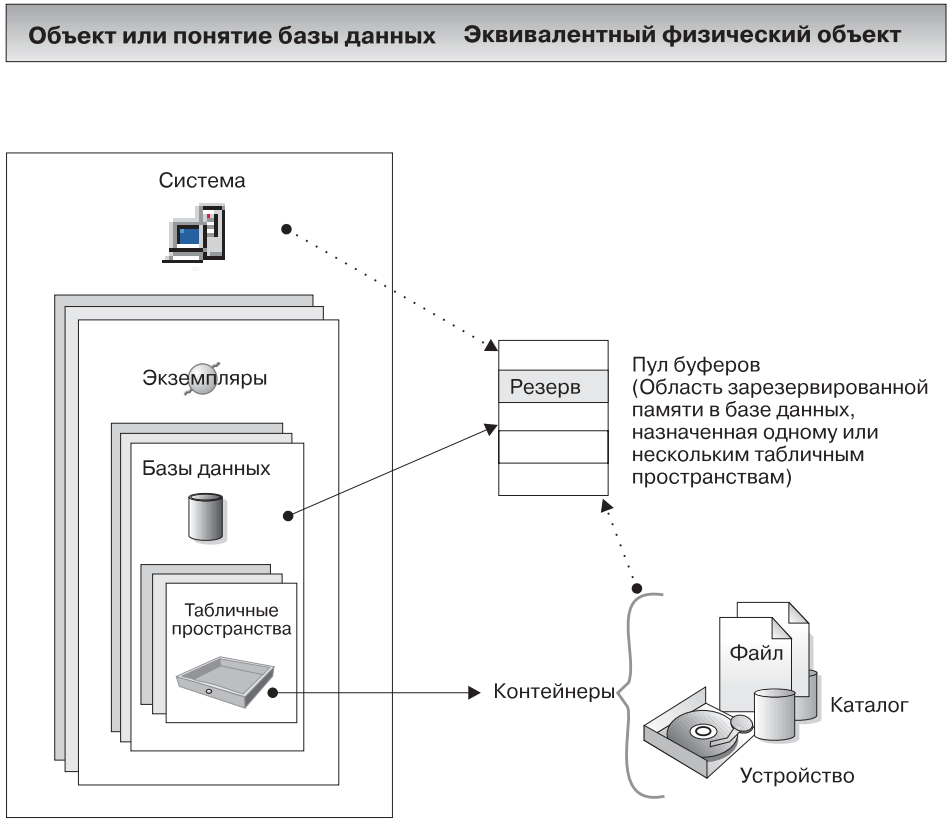

Рисунок 8. Пул буферов и контейнеры

Понятия, связанные с данным:

- “Indexes” в книге *SQL Reference, Том 1*
- “Tables” в книге *SQL Reference, Том 1*
- “Relational databases” в книге *SQL Reference, Том 1*
- “Schemas” в книге *SQL Reference, Том 1*
- “Views” в книге *SQL Reference, Том 1*
- “Table spaces and other storage structures” в книге *SQL Reference, Том 1*

Параметры конфигурации

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

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

- файл конфигурации менеджера баз данных для каждого экземпляра DB2
- файл конфигурации базы данных для каждой отдельной базы данных.

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

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

Параметры конфигурации менеджера баз данных хранятся в файле с именем `db2systm`. Этот файл создается при создании экземпляра менеджера баз данных. В средах на основе UNIX этот файл находится в подкаталоге `sqllib` каталога экземпляра менеджера баз данных. В операционной системе Windows этот файл по умолчанию создается в подкаталоге экземпляра каталога `sqllib`. Если задана переменная DB2INSTPROF, этот файл находится в подкаталоге `instance` каталога, заданного переменной DB2INSTPROF.

В среде многораздельных баз данных этот файл находится в совместно используемой файловой системе, так что все серверы разделов базы данных имеют доступ к одному и тому же файлу. Конфигурация менеджера баз данных на всех серверах разделов базы данных одна и та же.

Большинство параметров либо влияет на объем системных ресурсов, который будет отводиться одному экземпляру менеджера баз данных, либо задают настройку менеджера баз данных и разных подсистем связи, учитывающую особенности среды. Кроме того, существуют параметры, которые служат чисто информационным целям и не допускают изменения. Все эти параметры имеют глобальную применимость, независимо от всех отдельных баз данных, хранящихся в этом экземпляре менеджера баз данных.

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

Параметры отдельной базы данных хранятся в файле конфигурации с именем SQLDBCON. Этот файл хранится вместе с другими файлами управления базы данных в каталоге SQLnnnnn, где nnnnn - это номер, присваиваемый при создании базы данных. У каждой базы данных есть собственный файл конфигурации, и большинство параметров из этого файла задают объем ресурсов, выделяемых базе данных. Кроме того, файл содержит описательную информацию, а также флаги состояния базы данных.

В многораздельной среде базы данных для каждого раздела базы данных создается отдельный файл SQLDBCON. Значения в файлах SQLDBCON разделов базы данных могут различаться, но рекомендуется применять во всех разделах одинаковые значения параметров конфигурации базы данных.

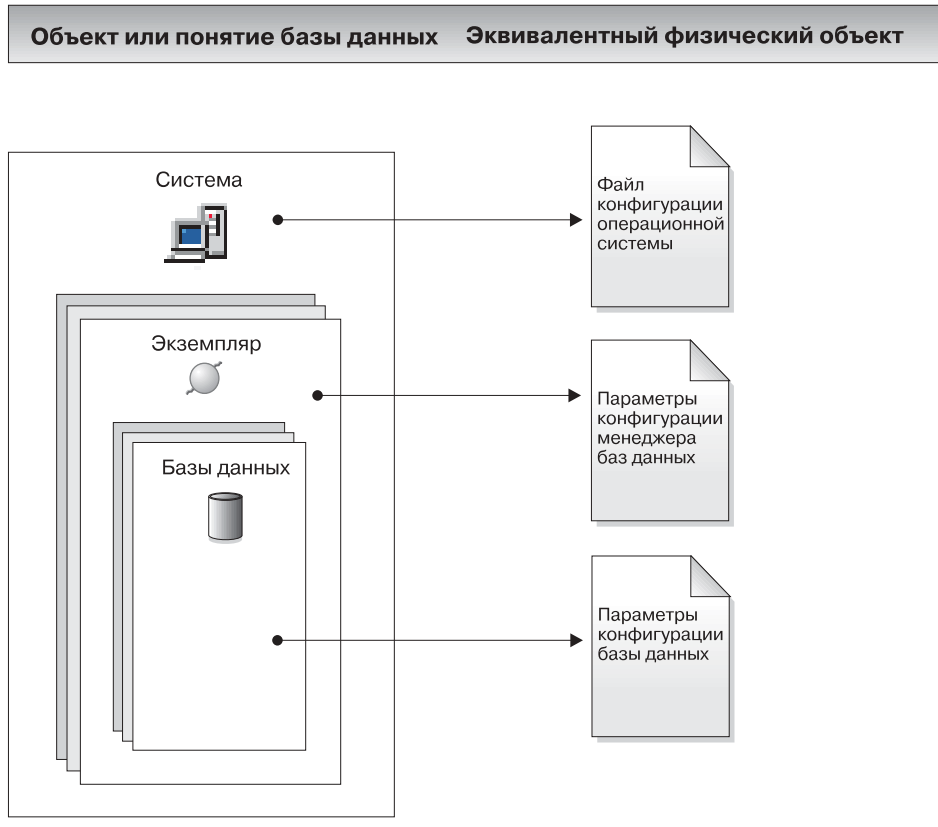

Рисунок 9. Файлы параметров конфигурации

Понятия, связанные с данным:

- “Настройка параметров конфигурации” в книге *Руководство администратора: Производительность*

Задачи, связанные с данной темой:

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

Логические правила для данных

В любой сфере деятельности данные чаще всего должны подчиняться определенным ограничениям или правилам. Например, номера сотрудников должны быть уникальны. В DB2® для задания таких правил применяются *ограничения*.

DB2 поддерживает следующие типы ограничений:

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

ограничение NOT NULL

Ограничение NOT NULL запрещает вставлять в столбец пустые значения.

ограничение уникальности

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

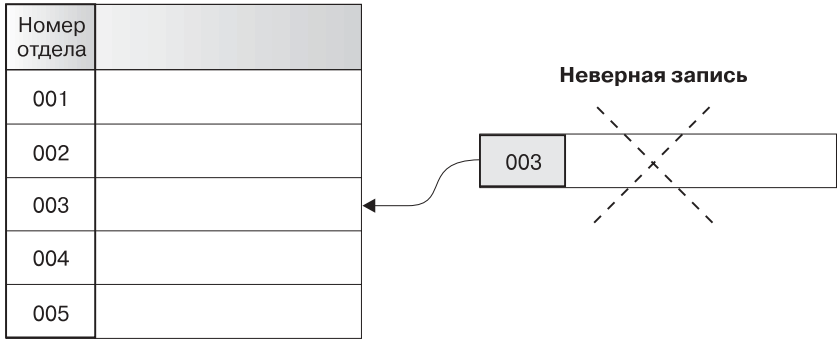

Рисунок 10. Ограничения уникальности предотвращают повторение данных

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

ограничение первичного ключа

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

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

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

Таблица 1. Таблица DEPARTMENT

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

Таблица 2. Таблица EMPLOYEE

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

ограничение внешнего ключа

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

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

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

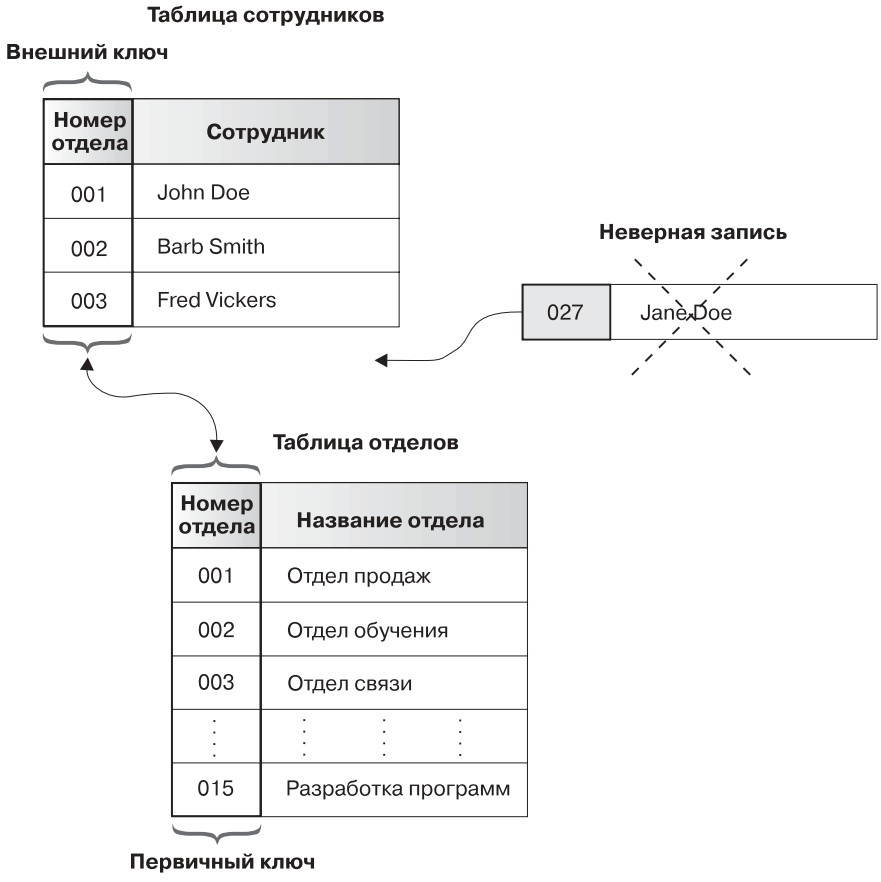

Рисунок 11. Ограничения внешнего и первичного ключей определяют отношения и защищают данные

проверочное ограничение

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

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

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

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

Понятия, связанные с данным:

- “Ограничения” на стр. 86
- “Триггеры” на стр. 91

Разработка стратегии резервного копирования и восстановления

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

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

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

Идея *резервного копирования* базы данных - та же, что и для любых других данных: копия данных снимается и затем сохраняется на другом носителе на случай ошибки или повреждения оригинала. В простейшем случае резервного копирования работу с базой данных прекращают для гарантии, чтобы при копировании не выполнялись последующие транзакции, а затем просто создают

ее резервную копию. Если затем база данных будет каким-либо образом повреждена или испорчена, вы сможете ее воссоздать.

Воссоздание базы данных называется *восстановлением*. *Восстановление версии* - это восстановление предыдущей версии базы данных с использованием образа, созданного при резервном копировании. *Восстановление повтором* - это повторное применение транзакций, записанных в файлах журналов базы данных, после того, как база данных или табличное пространство восстановлены из образа резервной копии.

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

Файлы журналов восстановления и файл хронологии восстановления создаются автоматически при создании базы данных (рис. 12 на стр. 23). Эти файлы журналов применяются при восстановлении утерянных и поврежденных данных. Файл журнала восстановления и файл хронологии восстановления нельзя изменять вручную, однако вы можете удалить записи из файла хронологии восстановления с помощью команды PRUNE HISTORY. Кроме того, в параметре конфигурации базы данных *rec_his_retentn* можно указать число дней, в течение которых должен храниться файл хронологии восстановления.

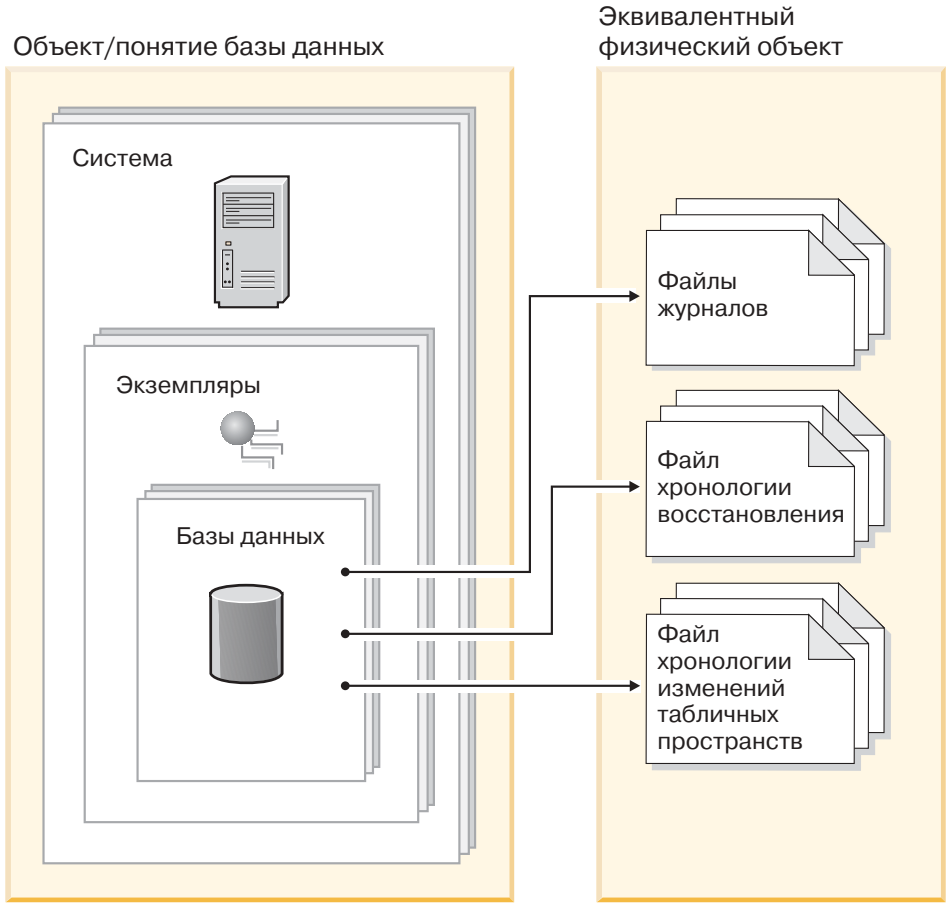

Рисунок 12. Файлы журнала восстановления и файл хронологии восстановления

Каждая база данных включает *журналы восстановления*, которые используются для восстановления информации при ошибках программ или системы. В сочетании с резервными копиями базы данных они используются для восстановления согласованности базы данных до момента, когда случилась ошибка.

Файл хронологии восстановления содержит сводную информацию о резервном копировании, с помощью которой можно определить опции восстановления, когда нужно восстановить всю базу данных или ее часть до заданного момента времени. Он используется для отслеживания событий, связанных с восстановлением, в частности, операций резервного копирования и восстановления. Этот файл расположен в каталоге базы данных.

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

Те данные, которые легко воссоздать, можно хранить в невосстановимых базах данных. К таким данным относятся данные из внешних источников, используемые только для чтения,а также данные, малый объем операций с которыми делает неоправданным дополнительную сложности управления файлами журналов и повтора транзакций при восстановлении. У *невосстановимых баз данных* оба параметра конфигурации, *logretain* и *userexit*, отключены. Это значит, что хранятся только журналы, требующиеся для восстановления после отказа. Эти журналы называются *активными журналами*; они содержат данные текущих транзакций. Восстановление версии при помощи *автономных* резервных копий - основное средство восстановления невосстановимой базы данных. (Автономность резервной копии означает, что во время резервного копирования другие программы не используют базу данных.) Такую базу данных можно восстановить только в автономном режиме. Она восстанавливается до состояния, предшествующего созданию образа резервной копии. Восстановление путем повтора транзакций для такой базы данных не поддерживается.

Данные, которые *нельзя* легко воссоздать, надо хранить в восстановимой базе данных. К ним относятся данные, источник которых удаляется после их загрузки, и данные, которые модифицируются прикладными программами или пользователями после загрузки. У *восстановимых баз данных* параметр конфигурации *logretain* имеет значение “RECOVERY”, или включен параметр конфигурации *userexit*, либо выполнены оба эти условия. Активные журналы для восстановления после сбоя по-прежнему доступны, но кроме того, появляются *архивные журналы* с данными принятых транзакций. Такую базу данных можно восстановить только в автономном режиме. Она восстанавливается до состояния, предшествующего созданию образа ее резервной копии. Однако при восстановлении с повтором транзакций можно выполнить *повтор транзакций* (то есть пройти дальше момента создания образа резервной копии), используя активные и архивные журналы либо до заданного момента времени, либо до окончания активных журналов.

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

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

Понятия, связанные с данным:

- “Crash Recovery” в книге *Справочное руководство по восстановлению данных и высокой доступности*
- “Version Recovery” в книге *Справочное руководство по восстановлению данных и высокой доступности*
- “Rollforward Recovery” в книге *Справочное руководство по восстановлению данных и высокой доступности*
- “Data Links server file backups” в книге *после GA V8*
- “Failure and recovery overview” в книге *DB2 Data Links Manager Administration Guide and Reference*

Ссылки, связанные с данной темой:

- “Параметр конфигурации Период хранения хронологии восстановления - rec_his_retentn” в книге *Руководство администратора: Производительность*
- “Рекомендации по настройке и резервному копированию для DB2 Data Links Manager” в книге *DB2 Data Links Manager Administration Guide and Reference*

Защита

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

Понятия, связанные с данным:

- “Аутентификация” на стр. 26
- “Авторизация” на стр. 27

Аутентификация

Аутентификация пользователя выполняется с использованием внешних по отношению к DB2 средств. Это средство защиты может быть частью операционной системы, отдельным продуктом, а в некоторых случаях может и вовсе не существовать. В системах на базе UNIX® такое средство защиты предусмотрено в самой операционной системе.

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

После аутентификации:

- Пользователь должен быть идентифицирован в DB2® с помощью имени авторизации SQL или *ID авторизации*. Это имя может быть тем же ID пользователя или отображенным значением. Например, в системах на основе UNIX *ID авторизации* DB2 получается преобразованием ID пользователя UNIX в верхний регистр, что соответствует соглашениям об именах DB2.
- Должен быть получен список групп, к которым принадлежит данный пользователь. Членство в группе может использоваться при авторизации пользователя. Группы - это объекты защиты, которые также нужно отобразить на имена авторизации DB2. Это отображение проводится подобно тому, как это делается для ID пользователей.

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

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

DB2 UDB в AIX® может регистрировать попытки ввода неверных паролей средствами операционной системы и определять, когда клиент превысит число разрешенных попыток входа в систему, заданное в параметре LOGINRETRIES.

Понятия, связанные с данным:

- “Защита” на стр. 25

Авторизация

Авторизация - это процесс, во время которого DB2® получает информацию об аутентифицированном пользователе, включающую сведения об операциях, которые он может выполнять, и объектах, к которым можно обращаться. Для каждого требования пользователя может выполняться несколько проверок авторизации в зависимости от вовлекаемых объектов и операций.

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

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

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

Понятия, связанные с данным:

- “Authorization and privileges” в книге *SQL Reference, Том 1*

- “Привилегии, полномочия и авторизация” в книге *Руководство администратора: Реализация*
- “Защита” на стр. 25

Глава 2. Параллельные системы баз данных

Разделение данных

DB2® расширяет возможности менеджера баз данных на параллельную многоузловую среду. *Раздел базы данных* - это часть базы данных, куда входят собственные данные, индексы, файлы конфигурации и журналы транзакций. Раздел базы данных называют иногда узлом или узлом базы данных.

Одноразделная база данных - это база данных с единственным разделом. В этом разделе хранятся все данные базы данных. В этом случае группы разделов базы данных, даже если они заданы, не дают дополнительных возможностей.

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

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

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

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

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

пользователям, обращающимся к данным многораздельной базы данных, не нужно знать физического местоположения этих данных.

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

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

Второй возможный вариант для размещения таблиц на каждом разделе базы данных - использование материализованных таблиц запросов и последующая их репликация. Вы можете создать материализованную таблицу запросов, содержащую всю необходимую информацию, а затем реплицировать ее на все узлы.

Параллелизм

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

- Параллелизм ввода-вывода
- Параллелизм запросов
- Параллелизм утилит

Параллелизм ввода-вывода

Если в табличном пространстве есть несколько контейнеров, менеджер баз данных может использовать *параллельный ввод/вывод*. Параллельный

ввод/вывод - это процесс одновременной записи или чтения на нескольких устройствах ввода/вывода; он может значительно улучшить пропускную способность.

Параллелизм запросов

Параллелизм запросов бывает двух типов: внутренний и внешний.

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

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

Внутрираздельный параллелизм

Внутрираздельный параллелизм - это возможность разбивать запрос на несколько частей. Некоторые утилиты DB2 также применяют этот тип параллелизма.

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

На рис. 13 на стр. 32 показан запрос, который разбивается на четыре составляющие, запускаемые параллельно; результаты при этом возвращаются быстрее, чем при выполнении запроса последовательным способом. Эти составляющие являются копиями друг друга. Для использования внутрираздельного параллелизма нужно соответствующим образом сконфигурировать базу данных. Степень параллелизма можно задать самому или позволить это сделать системе. Степень параллелизма - это число составляющих запроса, запускаемых параллельно.

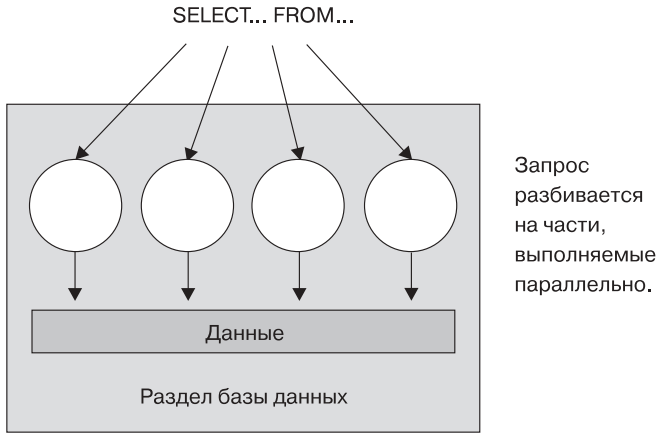

Рисунок 13. Внутрираздельный параллелизм

Межраздельный параллелизм

Межраздельный параллелизм - это возможность разбить запрос на несколько частей по нескольким разделам многораздельной базы данных на одном или нескольких компьютерах. Запрос выполняется параллельно. Некоторые утилиты DB2 также применяют этот тип параллелизма.

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

На рис. 14 на стр. 33 показан запрос, который разбивается на четыре составляющие, запускаемые параллельно; результаты при этом возвращаются быстрее, чем при выполнении запроса последовательным способом на одном компьютере.

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

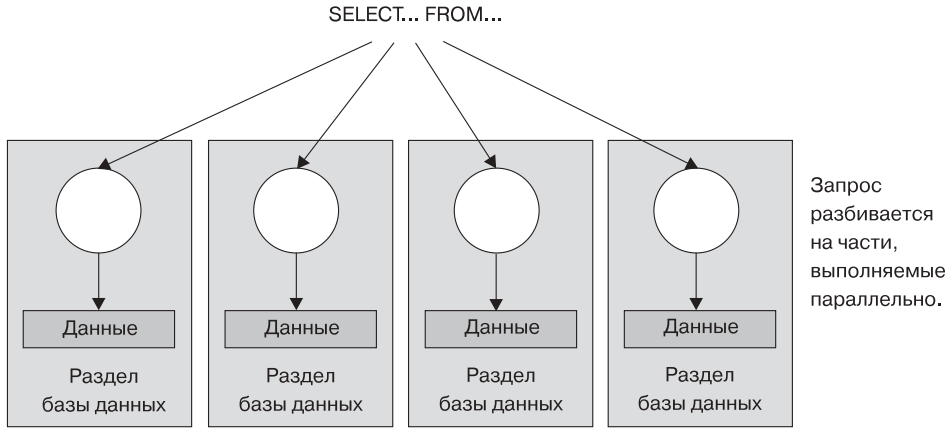

Рисунок 14. Межраздельный параллелизм

Одновременный внутриразделный и межраздельный параллелизм

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

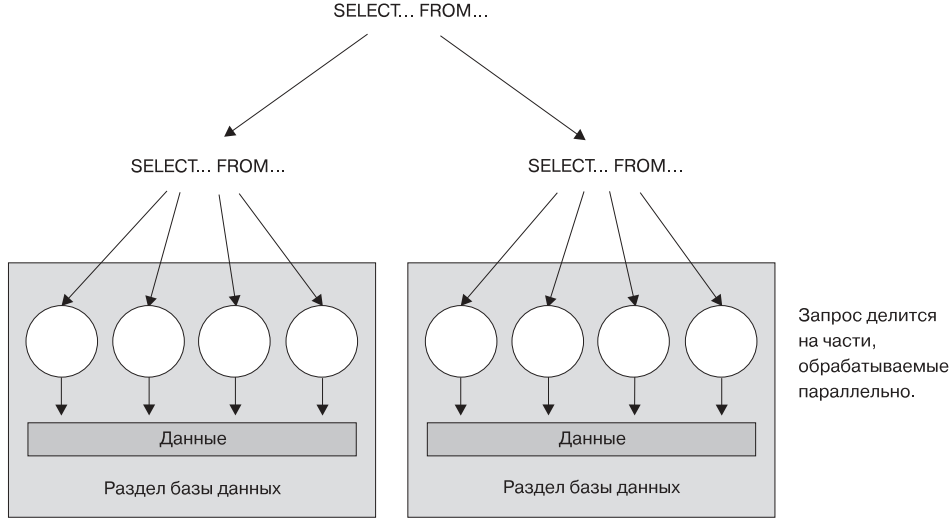

Рисунок 15. Одновременный межраздельный и внутриразделный параллелизм

Параллелизм утилит

Утилиты DB2 могут использовать преимущества внутриразделного параллелизма. Кроме того, они могут использовать межраздельный

параллелизм; если заданы разделы многораздельной базы данных, утилиты запускаются в каждом разделе параллельно.

Утилита загрузки может использовать преимущества внутриразделного параллелизма. Загрузка данных - задача с интенсивным использованием процессора. Утилита загрузки использует преимущества нескольких процессоров для таких задач, как синтаксический анализ и форматирование данных. Кроме того, она может использовать серверы параллельного ввода/вывода для параллельной записи данных в контейнеры.

В среде многораздельных баз данных команда LOAD использует преимущества внутриразделного параллелизма, межраздельного параллелизма и параллелизма ввода/вывода с применением параллельных вызовов на каждом разделе базы данных, где находятся таблицы.

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

Резервное копирование и восстановление данных - задачи с использованием интенсивного ввода/вывода. DB2 использует в операциях резервного копирования и восстановления как параллелизм ввода/вывода, так и внутриразделный параллелизм. Параллелизм ввода/вывода в резервном копировании используется для параллельного чтения из нескольких контейнеров табличных пространств и для параллельной асинхронной записи на несколько носителей резервных копий.

Понятия, связанные с данным:

• “Среды с различным числом процессоров и разделов” на стр. 34

Среды с различным числом процессоров и разделов

В этом разделе описываются следующие аппаратные среды:

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

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

Одноразделная с одним процессором

Эта среда состоит из памяти, диска и единственного процессора (смотрите рис. 16). Она может называться по-разному, например, автономной базой данных, базой данных клиент/сервер, последовательной базой данных, однопроцессорной системой и одноузловой или непараллельной средой.

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

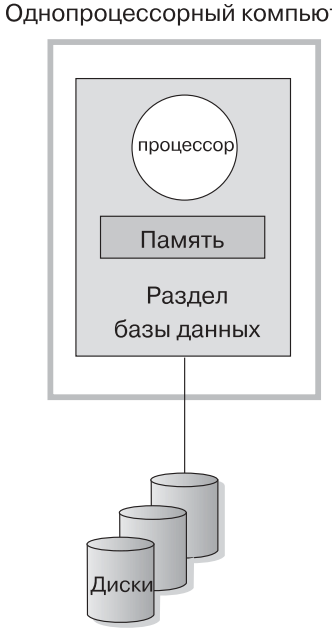

Рисунок 16. Один раздел с одним процессором

Мощность и масштабируемость

В этой среде можно добавлять дополнительные диски. Наличие одного или нескольких серверов ввода/вывода для каждого диска допускает несколько одновременных операций ввода/вывода.

Однопроцессорная система ограничивается тем дисковым пространством, с которым может справиться процессор. С увеличением рабочей нагрузки

единственный процессор может оказаться не в состоянии обработать запросы пользователей за приемлемое время, как бы вы не наращивали другие компоненты, например, памяти или диска. При достижении порога емкости или масштабируемости можно рассмотреть переход к одноразделной среде с несколькими процессорами.

Один раздел с несколькими процессорами

Эта среда обычно состоит из нескольких процессоров одинаковой мощности на одном компьютере (смотрите рис. 17) и называется *симметричной мультипроцессорной системой (SMP)*. При этом такие ресурсы, как дисковое пространство и память, *используются совместно*.

Если есть несколько процессоров, различные операции базы данных могут быть выполнены быстрее. Чтобы увеличить скорость обработки, DB2 может также поделить обработку одного запроса между несколькими доступными процессорами. Преимущество нескольких процессоров может использоваться и другими операциями, например при загрузке данных, резервном копировании и восстановлении табличных пространств, а также при создании индексов для существующих данных.

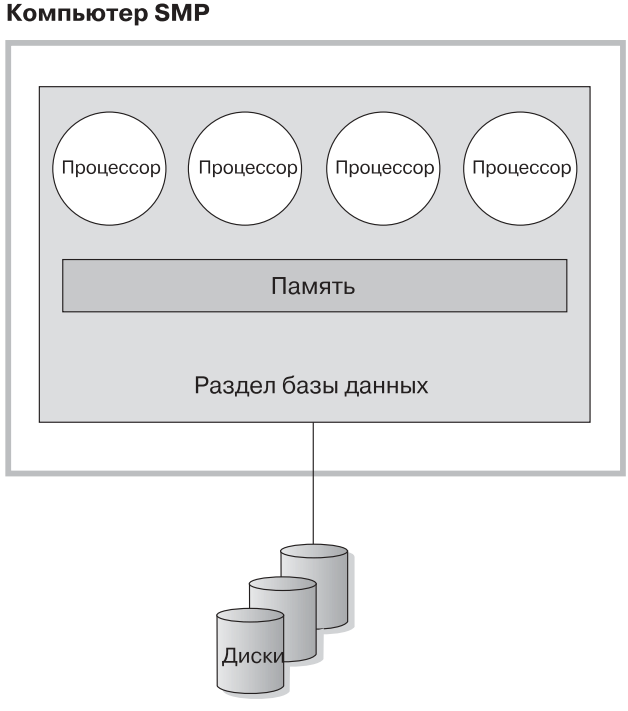

Рисунок 17. Симметричная мультипроцессорная система с одноразделной базой данных

Мощность и масштабируемость

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

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

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

Многораздельные конфигурации

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

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

Многораздельная с одним процессором

В этой среде существует несколько разделов базы данных. Каждый раздел располагается на своем компьютере и у него есть свой процессор, память и диски (смотрите рис. 18 на стр. 38). Все компьютеры соединяются средствами связи. Эта среда может называться по-разному, например, кластером, кластером однопроцессорных систем, средой широкомасштабной параллельной обработки (MPP) и конфигурацией с архитектурой ″ничто не используется совместно″. Последнее название точно отражает распределение ресурсов в этой среде. В отличие от среды SMP, в среде MPP память и диски не используются совместно. Среда MPP устраняет ограничения, которые ставит совместное использование памяти и дисков.

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

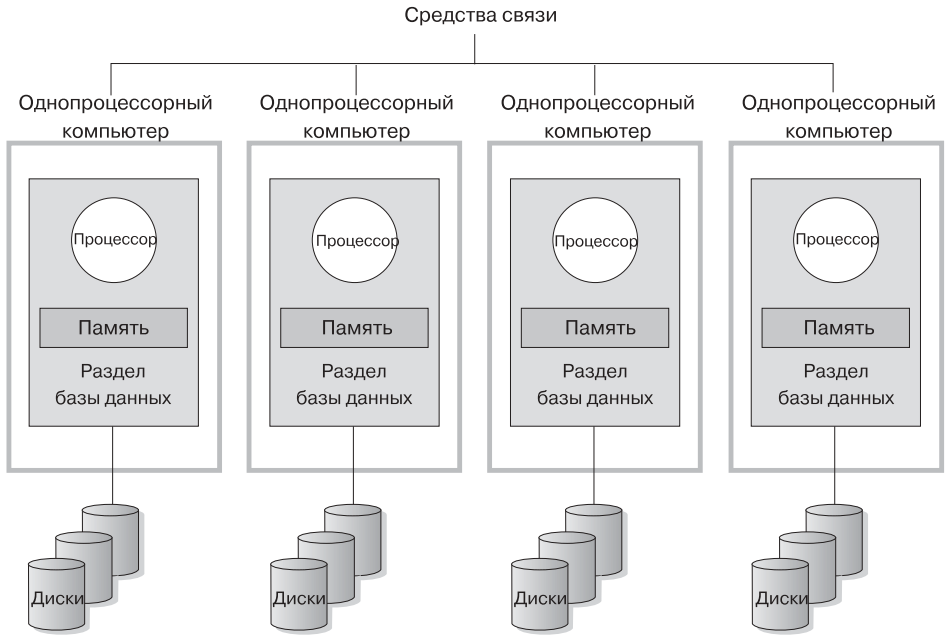

Рисунок 18. Система широкомасштабной параллельной обработки

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

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

Многораздельная с несколькими процессорами

Альтернатива конфигурации, в которой каждый раздел имеет единственный процессор - это конфигурация, где раздел использует несколько процессоров. Такую среду называют *кластером SMP* (смотрите рис. 19 на стр. 39).

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

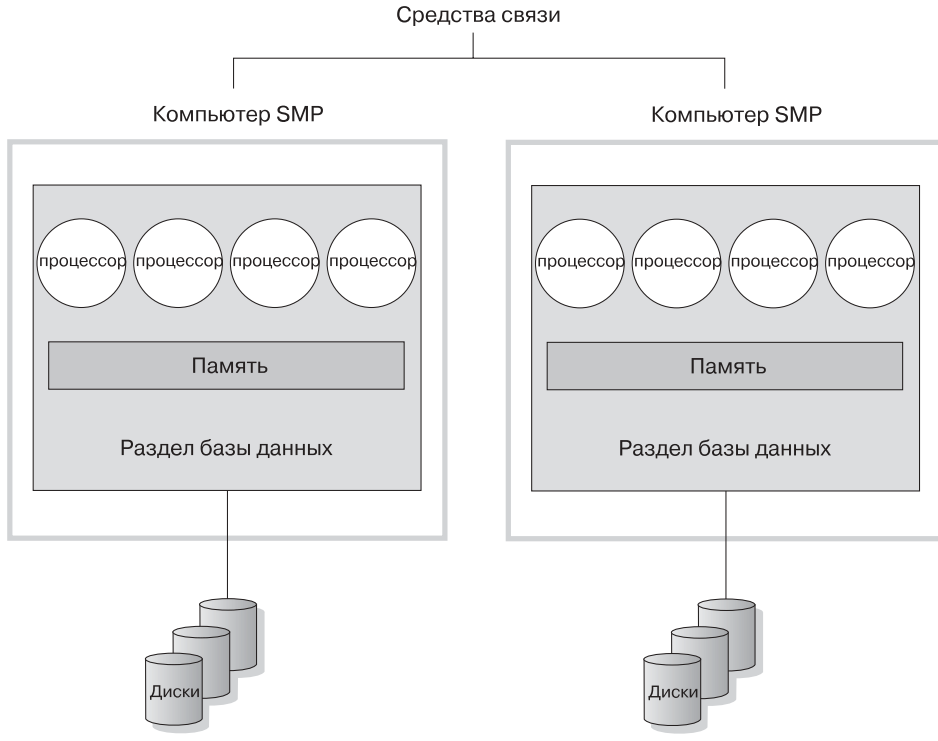

Рисунок 19. Кластер сред SMP

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

Логические разделы базы данных

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

Логические разделы базы данных обеспечивают масштабируемость. Несколько менеджеров баз данных, работающие в нескольких логических разделах, могут полнее использовать доступные ресурсы, чем один менеджер баз данных. На рис. 20 на стр. 40 показано, что на компьютере SMP можно добиться масштабируемости, добавив дополнительные разделы; это характерно для компьютеров с несколькими процессорами. Разбив базу данных на разделы, можно будет управлять каждым разделом и восстанавливать его отдельно от остальных.

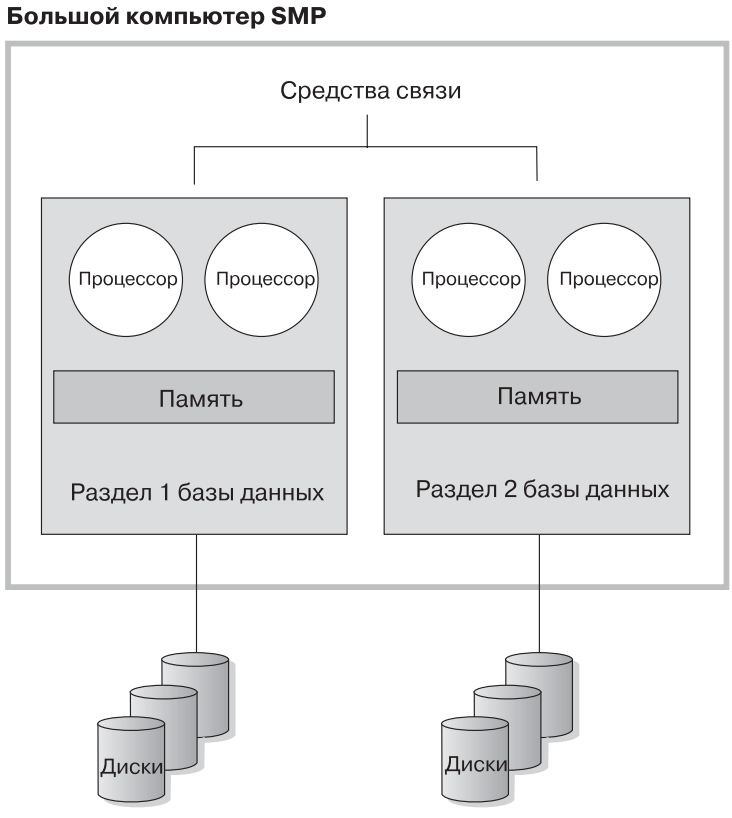

Рисунок 20. Многораздельная база данных, симметричная мультипроцессорная система

На рис. 21 на стр. 41 показано, как распространить конфигурацию, показанную на рис. 20, на несколько компьютеров, чтобы увеличить мощность обработки.

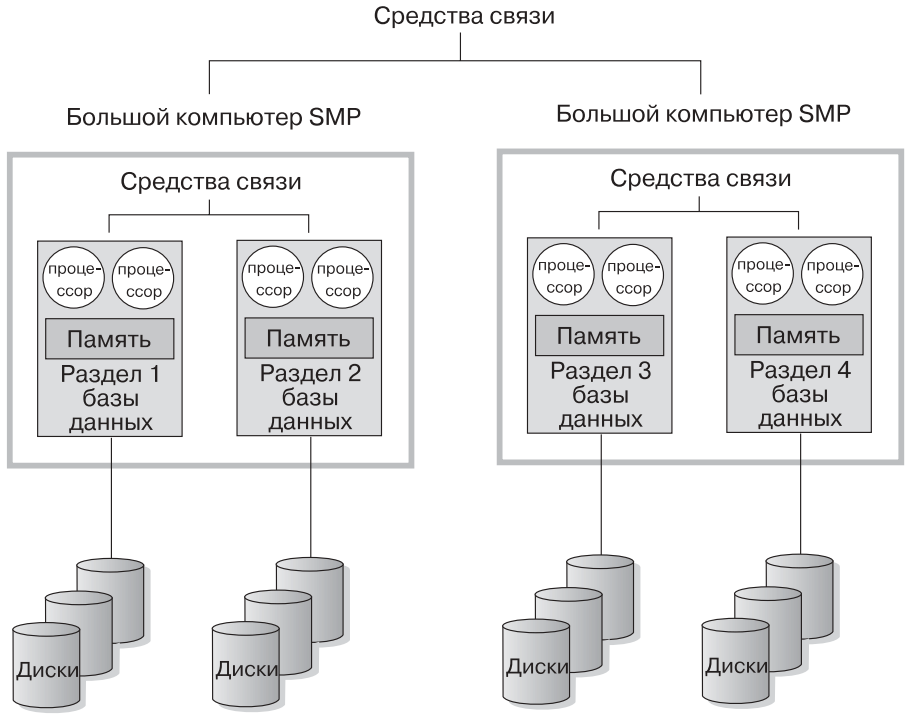

Рисунок 21. Многораздельная база данных, совместная кластеризация симметричных мультипроцессорных систем

Примечание: Возможность сконфигурировать несколько разделов на одном компьютере (независимо от числа процессоров) дает большую гибкость при разработке конфигураций с высокой доступностью к данным и стратегий восстановления после сбоев. При сбое компьютера раздел баз данных может быть автоматически перемещен и перезапущен на другом компьютере, где уже работает другой раздел этой базы данных.

Типы параллелизма, наиболее подходящие для различных аппаратных сред

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

Таблица 3. Типы параллелизма, возможные в каждой аппаратной среде

Аппаратная среда	**Параллелизм ввода/вывода**	**Внутренний параллелизм**	
		Внутрираздельный параллелизм	**Межраздельный параллелизм**
Один раздел, один процессор	Да	Нет(1)	Нет
Один раздел, несколько процессоров (SMP)	Да	Да	Нет

Таблица 3. Типы параллелизма, возможные в каждой аппаратной среде (продолжение)

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

Понятия, связанные с данным:

• “Параллелизм” на стр. 30

Глава 3. О хранилищах данных

Что такое хранилище данных?

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

Однако, при обращении аналитиков к рабочим данным напрямую может возникнуть несколько сложностей:

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

Эти проблемы решают хранилища данных. В *хранилище данных* вы можете поместить *содержательные данные*. Информационные данные, извлеченные из рабочих данных и преобразованные для принятия решений. Например, средствами хранилища данных можно скопировать данные продаж из рабочей базы данных, очистить данные, выполнить вычисления для составления по этой информации сводных данных и записать полученные сводные данные в другую базу данных, отдельную от рабочей. Пользователи могут строить запросы к этой отдельной базе данных (*хранилищу*), не затрагивая рабочих баз данных.

Объекты хранилища данных

В следующем разделе описываются объекты которые можно использовать при создании и поддержке хранилища данных.

Тематические области

Тематическая область идентифицирует и группирует процессы, связанные с логической областью деятельности. Например, при построении хранилища

данных по маркетингу и продажам можно определить тематическую область Продажи и тематическую область Маркетинг. Затем вы добавляете процессы, связанные с продажами, в тематическую области Продажи. Подобным образом определения, связанные с данными маркетинга, добавляются в тематическую область Маркетинг.

Источники хранилища

Источники хранилища данных идентифицируют таблицы и файлы, которые будут обеспечивать данными хранилище. Центр хранилищ данных использует спецификации в источниках хранилища для обращения к данным. Источником может быть практически любой реляционный или нереляционный источник (таблица, производная таблица, файл или заранее определенный псевдоним), источник SAP R/3 или WebSphere® Site Analyzer, который подключен к вашей сети.

Потребители хранилища

Потребители хранилища - это таблицы или файлы баз данных, содержащие преобразованные данные. Как и источники хранилища данных, потребители хранилища применяются пользователями для предоставления данных другим потребителями хранилища. Центральной хранилище может предоставлять данные серверам отделов, а главная таблица фактов в хранилище данных может предоставлять данные для сводных таблиц.

Агенты хранилища и узлы агентов хранилища

Агенты хранилища управляют потоком данных между источниками и потребителями данных. Агенты хранилища доступны в операционных системах AIX, Linux, iSeries, z/OS, Windows® NT, Windows 2000 и Windows XP, а также для операционной среды Solaris. Для связи с другими базами данных агенты используют драйверы Open Database Connectivity (ODBC) или DB2® CLI.

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

Агенты могут быть локальными или удаленными. Локальный агент хранилища устанавливается на той же рабочей станции, что и сервер хранилища. Удаленный агент хранилища устанавливается на другой рабочей станции, откуда доступна связь с сервером хранилища.

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

Узел агента по умолчанию, называемый Default DWC Agent Site, - это локальный агент, который Центр хранилищ данных определяет во время установки управляющей базы данных хранилища.

Процессы и шаги

Процесс состоит из серий шагов, выполняющих задачи по преобразованию и перемещению данных для конкретного хранилища. Как правило, процесс перемещает данные в хранилище. Затем данные агрегируются и суммируются для применения в хранилище. В результате выполнения процесса может быть получена отдельная одномерная таблица или набор сводных таблиц. Процесс также может выполнять некоторый вид преобразования данных.

Шаг - определение отдельной операции в рамках хранилища данных. С помощью операторов SQL и вызовов программ, шаги определяют способ перемещения и преобразования данных. При запуске шага может выполняться передача данных между источником хранилища и потребителем хранилища и все необходимые преобразования этих данных.

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

- Связь с исходными данными.
- Определение выходной таблицы или файла, и связь с ней.
- Механизм (оператор SQL или программу) для определения и заполнения выходной таблицы или файла
- Опции обработки и расписание, по которому заполняется выходная таблица или файл.

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

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

В этом случае нужно создать процесс, содержащий несколько шагов. Каждый шаг выполняет отдельную задачу, например извлекает данные из баз данных или преобразует их в нужный формат. Для полного преобразования, форматирования данных и их внесения в конечную таблицу может потребоваться создание нескольких шагов.

При выполнении шаг или процесс может повлиять на данные одним из следующих способов:

- Заменить все данные в потребителе хранилища новыми.
- Добавить новые данные к существующим.
- Добавить отдельную редакцию данных.

- Изменить существующие данные

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

В следующих разделах описываются разные типы шагов, которые вы найдете в Центре хранилищ данных.

Шаги SQL

Существует два типа шагов SQL. Шаг SQL Выбрать и Вставить использует оператор SQL SELECT для извлечения данных из источника хранилища и генерирует оператор INSERT для вставки данных в таблицу назначения хранилища. Шаг SQL Выбрать и Изменить получает данные из источника хранилища с помощью оператора SQL SELECT и изменяет существующие данные в таблице назначения хранилища.

Шаги программ

Существует несколько типов шагов программ: программы DB2 for iSeries™, DB2 for z/OS™, DB2 for UDB, Visual Warehouse™ 5.2 DB2, программы сервера OLAP, программы работы с файлами и шаги репликации. Эти шаги запускают определенные программы и утилиты.

Преобразователи

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

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

Шаги пользовательских программ

Шаг пользовательской программы - это логический объект в Центре хранилищ данных, представляющий преобразование, связанное с деятельностью компании, которое должен запускать Центр хранилищ данных. Так как во всех компаниях выполняются разные преобразования, отдельная компания может создать собственные шаги программ или воспользоваться такими инструментами, как средства ETI или Vality.

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

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

Соединители

Следующие соединители входят в состав Менеджера хранилищ DB2, но приобретаются отдельно. Эти соединители помогут вам извлекать данные и метаданные из репозиториев электронной коммерции.

- Соединитель DB2 Warehouse Manager для SAP R/3
- Соединитель менеджера хранилищ данных DB2 для Web

При помощи соединителя для SAP R/3 можно добавлять извлеченные данные в хранилище данных, преобразовывать их с использованием Центра хранилищ данных DB2 или анализировать их с использованием инструментов DB2 или средств других производителей. При помощи соединителя для Web можно переносить данные ″потока действий пользователя″ из IBM® WebSphere Site Analyzer в хранилище данных.

Задачи хранилища данных

При создании хранилища данных решаются следующие задачи:

- Идентификация данных источника (или рабочих данных) и определение их в качестве источников хранилища.
- Создание базы данных для использования в качестве хранилища и определение потребителей хранилища.
- Определение тематической области для групп процессов, которые будут определены в хранилище.
- Задание способов перемещения и преобразования данных источника в формат для базы данных хранилища с помощью определения шагов процесса.
- Проверка уже определенных шагов и составление расписания их автоматического запуска.
- Управление хранилищем с использованием задаваемой защиты и мониторинг базы данных с помощью записной книжки Выполняемые задания.
- Создание каталога данных в хранилище, если у установлен Менеджер хранилищ DB2®. Каталог данных - это база данных, содержащая деловые метаданные. Метаданные компании, с помощью которых пользователи идентифицируют и находят доступную им информацию в организации. Метаданные хранилища данных могут быть опубликованы в каталоге данных. В каталоге данных можно выполнять поиск, чтобы определять, какие данные доступны в хранилище.

- Определение модели схемы типа звезда для данных в хранилище. Схема типа звезда - это особая структура, которая состоит из нескольких таблиц ассоциаций, описывающих различные аспекты деятельности предприятия, и одной таблицы фактов, содержащей соответствующие фактические материалы и показатели. Например, если ваша фирма производит товары, некоторые таблицы ассоциаций могут описывать продукты, рынки и сроки. Таблица фактов содержит информацию транзакций о продуктах, заказываемых в каждом регионе в зависимости от времени года.

Часть 2. Проектирование баз данных

Глава 4. Логическая структура базы данных

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

Проектирование базы данных - это нелинейный процесс; скорее всего, при разработке структуры базы данных некоторые шаги придется повторять.

Что хранить в базе данных

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

Объект - это человек, предмет или понятие, о которых вы хотите хранить информацию. Примеры объектов в таблице примеров - это сотрудники, отделы и проекты.

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

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

Таблица 4. Экземпляры объекта сотрудник и их атрибуты

EMPNO	FIRSTNME	LASTNAME	WORKDEPT	JOB
000010	Christine	Haas	A00	Президент
000020	Michael	Thompson	B01	Начальник
000120	Sean	O'Connell	A00	Клерк

Таблица 4. Экземпляры объекта сотрудник и их атрибуты (продолжение)

EMPNO	FIRSTNME	LASTNAME	WORKDEPT	JOB
000130	Dolores	Quintana	C01	Аналитик
000030	Sally	Kwan	C01	Начальник
000140	Heather	Nicholls	C01	Аналитик
000170	Masatoshi	Yoshimura	D11	Проектировщик

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

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

- Сотрудники распределены по отделам
  - Долорес Кинтана работает в отделе C01
- Сотрудники работают на определенных должностях
  - Долорес работает аналитиком
- Сотрудники руководят отделами
  - Салли руководит отделом C01.

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

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

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

Понятия, связанные с данным:

- “Отношения в базе данных” на стр. 53
- “Определения столбцов” на стр. 56

Отношения в базе данных

В базе данных может быть определено несколько типов отношений. Рассмотрим возможные отношения между сотрудниками и отделами.

Отношения "один-ко-многим" и "многие-к-одному"

Сотрудник может работать только в одном отделе - это *однозначное* отношение для сотрудника. Но в одном отделе может быть много сотрудников - это *многозначное* отношение для отдела. Отношение между отделами и сотрудниками - это отношение типа *один-ко-многим*.

Чтобы определить таблицы для каждого отношения один-ко-многим и многие-к-одному:

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

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

Таблица 5. Отношение Многие-к-одному

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

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

Ниже показано, как эти отношения отражаются в таблицах:

Таблица EMPLOYEE:

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

Таблица DEPARTMENT:

DEPTNO	ADMRDEPT
C01	A00
D01	A00
D11	D01

Отношения многие-ко-многим

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

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

Таблица деятельности сотрудников (EMP_ACT):

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

Отношения один-к-одному

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

В следующем примере показано, как отношение один-к-одному отражается в таблице:

Таблица DEPARTMENT:

DEPTNO	MGRNO
A00	000010
B01	000020
D11	000060

Одинаковые значения должны соответствовать одному объекту

Возможны несколько таблиц, описывающих атрибуты одного набора объектов. Например, в таблице EMPLOYEE есть номер отдела, в котором работает сотрудник, а в таблице DEPARTMENT - начальник, руководящий отделом с данным номером. Чтобы получить оба набора атрибутов одновременно, можно объединить эти две таблицы по совпадающему столбцу, как показано в следующем примере. Значения в WORKDEPT и DEPTNO соответствуют одному и тому же объекту, что дает *способ объединения* таблиц DEPARTMENT и EMPLOYEE.

Таблица DEPARTMENT:

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
D21	Административный отдел	000070	D01

Таблица EMPLOYEE:

EMPNO	FIRSTNAME	LASTNAME	WORKDEPT	JOB
000250	Daniel	Smith	D21	Клерк

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

Понятия, связанные с данным:

- “Что хранить в базе данных” на стр. 51
- “Определения столбцов” на стр. 56

Определения столбцов

Чтобы определить столбец в реляционной таблице:

1. Выберите имя для столбца.

   У каждого столбца должно быть уникальное в своей таблице имя.

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

   Тип данных и *длина* указывают тип данных и максимальную длину, допустимые для столбца. Можно выбрать типы данных из предоставляемых менеджером баз данных или же создать свои собственные пользовательские типы.

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

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

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

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

   Структурированный тип - это пользовательский тип, структура которого определена в базе данных. Он содержит набор именованных *атрибутов*, у каждого из которых есть тип. Структурированный тип может быть

определен как *подтип* другого структурированного типа, называемого его *надтипом*. Подтип наследует все атрибуты своего надтипа, но для него можно определить и дополнительные атрибуты. Набор структурированных типов, относящихся к одному надтипу, называется *иерархией типов*, а надтип, для которого нет надтипа, называется *корневым типом* иерархии типов.

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

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

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

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

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

- Внешняя функция, которая написана на одном из языков программирования

- Функция с источником, используемая для вызова других пользовательских функций

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

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

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

   - Значение столбца неприменимо к этой строке.

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

   - Значение применимо, но пока неизвестно.

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

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

Первичные ключи

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

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

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

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

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

Если у таблицы нет ″естественного″ ключа уникальности, или если для различения строк используется последовательность их появления в базе, может быть полезно использовать отметки времени.

Первичные ключи для некоторых таблиц из примеров:

Таблица	**Ключевой столбец**
Таблица сотрудников (Employee)	EMPNO
Таблица отделов (Department)	DEPTNO
Таблица проектов (Project)	PROJNO

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

Таблица 6. Первичный ключ таблицы PROJECT

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

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

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

Таблица 7. Составной первичный ключ для таблицы EMP_ACT

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

Определение подходящих ключевых столбцов

Для определения столбцов, которые разумно выбрать ключевыми, найдите наименьшее число столбцов, которое однозначно определяет объект. Может найтись несколько подходящих ключей. В Табл. 8 можно выбирать различные ключи.Каждый из столбцов EMPNO, PHONENO и LASTNAME однозначно определяет сотрудника.

Таблица 8. Таблица EMPLOYEE

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

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

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

Из трех ключей-кандидатов в примере только EMPNO отвечает всем этим критериям. У сотрудника может не быть телефонного номера, когда он начинает работать в компании. Сотрудник может сменить фамилию, и, хотя фамилии случайно могут оказаться уникальными, это не гарантировано. Столбец номеров сотрудников - это наилучший вариант первичного ключа. Сотруднику только однажды присваивается уникальный номер, и этот номер,

как правило, не изменяется за время, пока сотрудник остается в компании. Так как у каждого сотрудника должен быть номер, значения в столбце номеров сотрудников являются определенными.

Понятия, связанные с данным:

• “Столбцы идентификации” на стр. 61

Столбцы идентификации

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

Значения для столбца идентификации могут генерироваться всегда или по умолчанию.

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

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

Счетчик для столбца идентификации увеличивается (или уменьшается) независимо от транзакции. Если некоторая транзакция увеличивает счетчик

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

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

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

Понятия, связанные с данным:

- “Первичные ключи” на стр. 58

Нормализация

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

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

Форма Описание

Первая

Во всех позициях строк и столбцов стоит по *одному* значению, нигде нет набора значений.

Вторая
: Каждый столбец, не входящий в ключ, зависит от ключа.

Третья
: Каждый неключевой столбец независим от других неключевых столбцов; он зависит только от ключа.

Четвертая
: Ни в одной строке не содержится нескольких независимых многозначных фактов об одном объекте.

Первая нормальная форма

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

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

Таблица 9. Таблица, нарушающая первую нормальную форму

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

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

Таблица 10. Таблица, соответствующая первой нормальной форме

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

Вторая нормальная форма

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

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

Таблица 11. Таблица, нарушающая вторую нормальную форму

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

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

Недостатки такой структуры:

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

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

Таблица 12. Таблица PART_STOCK, соответствующая второй нормальной форме

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

Таблица 13. Таблица WAREHOUSE, соответствует второй нормальной форме

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

Таблица 13. Таблица WAREHOUSE, соответствует второй нормальной форме (продолжение)

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

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

Третья нормальная форма

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

Первая таблица следующего примера содержит столбцы EMPNO и WORKDEPT. Допустим, добавляется столбец DEPTNAME (смотрите Табл. 15 на стр. 66). Новый столбец зависит от WORKDEPT, а первичный ключ - это EMPNO. Получившаяся таблица будет нарушать третью нормальную форму. Если мы изменим DEPTNAME для одного из сотрудников, например, для Джона Паркера, это не изменит названий отделов для других сотрудников этого отдела. Но теперь у отдела E11 будет два разных названия. Получающаяся несогласованность показана в измененной версии таблицы.

Таблица 14. Ненормализованная таблица EMPLOYEE_DEPARTMENT до изменения

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

Таблица 15. Ненормализованная таблица EMPLOYEE_DEPARTMENT после изменения. Информация в таблице стала несогласованной.

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

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

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

В результате нормализации будут определены следующие таблицы:

Таблица 16. Таблица EMPLOYEE после нормализации таблицы EMPLOYEE_DEPARTMENT

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

Таблица 17. Таблица DEPARTMENT после нормализации таблицы EMPLOYEE_DEPARTMENT

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

Четвертая нормальная форма

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

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

Таблица 18. Таблица, нарушающая четвертую нормальную форму

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

Вместо этого такие отношения надо представить в двух таблицах:

Таблица 19. Таблица EMPLOYEE_SKILL, соответствующая четвертой нормальной форме

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

Таблица 20. Таблица EMPLOYEE_LANGUAGE, соответствующая четвертой нормальной форме

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

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

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

Многомерная кластеризация

Многомерная кластеризация (MDC) - это удобный способ обеспечить гибкую, непрерывную и автоматическую кластеризацию данных по нескольким измерениям. С ее помощью можно значительно повысить производительность обработки запросов, а также сократить количество выполняемых операций обслуживания данных, таких как реорганизация, и количество операций обслуживания индекса во время вставки, обновления и удаления данных. Многомерная кластеризация в основном предназначена для хранилищ данных и больших баз данных. Кроме того, она может применяться в среде динамической обработки транзакций (OLPT).

Многомерная кластеризация позволяет физически кластеризовать данные таблицы по нескольким ключам, или измерениям. В версиях DB2® младше 8 поддерживается только одномерная кластеризация данных при помощи индексов кластеризации. Индекс кластеризации позволяет поддерживать порядок размещения данных на физических страницах, совпадающий с порядком следования ключей индекса, при вставке и обновлении записей таблицы. Индексы кластеризации значительно повышают производительность обработки запросов с диапазоном значений, содержащих предикаты с ключом индекса кластеризации. Это связано с тем, что в случае хорошей кластеризации достаточно обратиться только к части таблицы, причем эффективность предварительной выборки будет высокой, если страницы расположены последовательно.

Многомерная кластеризация дает те же преимущества, но позволяет применять несколько измерений, или ключей кластеризации. Она повышает производительность обработки тех запросов с диапазоном значений, которые используют одно или несколько измерений таблицы. Выигрыш в производительности дает и то, что потребуется обращаться только к страницам, содержащим записи с указанными значениями измерения, и то, что такие страницы будут сгруппированы по экстентам. Кроме того, обычно степень кластеризации таблицы, для которой создан индекс кластеризации, уменьшается в процессе заполнения таблицы, а таблица с многомерной кластеризацией автоматически и непрерывно поддерживает кластеризацию по всем измерениям. Следовательно, для восстановления исходного порядка размещения данных на диске не потребуется реорганизовывать таблицу.

При создании таблицы можно задать один или несколько ключей в качестве измерений, по которым будет выполняться кластеризация данных. Каждое из этих измерений, как и ключ индекса, может состоять из одного или нескольких столбцов. Для каждого указанного измерения автоматически создается *блочный индекс измерения*, который применяется оптимизатором для быстрого и эффективного доступа к данным, расположенным вдоль этого измерения. Кроме того, автоматически создается *составной блочный индекс*, содержащий все столбцы ключей измерений. Он применяется для поддержания кластеризации данных при выполнении операций вставки и обновления. Составной блочный индекс не создается, если одно из измерений уже содержит все столбцы ключей измерений. Он также иногда применяется оптимизатором для эффективного доступа к данным, содержащим определенные значения измерений.

В таблице с многомерной кластеризацией каждое уникальное сочетание значений измерений образует логическую *ячейку*, которая физически представляет собой блоки страниц. Под блоком понимается набор страниц, последовательно расположенных на диске. Набор блоков, содержащих страницы, данные которых содержат одинаковое значение ключа для одного из блочных индексов измерения, называется *срезом*. Каждая страница таблицы находится только в одном блоке, и все блоки таблицы содержат одинаковое *число страниц*. Число страниц в блоке совпадает с размером экстента, поэтому блоки выровнены по границам экстентов.

Рассмотрим таблицу с многомерной кластеризацией, которая содержит записи о розничных продажах на внутреннем рынке. Таблица кластеризована по измерениям YearAndMonth (год и месяц) и Region (регион). Записи таблицы хранятся в блоках, состоящих из последовательно расположенных страниц дисковой памяти, число которых совпадает с размером экстента. На рис. 22 на стр. 70 блок представлен в виде прямоугольника. Нумерация блоков совпадает с логическим порядком экстентов, выделенных таблице. Сетка на диаграмме представляет логическое разбиение на блоки, а каждый квадрат представляет логическую ячейку. Каждый столбец или строка сетки представляет срез определенного измерения. Например, все записи, содержащие в столбце региона значение 'South-central', находятся в блоках из среза, соответствующего столбцу сетки 'South-central'. Все блоки из этого среза содержат только те записи, в которых задан регион 'South-central'. Следовательно, блок содержится в этом срезе или столбце сетки тогда и только тогда, когда он содержит записи, в которых задан регион 'South-central'.

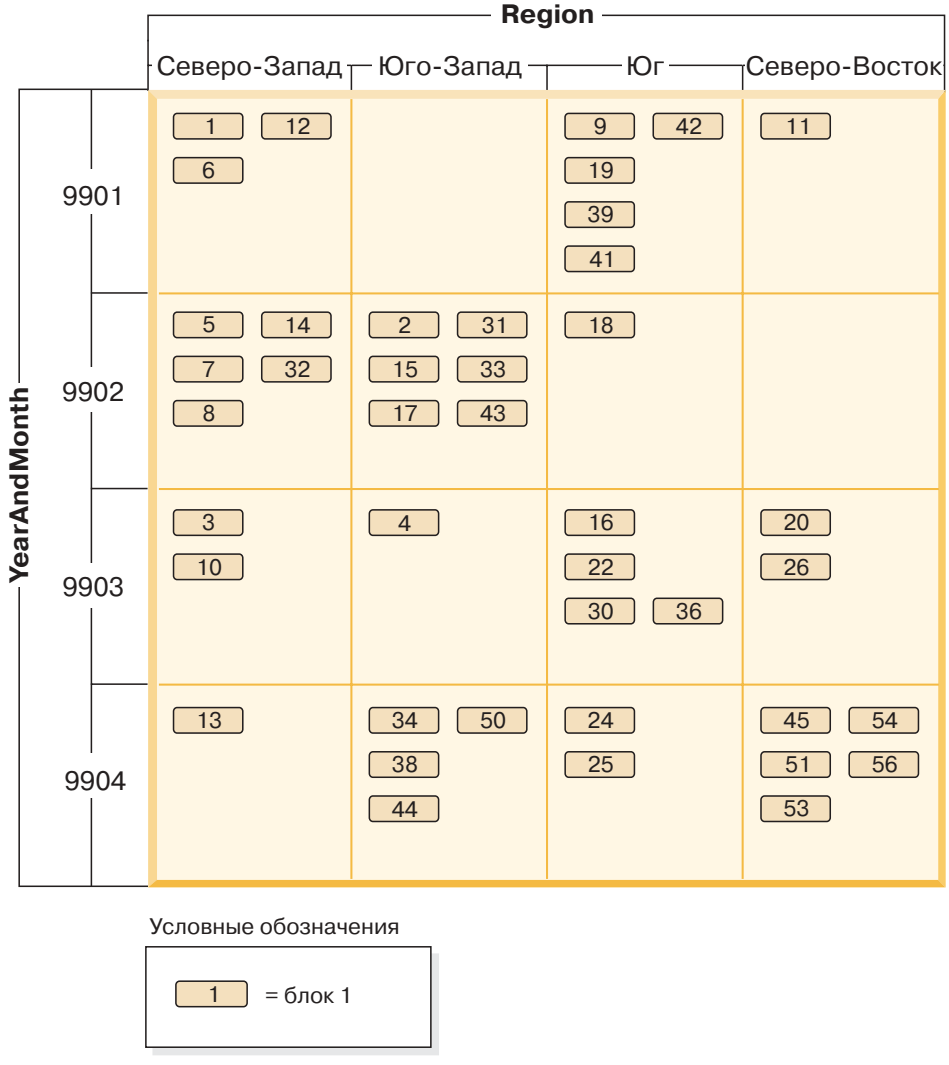

Рисунок 22. Многомерная таблица с измерениями Region и YearAndMonth

Для того чтобы было проще определить, какие блоки образуют срез, то есть какие блоки состоят из записей с одинаковым значением ключа измерения, при создании таблицы для каждого измерения автоматически создается блочный индекс измерения.

На рис. 23 на стр. 71 один блочный индекс создан для измерения YearAndMonth, а второй - для измерения Region. Структура блочных индексов измерения совпадает со структурой обычного индекса RID, за исключением того, что на последнем уровне ключи указывают на идентификатор блока (BID), а не на идентификатор записи (RID). Поскольку каждый блок может содержать много

страниц записей, размер блочных индексов гораздо меньше размера индексов RID. Кроме того, их требуется обновлять только при добавлении новых блоков в ячейку или удалении пустых блоков из ячейки.

Срез, или набор блоков страниц с записями, содержащими определенное значение ключа измерения, представлен в соответствующем блочном индексе измерения в виде списка BID для этого значения ключа.

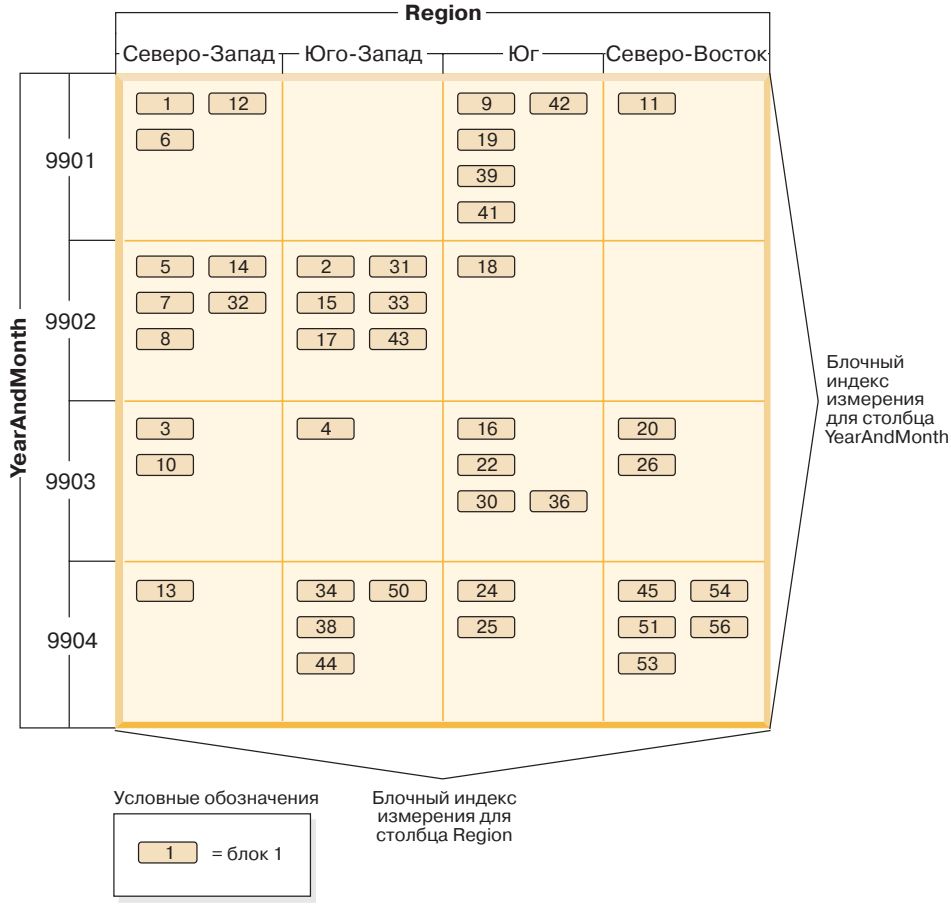

Рисунок 23. Блочные индексы измерений Region и YearAndMonth многомерной таблицы

На рис. 24 на стр. 72 показано, каким образом выглядит ключ блочного индекса измерения Region. Он состоит из значения ключа, 'South-central', и списка BID. Каждый BID содержит расположение блока. На рис. 24 на стр. 72 номера блоков совпадают с номерами блоков из среза 'South-central', показанного на сетке таблицы Sales (см. рис. 22 на стр. 70).

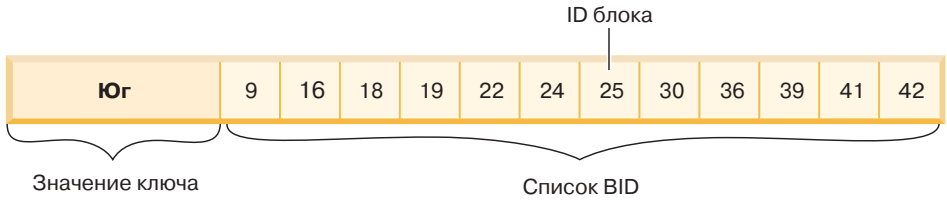

Рисунок 24. Ключ блочного индекса измерения Region

Аналогично, для того чтобы найти блоки, в которых все записи содержат значение ’9902’ в измерении YearAndMonth, необходимо найти это значение в блочном индексе измерения YearAndMonth, показанном на рис. 25.

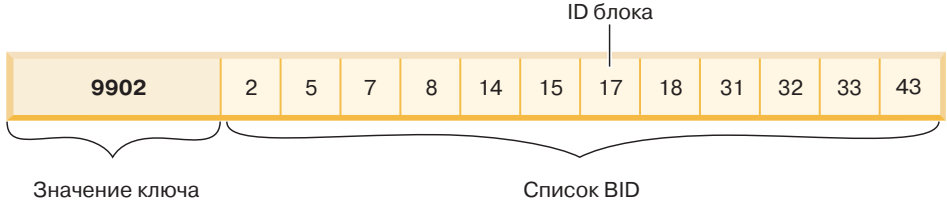

Рисунок 25. Ключ блочного индекса измерения YearAndMonth

При вставке записи в таблицу Sales DB2 определит, существует ли ячейка с такими значениями измерений. Если да, то DB2 вставит запись в существующий блок этой ячейки, либо добавит блок в ячейку, если все текущие блоки ячейки заполнены. Если ячейка не существует, DB2 создаст новую ячейку и добавит в нее блок. Такое автоматическое обслуживание выполняется с помощью дополнительного блочного индекса, создаваемого вместе с таблицей с многомерной кластеризацией. Этот блочный индекс содержит все столбцы измерений таблицы. В нем каждое значение ключа соответствует определенной ячейке таблицы, а список BID, связанный с этим значением, соответствует списку блоков ячейки (см. рис. 26 на стр. 73). Такой тип индекса называется *составным блочным индексом*.

Составной блочный индекс содержит ключ для каждой ячейки таблицы, содержащей записи. Такой блочный индекс применяется для быстрого и эффективного поиска блоков с записями, содержащими определенный набор значений соответствующих измерений. Составной блочный индекс используется при обработке запросов для доступа к данным таблицы, содержащим определенные значения измерений. Кроме того, он применяется для динамического поддержания физической кластеризации данных по измерениям таблицы во время выполнения операций вставки.

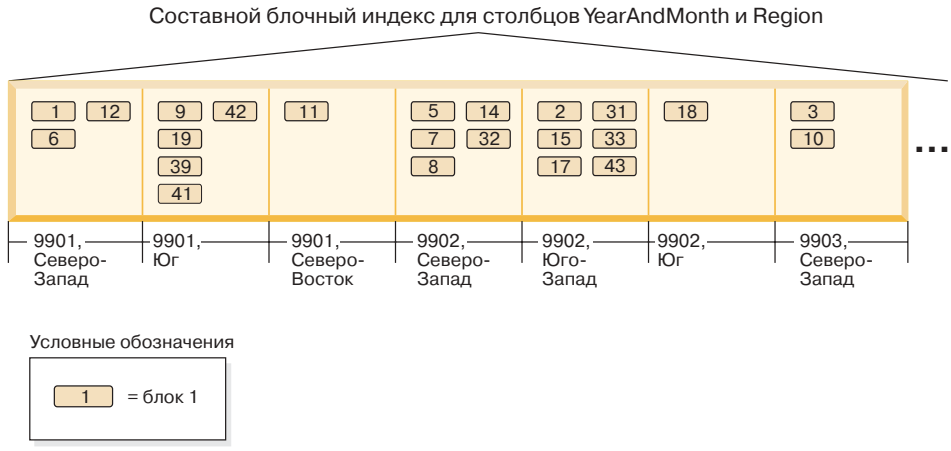

Рисунок 26. Составной блочный индекс для YearAndMonth и Region

Например, для того чтобы найти все записи таблицы Sales, у которых в поле Region содержится значение 'Northwest', а в поле YearAndMonth - значение '9903', DB2 найдет значеник ключа 9903,Northwest в составном блочном индексе, как показано на рис. 27. Ключ состоит из значения ключа, '9903,Northwest', и списка BID. В данном случае список BID содержит только два значения (3 и 10), поэтому только два блока таблицы Sales содержат записи с указанными значениями.

Рисунок 27. Ключ из составного блочного индекса для YearAndMonth и Region

Для того чтобы проиллюстрировать применение этого индекса во время вставки, рассмотрим в качестве примера вставку другой записи со значениями измерений 9903 и Northwest. DB2 найдет соответствующее значение ключа в составном блочном индексе. С этим значением связаны BID блоков 3 и 10. Эти блоки включают все те (и только те) записи, которые содержат указанные значения ключей измерений. Следовательно, DB2 вставит новую запись в один из этих блоков, если на страницах одного из этих блоков есть место. Если все страницы этих блоков заполнены, DB2 выделит таблице новый блок, либо воспользуется одним из пустых блоков таблицы. Обратите внимание, что в этом примере блок 48 не используется таблицей. DB2 вставит запись на страницу

этого блока и выделит этот блок соответствующей ячейке, добавив его BID в составной блочный индекс и во все блочные индексы измерений. На рис. 28 показаны ключи блочных индексов измерений после добавления блока 48.

9903	3	4	10	16	20	22	26	30	36	48

Северо-Запад	1	3	5	6	7	8	10	12	14	32	48

9903, Северо-Запад	3	10	48

Рисунок 28. Ключи блочных индексов измерений после добавления блока 48

Если таблица Sales кластеризована по трем измерениям, то для поиска набора блоков с записями, соответствующими запросу для подмножества всех измерений таблицы, могут применяться отдельные блочные индексы измерений. Если в таблице определены измерения YearAndMonth (год и месяц), Region (регион) и Product (продукт), то ее можно логически представить в виде куба, как показано на рис. 29 на стр. 75.

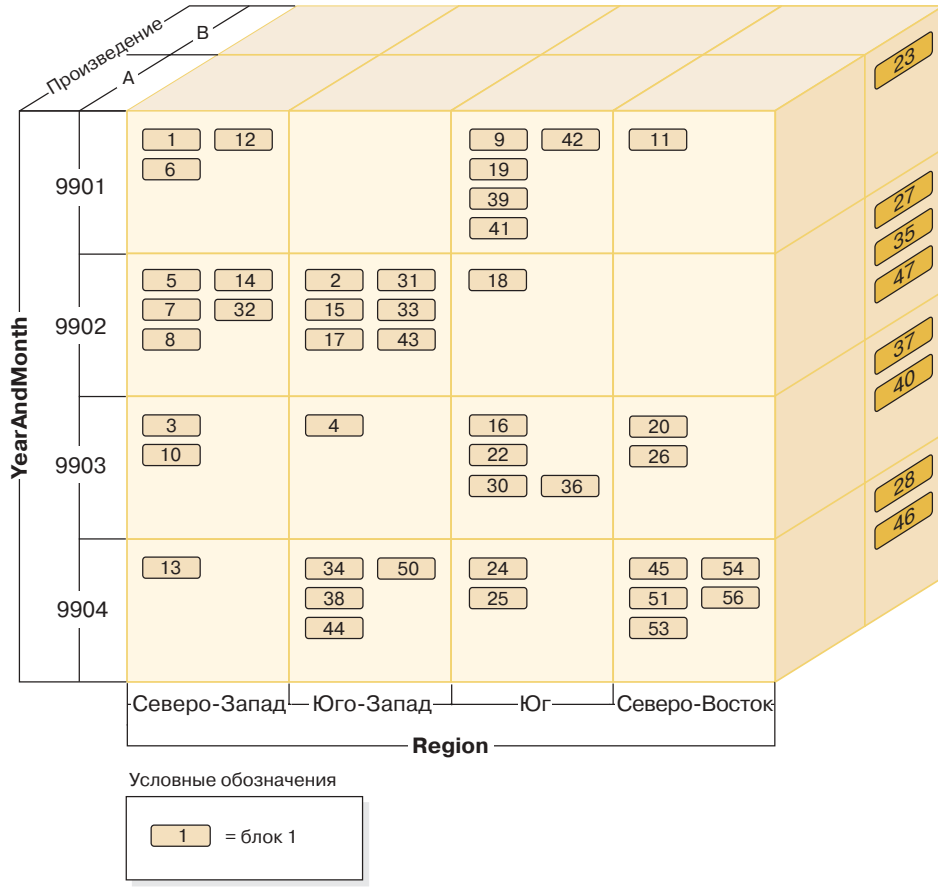

Рисунок 29. Многомерная таблица с измерениями Region, YearAndMonth и Product

Для таблицы с тремя измерениями будет создано четыре блочных индекса: по одному для каждого измерения, YearAndMonth, Region и Product, и один блочный индекс, содержащий все столбцы измерений в качестве ключей. Для того чтобы найти все записи, в которых Product=Product A и Region=Northeast, DB2 определит слой, содержащий все блоки со значением Product=Product A. Для этого в блочном индексе измерения Product будет найден ключ Product A. (Смотрите раздел рис. 30.) Затем DB2 определит блоки, в которых все записи содержат поле Region=Northeast. Для этого в блочном индексе измерения Region будет найден ключ Northeast. (Смотрите раздел рис. 31 на стр. 76.)

Произведение A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...

Рисунок 30. Ключ блочного индекса измерения Product

Северо-Восток	11	20	23	26	27	28	35	37	40	45	46	47	51	53 ...

Рисунок 31. Ключ блочного индекса измерения Region

Для того чтобы найти набор блоков, в которых все записи содержат оба значения, необходимо определить пересечение двух слоев. Для этого над списками BID индексов выполняется логическая операция AND. В обоих списках содержатся следующие значения BID: 11, 20, 26, 45, 54, 51, 53 и 56.

После получения списка блоков DB2 может выполнить сокращенный реляционный просмотр каждого блока. При этом для каждого блока будет выполнена только одна операция ввода/вывода, так как блок хранится на диске в виде экстента, поэтому его можно считать в пул буферов за одну операцию. Предикаты требуется применить только к одной записи в каждом блоке, так как все записи блока гарантированно содержат одинаковые значения ключей измерений. Если присутствуют другие предикаты, их требуется проверить для других записей блока.

Для таблиц с многомерной кластеризацией можно создать обычные индексы, содержащие RID. Такие индексы можно объединить с блочными индексами путем выполнения логических операций AND и OR. С помощью советника по выбору индексов для таблицы с многомерной кластеризацией можно подобрать индексы, содержащие RID, однако нельзя подобрать измерения.

Во всех остальных отношениях таблицы с многомерной кластеризацией рассматриваются как обычные таблицы. Это означает, что для них можно определить триггеры, реляционные ограничения целостности, производные таблицы и материализованные таблицы запросов.

Многомерная кластеризация и разделы базы данных:

Многомерная кластеризация может применяться в многораздельной базе данных. Точнее, многомерная кластеризация может дополнять применение нескольких разделов базы данных. Для примера рассмотрим таблицу Sales, кластеризованную по измерениям YearAndMonth, Region и Product. Эту таблицу можно распределить по разделам базы данных, используя для этой цели поле Region (или любое другое поле). В случае поля Region все блоки для одного региона будут размещены в одном разделе базы данных. При этом в каждом разделе может храниться несколько значений региона. Каждый раздел будет содержать таблицу с многомерной кластеризацией, включающую те слои логического куба, которые соответствуют значениям поля Region, связанным с этим разделом. Например, слои South-central и Northwest могут находиться в одном разделе, а слои Southwest и Northeast - в другом. Кроме того, в каждом разделе будут храниться те блочные индексы, которые содержат значения ключей для блоков, расположенных в этом разделе.

Особенности загрузки таблиц с многомерной кластеризацией:

Если вы часто прокручиваете данные в хранилище данных, то для повышения эффективности этой операции можно воспользоваться таблицами с многомерной кластеризацией. При выполнении операции загрузки для таблицы с многомерной кластеризацией вначале используются пустые блоки таблицы, и лишь затем таблице выделяются новые блоки для остальных данных. После удаления набора данных, например, старых данных за месяц, можно воспользоваться утилитой загрузки для получения данных за следующий месяц. Эта утилита может использовать блоки, которые стали пустыми после принятой операции удаления.

Особенности ведения журнала для таблиц с многомерной кластеризацией:

Если столбцы, для которых ранее были созданы индексы RID, будут преобразованы в измерения (то есть для них будут созданы блочные индексы), то значительно упростится процедура обслуживания индексов и ведения журнала. DB2 потребуется удалять BID из блочных индексов и регистрировать эту операцию в журнале только при удалении последней записи из блока Аналогично, DB2 требуется вставлять BID в блочные индексы и регистрировать эту операцию в журнале только при вставке записи в новый блок (при вставке первой записи ячейки или при вставке в ячейку, все блоки которой заполнены). Поскольку блоки могут содержать от 2 до 256 страниц записей, затраты на обслуживание блочных индексов и регистрацию соответствующих операций в журнале будут сравнительно небольшими. При этом в журнал по прежнему будет заноситься информация обо всех операциях вставки и удаления, выполняемых для таблицы и для индексов RID.

Особенности блочных индексов, созданных для таблиц с многомерной кластеризацией:

При определении измерения таблицы с многомерной кластеризацией создается блочный индекс измерения. Кроме того, для всех определенных измерений создается составной блочный индекс. Если для таблицы с многомерной кластеризацией определено только одно измерение, DB2 создаст только один блочный индекс, который будет играть роль блочного индекса измерения и роль составного блочного индекса. Аналогично, при создании таблицы с многомерной кластеризацией, в которой определено измерение для столбца A и для столбцов (A, B), DB2 создаст блочный индекс измерения для столбца A и для столбцов A и B. Поскольку составной блочный индекс - это блочный индекс, содержащий все измерения таблицы, его роль в данном случае будет играть блочный индекс измерения для столбцов A и B.

Составной блочный индекс применяется при обработке запросов для доступа к данным таблицы, содержащим определенные значения измерений. Обратите внимание, что при выборе составного блочного индекса для обработки запроса

учитывается порядок его ключевых компонентов. Порядок ключевых компонентов определяется порядком столбцов, заданных во всем условии ORGANIZE BY DIMENSIONS при создании таблицы с многомерной кластеризацией. Например, если таблица была создана с помощью оператора `CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int) ORGANIZE BY DIMENSIONS (c1, c4, (c3,c1), c2)` то составной блочный индекс будет создан для столбцов (c1,c4,c3,c2). Хотя столбец c1 дважды указан в определении измерений, в составной блочный индекс соответствующий ключевой компонент будет входить только один раз. При этом применяется первое вхождение имени столбца в определение измерений. Порядок ключевых компонентов составного блочного индекса не влияет на обработку операций вставки, однако может влиять на обработку запросов. Следовательно, если нужно получить составной блочный индекс, содержащий столбцы в следующем порядке: (c1,c2,c3,c4), то таблица должна быть создана с помощью следующего оператора `CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int) ORGANIZE BY DIMENSIONS (c1, c2, (c3,c1), c4)`.

Понятия, связанные с данным:

- “Indexes” в книге *SQL Reference, Том 1*
- “Рекомендации по выбору измерений для таблиц с многомерной кластеризацией” на стр. 78
- “Управление таблицами и индексами для таблиц MDC” в книге *Руководство администратора: Производительность*
- “Стратегии оптимизации для таблиц MDC” в книге *Руководство администратора: Производительность*
- “Рекомендации по созданию таблиц с многомерной кластеризацией” на стр. 82

Ссылки, связанные с данной темой:

- “Режимы блокировки при просмотре таблиц и индексов RID для таблиц MDC” в книге *Руководство администратора: Производительность*
- “Блокировка при поблочном просмотре индексов таблиц MDC” в книге *Руководство администратора: Производительность*

Рекомендации по выбору измерений для таблиц с многомерной кластеризацией

Запросы, для которых рекомендуется применять многомерную кластеризацию:

Первое, что нужно сделать при выборе измерений для таблицы - определить, какие запросы выиграют от кластеризации на уровне блоков. Рекомендуется создать ключи измерений для тех столбцов, которые применяются в запросах, содержащих равенства и диапазоны, в особенности для столбцов с большой

мощностью. Кроме того, рекомендуется создать измерения для внешних ключей в таблице фактов с многомерной кластеризацией, участвующей в объединении типа ″звезда″ с таблицами измерений. Оцените, какой выигрыш в производительности можно получить за счет автоматической, непрерывной кластеризации по нескольким измерениям, а также кластеризации на уровне экстентов, или блоков.

Ниже приведены примеры запросов, производительность обработки которых повысится в случае применения многомерной кластеризации. Во всех примерах рассматривается таблица с многомерной кластеризацией t1 с измерениями c1, c2 и c3.

Пример 1:

```
SELECT .... FROM t1 WHERE c3 < 5000
```

Этот запрос содержит предикат диапазона для одного измерения, поэтому его можно внутренне преобразовать таким образом, чтобы для доступа к таблице применялся блочный индекс измерения c3. В индексе будут найдены идентификаторы блоков (BID), заданные в ключах, значения которых меньше 5000. Для получения необходимых записей будет выполнен сокращенный реляционный просмотр итогового набора блоков.

Пример 2:

```
SELECT .... FROM t1 WHERE c2 IN (1,2037)
```

Этот запрос содержит предикат IN для одного измерения, поэтому для выбора записей может применяться блочный индекс. Запрос можно внутренне преобразовать таким образом, чтобы для доступа к таблице применялся блочный индекс измерения c2. В индексе будут найдены BID тех ключей, значения которых лежат в диапазоне от 1 до 2037. Для получения необходимых записей будет выполнен сокращенный реляционный просмотр итогового набора блоков.

Пример 3:

```
SELECT .... FROM t1
   WHERE c2 > 100 AND c1 = '16/03/1999' AND c3 > 1000 and c3 < 5000
```

Этот запрос содержит предикаты диапазона для столбцов c2 и c3, предикат равенства для столбца c1 и операцию AND. Его можно внутренне преобразовать таким образом, чтобы для доступа к таблице применялись блочные индексы для отдельных измерений. В результате просмотра блочного индекса c2 будут найдены BID, связанные с ключами, значения которых больше 100. В результате просмотра блочного индекса c3 будут найдены BID, связанные с теми ключами, значения которых лежат в диапазоне от 1000 до 5000. В результате просмотра блочного индекса c1 будут найдены BID тех ключей, значение которых равно ’16/03/1999’. Для списков BID, полученных в результате просмотра индексов,

будет выполнена логическая операция AND, в результате которой будет найдено пересечение списков. Для получения необходимых записей будет выполнен сокращенный реляционный просмотр итогового набора блоков.

Пример 4:

```
SELECT .... FROM t1 WHERE c1 < 5000 OR c2 IN (1,2,3)
```

Этот запрос содержит предикат диапазона для измерения c1 и предикат IN для измерения c2, а также логическую операцию OR. Его можно внутренне преобразовать таким образом, чтобы для доступа к таблице применялись блочные индексы измерений c1 и c2. В результате просмотра блочного индекса измерения c1 будут найдены значения, не превышающие 5000, а в результате просмотра блочного индекса измерения c2 будут найдены значения 1, 2 и 3. Для списков BID, полученных в результате просмотра блочных индексов, будет выполнена логическая операция OR. После этого для получения необходимых записей будет выполнен сокращенный реляционный просмотр итогового набора блоков.

Пример 5:

```
SELECT .... FROM t1 WHERE c1 = 15 AND c4 < 12
```

Этот запрос содержит предикат равенства для измерения c1, предикат диапазона для столбца, который не является измерением, и логическую операцию AND. Его можно внутренне преобразовать таким образом, чтобы с помощью блочного индекса измерения c1 был получен список блоков из среза таблицы, в котором c1=15. Если для столбца c4 создан индекс RID, то путем просмотра этого индекса можно получить RID записей, в которых значение c4 меньше 12. После этого для итогового списка блоков и списка записей можно выполнить логическую операцию AND. В результате будут исключены те RID, которые отсутствуют в блоках, содержащих записи со значением c1, равным 15. Из таблицы будут получены только те RID, которые будут найдены в этих блоках.

Если для столбца c4 не создан индекс RID, то можно найти необходимые блоки в блочном индексе, а затем выполнить сокращенный реляционный просмотр каждого блока, применяя предикат ″c4 меньше 12″ к каждой записи.

Пример 6:

```
SELECT .... FROM t1 WHERE c1 < 5 OR c4 = 100
```

Этот запрос содержит предикат диапазона для измерения c1, предикат равенства для столбца c4, который не является измерением, а также операцию OR. Если для столбца c4 создан индекс RID, то запрос можно внутренне преобразовать таким образом, чтобы была выполнена логическая операция OR для блочного индекса измерения c1 и индекса RID столбца c4. Если для столбца c4 не создан индекс, то можно выполнить просмотр таблицы, в ходе которого потребуется

проверить все записи. Во время применения логической операции OR будет выполнен просмотр блочного индекса измерения c1, в ходе которого будут найдены значения меньше 4, а также просмотр индекса RID столбца c4, в ходе которого будут найдены значения, равные 100. Для каждого найденного блока будет выполнен сокращенный реляционный просмотр. Все записи из этих блоков будут добавлены в набор результатов, и, кроме того, будут получены все RID, расположенные вне этих блоков.

Пример 7:

```
SELECT .... FROM t1,d1,d2,d3
   WHERE t1.c1 = d1.c1 and d1.region = 'NY'
      AND t2.c2 = d2.c3 and d2.year='1994'
      AND t3.c3 = d3.c3 and d3.product='basketball'
```

Этот запрос содержит объединение типа ″звезда″. В данном примере t1 - это таблица фактов с внешними ключами c1, c2 и c3, соответствующими первичным ключам таблиц измерений d1, d2 и d3. Таблицы измерений не обязательно являются таблицами с многомерной кластеризацией. Region, year и product - это столбцы соответствующих таблиц измерений, для которых можно создать обычные или блочные индексы (если таблицы измерений являются таблицами с многомерной кластеризацией). При обращении к значениями c1, c2 и c3 таблицы фактов можно выполнить просмотр блочных индексов измерений, связанных с этими столбцами. После этого можно выполнить логическую операцию AND для полученных списков BID. Для получения необходимых записей нужно выполнить сокращенный реляционный просмотр блоков из итогового списка.

Плотность ячеек:

При создании таблицы с многомерной кластеризацией рекомендуется оценить ожидаемую плотность ячеек таблицы исходя из текущего и ожидаемого содержимого таблицы. Если исходить только из эффективности обработки запроса, то можно выбрать такой набор измерений, при котором возможное число ячеек таблицы будет очень велико, если учесть все возможные значения каждого измерения. Максимальное число ячеек таблицы представляет собой декартово произведение мощностей измерений. Например, если в качестве измерений таблицы выбраны столбцы Day, Region и Product, и в таблице хранятся данные за пять лет, то число различных ячеек таблицы будет составлять `1821 дней * 12 регионов * 5 продуктов = 109260`. Даже если ячейка хранит всего несколько записей, она будет занимать целый блок страниц. Если размер блока достаточно велик, то таблица будет занимать гораздо больше памяти, чем нужно для хранения данных.

Существует несколько вариантов решения этой проблемы. Во-первых, можно сократить размер блока, уменьшив размер экстента. В этом случае в блоке ячейки, содержащей всего несколько записей, меньше памяти будет расходоваться впустую. С другой стороны, ячейки, содержащие много записей,

будут занимать больше блоков. В результате возрастет число идентификаторов блоков (BID), связанных с такими ячейками в блочных индексах, что приведет к увеличению размера индексов и, в перспективе, к увеличению числа операций вставки и удаления данных из индекса. Последнее связано с тем, что мелкие блоки будут быстрее заполняться и опустошаться. Кроме того, большие ячейки будут представлены в виде большого числа небольших групп кластеризованных данных, а не в виде небольшого числа больших групп кластеризованных данных.

Во-вторых, для того чтобы сократить размер таблицы с многомерной кластеризацией можно уменьшить мощность одного или нескольких измерений, понизив степень детализации. Например, в предыдущем примере можно сохранить измерения Region и Product, но заменить измерение Day на измерение YearAndMonth. Мощность измерений YearAndMonth, Region и ProductThis составляет 60, 12 и 5, соответственно, поэтому возможное число ячеек будет равно 3600. В каждой ячейке будет храниться более широкий диапазон значений, поэтому менее вероятно, что такая ячейка будет содержать мало записей. Кроме того, следует учесть, какие предикаты чаще всего задаются для столбцов измерений, например, задают ли они условия на месяц, квартал или число. В зависимости от этого можно решить, следует ли изменять степень детализации измерения. Например, если большинство предикатов задают условие на число месяца, и таблица кластеризована по месяцам, то с помощью блочного индекса измерения YearAndMonth DB2® быстро определит, какие месяцы содержат указанные числа, и найдет связанные с этими месяцами блоки. Однако при просмотре этих блоков предикат, задающий условие на число месяца, потребуется применить к каждой записи. Если же таблица будет кластеризована по дням (в этом случае мощность измерения будет больше), то с помощью блочного индекса можно будет сразу найти блоки, в которых нужно выполнить просмотр, и предикат потребуется применить только к первой записи каждой найденной ячейки. В этом случае можно уменьшить плотность ячеек за счет другого измерения. Например, измерение Region, содержащее 12 разных значений, с помощью пользовательской функции можно сократить до списка West, North, South и East.

Понятия, связанные с данным:

- “Многомерная кластеризация” на стр. 68

Рекомендации по созданию таблиц с многомерной кластеризацией

Перемещение данных из существующей таблицы в таблицу с многомерной кластеризацией:

Для перемещения данных из существующей таблицы в таблицу с многомерной кластеризацией импортируйте данные, отбросьте исходную таблицу, создайте таблицу с многомерной кластеризацией (с помощью оператора CREATE TABLE с условием ORGANIZE BY DIMENSIONS) и загрузите в нее данные.

Таблицы с многомерной кластеризацией в табличных пространствах, управляемых системой:

По умолчанию в табличных пространствах, управляемых системой, файл расширяется путем добавления одной страницы. В случае многомерной кластеризации данные кластеризуются по блокам, размер которых совпадает с размером экстента табличного пространства. При расширении таблицы с многомерной кластеризацией к ней добавляется один блок страниц, или экстент. Следовательно, при хранении многомерных таблиц в табличном пространстве, управляемом системой, необходимо включить многостраничное размещение файлов. В этом случае файл будет сразу расширяться на требуемое число страниц, что позволит значительно повысить производительность.

Для того чтобы включить многостраничное размещение файлов, вызовите утилиту *db2empfa*. После того как многостраничное размещение файлов включено, выключить его невозможно.

Таблица с тремя измерениями:

Если вы знаете, какие предикаты часто используются в запросах, таблицу можно кластеризовать по столбцам, применяемым в этих предикатах, указав условие ORGANIZE BY DIMENSIONS.

Пример 1:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, c3, c4)
```

Таблица в примере 1 кластеризована по значениям трех столбцов, образующих логический куб (то есть, по трем измерениям). При обработке запросов таблицу можно разделить на слои по одному или нескольким измерениям, для того чтобы реляционные операции требовалось выполнять только для блоков из некоторых слоев или ячеек. Обратите внимание, что размер блока, то есть число страниц в блоке, совпадает с размером экстента таблицы.

Таблица с измерениями, включающими несколько столбцов:

Каждое измерение может состоять из одного или нескольких столбцов. Таким образом, можно создать таблицу, кластеризованную по измерению, содержащему два столбца.

Пример 2:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, (c3, c4))
```

В примере 2 таблица кластеризована по двум измерениям, c1 и (c3, c4). Следовательно, при обработке запроса таблица может быть логически

разделена на слои в измерении c1 или в составном измерении (c3, c4). У таблицы будет то же число блоков, что и у таблицы из примера 1, и два, а не три блочных индекса измерения. В примере 1 создается три блочных индекса измерения, по одному для каждого из столбцов c1, c3 и c4. В примере 2 создается только два блочных индекса измерения: один для столбца c1 и один для столбцов c3 и c4. Основное различие между этими таблицами состоит в том, что в примере 1 при обработке запросов, использующих только столбец c4, можно воспользоваться блочным индексом измерения c4, для того чтобы быстро найти те блоки, которые содержат необходимые данные. В примере 2 столбец c4 представляет собой второй компонент ключа блочного индекса измерения, поэтому обработка запросов, использующих только столбец c4, займет больше времени. Однако в этом примере DB2® будет хранить и обслуживать на один индекс меньше.

Таблица с измерениями, заданными в виде выражений со столбцами:

Измерения можно определить в виде выражений, включающих столбцы. Такая возможность может применяться для уменьшения степени детализации измерений, например, для замены адреса на географическое расположение или регион, либо для замены даты на неделю, месяц или год. Для уменьшения степени детализации измерений могут применяться генерируемые столбцы. Такой тип определения столбца позволяет создавать столбцы с помощью выражений, представляющих измерения. В примере 3 оператор создает таблицу, кластеризованную по одному из исходных столбцов и двум выражениям, включающим столбцы.

Пример 3:

```
CREATE TABLE T1(c1 DATE, c2 INT, c3 INT, c4 DOUBLE,
   c5 DOUBLE GENERATED ALWAYS AS (c3 + c4),
   c6 GENERATED ALWAYS AS (MONTH(C1))
      ORGANIZE BY DIMENSIONS (c2, c5, c6)
```

В примере 3 столбец c5 - это выражение, включающее столбцы c3 и c4, а столбец c6 уменьшает степень детализации столбца c1. Этот оператор создает таблицу, кластеризованную по значениям столбцов c2, c5 и c6.

Монотонность:

Выражение со столбцами может содержать любое допустимое выражение, в том числе пользовательскую функцию (UDF). Однако для обработки запросов, включающих диапазон значений, то есть для поиска диапазона значений в блочном индексе измерения, необходимо, чтобы выражение было монотонным. Измерения, заданные в виде немонотонных выражений, допускают применение только предикатов равенства. Примером немонотонной функции может служить функция MONTH( ), указанная в определении столбца c6 из примера 3. Если столбец c1 содержит дату, системное время или строковое представление

одного из этих значений, то функция возвращает целочисленные значения из диапазона 1-12. И хотя вывод функции детерминирован, он похож на вывод ступенчатой функции (т.е. циклический шаблон):

```
MONTH(date('99/01/05')) = 1
MONTH(date('99/02/08')) = 2
MONTH(date('99/03/24')) = 3
MONTH(date('99/04/30')) = 4
...
MONTH(date('99/12/09')) = 12
MONTH(date('00/01/18')) = 1
MONTH(date('00/02/24')) = 2
...
```

Следовательно, при увеличении даты в этом примере значение функции MONTH(date) не будет постоянно возрастать. Точнее, нельзя утверждать, что если date1 больше date2, то значение MONTH(date1) больше либо равно MONTH(date2). Если это условие не выполнено, то функция не является монотонной. Немонотонные выражения допустимы, однако в этом случае предикат диапазона, заданный для исходного столбца, нельзя преобразовать в эквивалентный предикат диапазона для соответствующего измерения. Тем не менее, предикаты диапазона для всего выражения допустимы, например, можно указать `where month(c1) between 4 and 6`. В этом случае индекс измерения может использоваться обычным образом, то есть из него будут выбраны ключи со значениями от 4 до 6.

Для того чтобы функция стала монотонной, необходимо добавить год к значению месяца. В DB2 предусмотрено расширение встроенной функции INTEGER для определения монотонных выражений, включающих дату. Функция INTEGER(date) возвращает целочисленное представление даты, с помощью которого можно определить год и месяц. Например, функция `INTEGER(date('2000/05/24'))` возвращает значение `20000524`, поэтому `INTEGER(date('2000/05/24'))/100 = 200005`. Функция INTEGER(date)/100 является монотонной.

Аналогично, для встроенных функций DECIMAL и BIGINT предусмотрены расширения, позволяющие создать монотонные функции. DECIMAL(timestamp) возвращает десятичное представление системного времени. Эту функцию можно использовать в монотонных выражениях для получения возрастающей последовательности значений, включающих месяц, день, час, минуты и т.д. BIGINT(date) возвращает представление даты в виде большого целого числа, аналогичное значению INTEGER(date).

При создании генерируемого столбца таблицы и при создании измерения с помощью выражения DB2 пытается определить, монотонно ли выражение. Некоторые функции, в том числе DATENUM(), DAYS() и YEAR(), сохраняют монотонность. Кроме того, монотонность сохраняют некоторые математические операции, в том числе деление, умножение и сложение столбцов

и констант. Если DB2 определит, что выражение не сохраняет монотонность, либо не сможет определить, монотонно ли выражение, то для базового столбца измерения будут допустимы только предикаты равенства.

Понятия, связанные с данным:

- "Export Overview" в книге *Data Movement Utilities Guide and Reference*
- "Многомерная кластеризация" на стр. 68
- "Рекомендации по выбору измерений для таблиц с многомерной кластеризацией" на стр. 78

Задачи, связанные с данной темой:

- "Определение измерений таблицы" в книге *Руководство администратора: Реализация*

Ссылки, связанные с данной темой:

- "CREATE TABLE statement" в книге *SQL Reference, Том 2*
- "db2empfa - Enable Multipage File Allocation Command" в книге *Command Reference*

Ограничения

Ограничение - это правило, которому подчиняется менеджер баз данных.

Существует три типа ограничений:

- *Ограничение уникальности* - это правило, запрещающее существование повторяющихся значений в одном или нескольких столбцах таблицы. Поддерживаемые ограничения уникальности включают ключи уникальности и первичные ключи. Ограничение уникальности может быть задано, например, для идентификатора поставщика в таблице поставщиков, чтобы один и тот же идентификатор не мог быть присвоен двум поставщикам.
- *Реляционное ограничение* - это логическое правило для значений в одном или нескольких столбцах одной или нескольких таблиц. Предположим, что в наборе таблиц содержит общую информацию о поставщиках компании. Время от времени имя какого-либо поставщика изменяется. В этом случае можно задать реляционное ограничение, согласно которому ID поставщика в таблице должен совпадать с ID поставщика в информации о поставщике. Это ограничение не позволяет выполнять операции вставки, изменения и удаления, которые могли бы привести к удалению информации о поставщике.
- *Проверочное ограничение таблицы* задает ограничения на данные, добавляемые в эту таблицу. Проверочное ограничение таблицы, например, может указывать, что размер заработной платы работника при добавлении или изменении информации о заработной плате в таблице, содержащей информацию о персонале, не должен быть меньше $20,000.

Реляционные и проверочные ограничения таблицы можно включать и выключать. Например, как правило, при загрузке большого объема данных в базу данных эти ограничения лучше отключить.

Ограничения уникальности

Ограничение уникальности - это правило, согласно которому значения ключей являются допустимыми, только если они уникальны в рамках таблицы. Ограничения уникальности необязательны и определяются в операторах CREATE TABLE или ALTER TABLE с помощью условий PRIMARY KEY или UNIQUE. Все столбцы, указанные в ограничении уникальности должны быть определены как NOT NULL. Для соблюдения уникальности ключей при внесении изменений в столбцы с ограничением уникальности менеджер баз данных применяет индекс уникальности.

Таблица может содержать любое число ограничений уникальности, но не больше одного ограничения уникальности, определенного в качестве первичного ключа. Кроме того, таблица не может иметь несколько ограничений уникальности на одном и том же наборе столбцов.

Ограничение уникальности, на которое указывает внешний ключ реляционного ограничения, называется *родительским ключом*.

При определении ограничения уникальности в операторе CREATE TABLE менеджер баз данных автоматически создает индекс уникальности и обозначает его как первичный или системный индекс уникальности.

Если ограничение уникальности определено в операторе ALTER TABLE и для того же набора столбцов существует индекс, то этот индекс становится системным индексом уникальности. Если такой индекс не существует, то менеджер баз данных автоматически создает индекс уникальности и обозначает его как первичный или системный индекс уникальности.

Учтите, что определение ограничения уникальности и создание индекса уникальности не одно и то же. Хотя оба эти понятия применяются для обеспечения уникальности, индекс уникальности допускает существование столбцов с пустыми значениями и, в общем случае, не может применяться в качестве родительского ключа.

Реляционные ограничения

Реляционная целостность - это состояние базы данных, в котором допустимы все значения всех внешних ключей. *Внешний ключ* - это столбец или набор столбцов таблицы, значения которых должны соответствовать по крайней мере одному значению первичного ключа или ключа уникальности в строке родительской таблицы. *referential constraint* - это правило, согласно которому значения внешнего ключа являются правильными только при выполнении одного из следующих условий:

- Они представляют собой существующие значения родительского ключа.
- Некоторые компоненты внешнего ключа имеют пустые значения.

Таблица, содержащая родительский ключ, называется *родительской таблицей* реляционного ограничения, а таблица, содержащая внешний ключ - *зависимой* от этой таблицы.

Реляционные ограничения необязательны и определяются в операторах CREATE TABLE и ALTER TABLE. Менеджер баз данных соблюдает реляционные ограничения при выполнении операторов INSERT, UPDATE, DELETE, ALTER TABLE, ADD CONSTRAINT и SET INTEGRITY.

Реляционные ограничения с правилом удаления или модификации RESTRICT применяются перед всеми остальными реляционными ограничениями. Реляционные ограничения с правилом удаления или модификации NO ACTION, в большинстве случаев, действуют так же, как RESTRICT.

Реляционные ограничения, проверочные ограничения и триггеры могут применяться в различных комбинациях.

В правилах реляционной целостности применяются следующие понятия и термины:

Родительский ключ
Первичный ключ или ключ уникальности реляционного ограничения.

Родительская строка
Строка, у которой есть хотя бы одна зависимая строка.

Родительская таблица (Parent table)
Таблица, содержащая родительский ключ реляционного ограничения. Таблица может быть родительской для произвольного числа реляционных ограничений. Одна таблица может одновременно выступать в качестве родительской в одном реляционном ограничении и в роли зависимой - в другом.

Зависимая таблица (Dependent table)
Таблица, определение которой содержит, по крайней мере, одно реляционное ограничение. Таблица может быть зависимой для произвольного числа реляционных ограничений. Одна таблица может одновременно выступать в качестве зависимой в одном реляционном ограничении и в роли родительской - в другом.

Таблица-потомок (Descendent table)
Таблица является потомком таблицы T, если она зависит от таблицы T или является потомком таблицы, зависящей от T.

Зависимая строка
Строка, у которой есть хотя бы одна родительская строка.

Строка-потомок

Строка является потомком строки r, если она зависит от строки r или является потомком таблицы, зависящей от r.

Реляционный цикл

Набор реляционных ограничений, в котором каждая таблица является дочерней для самой себя.

Автореферентная таблица

Таблица, являющаяся и родительской, и зависимой таблицей в одном и том же реляционном ограничении. Это ограничение называется *автореферентной связью*.

Автореферентная строка

Строка-родитель сама себе.

Правило вставки

Согласно правилу вставки реляционного ограничения, непустое значение вставки внешнего ключа должно совпадать с некоторым значением родительского ключа родительской таблицы. Значение составного ключа считается пустым, если все его компоненты пусты (равны null). Это правило накладывается при задании внешнего ключа.

Правило модификации

Правило модификации реляционного ограничения задается при определении реляционного ограничения. К правилам модификации относятся NO ACTION и RESTRICT. Правило модификации применяется при изменении строки родительской или зависимой таблицы.

В случае родительской строки, если обновляется значение в столбце родительского ключа, то применяются следующие правила:

- При правиле изменения RESTRICT изменение не производится, если какая-либо строка зависимой таблицы соответствует исходному значению ключа.
- Если после выполнения оператора UPDATE (но до применения триггеров AFTER) хотя бы у одной строки зависимой таблицы не будет соответствующего родительского ключа, при правиле изменения NO ACTION изменение не выполняется.

В случае зависимой строки, если указан внешний ключ, то подразумевается правило модификации NO ACTION. Правило NO ACTION означает, что после выполнения оператора изменения непустое значение изменения внешнего ключа должно совпадать с некоторым значением родительского ключа родительской таблицы.

Значение составного ключа считается пустым, если все его компоненты пусты (равны null).

Правило удаления

Правило удаления реляционного ограничения задается при определении реляционного ограничения. К правилам удаления относятся NO ACTION, RESTRICT, CASCADE и SET NULL. Правило SET NULL можно указывать, только если некоторые столбцы внешнего ключа допускают пустые значения.

Правило удаления реляционного ограничения применяется при удалении строки родительской таблицы. Более точно, это правило применяется в случае, если строка родительской таблицы является объектом операции удаления или распространенного удаления (ее определение приводится ниже), и у этой строки есть зависимые строки в зависимой таблице реляционного ограничения. Рассмотрим пример, в котором P является родительской таблицей, D - зависимой, а p - родительская строка, над которой выполняется операция удаления или распространенного удаления. Правило удаления действует следующим образом:

- Для правил RESTRICT и NO ACTION, возникает ошибка, и строки не удаляются.
- Для правила CASCADE, операция удаления распространяется на строки зависимые от p в таблице D.
- Для правила SET NULL, всем допускающим пустые значения столбцам внешнего ключа в каждой строке таблицы D, зависимой от p, присваивается пустое значение.

Каждому реляционному ограничению, в котором таблица выступает в роли родительской, соответствует собственное правило удаления, а результат операции удаления определяется набором всех применяемых правил удаления. Таким образом, строку нельзя удалить, если у нее есть зависимые строки в реляционном ограничении с правилом удаления RESTRICT или NO ACTION, либо операция удаления распространяется на любую из зависимых от нее строк, определенных в реляционном ограничении с правилом удаления RESTRICT или NO ACTION.

Удаление строки из родительской таблицы P затрагивает другие таблицы и может повлиять на строки в этих таблицах:

- Если таблица D зависит от P и действует правило удаления RESTRICT или NO ACTION, то таблица D участвует в операции, но остается без изменений.
- Если таблица D зависит от P и действует правило удаления SET NULL, то таблица D участвует в операции и строки таблицы D могут быть изменены в ходе ее выполнения.
- Если таблица D зависит от P и действует правило удаления CASCADE, то таблица D участвует в операции и строки таблицы D могут быть удалены в ходе ее выполнения.

При удалении строк таблицы D говорят, что операция удаления в таблице P распространяется на таблицу D. Если таблица D также является родительской, то описанные выше действия относятся и к ее зависимым таблицам.

Любая таблица, которая может быть вовлечена в операцию удаления в таблице P, называется *связанной по удалению* с таблицей P. Таким образом, таблица связана по удалению с таблицей P, если она зависит от этой таблицы или от таблицы, в которую передаются операции удаления из таблицы P.

Проверочные ограничения таблицы

Проверочное ограничение таблицы - это правило, которое задает допустимость значений в одном или нескольких столбцах каждой строки таблицы. Это ограничение необязательно и определяется в операторах CREATE TABLE и ALTER TABLE. Проверочные ограничения таблицы задаются в ограниченной форме условия поиска. Согласно одному из ограничений, имя столбца в проверочном ограничении таблицы T должно соответствовать столбцу в таблице T.

Таблица может содержать любое число проверочных ограничений. Выполнение проверочного ограничения таблицы обеспечивается путем применения его условия поиска ко всем вставляемым и изменяемым строкам. При возврате для любой строки результата ложь возникает ошибка.

Если для таблицы с данными в операторе ALTER TABLE определены проверочные ограничения, то существующие данные проверяются по новому условию перед завершением выполнения оператора ALTER TABLE. С помощью оператора SET INTEGRITY таблицу можно перевести в состояние *ожидания проверки*, что позволяет выполнить оператор ALTER TABLE без проверки данных.

Ссылки, связанные с данной темой:

- “SET INTEGRITY statement” в книге *SQL Reference, Том 2*
- “Interaction of triggers and constraints” в книге *SQL Reference, Том 1*

Триггеры

Триггер определяет набор действий, выполняемых в ответ на операцию вставки, изменения или удаления в указанной таблице. При выполнении такой операции SQL триггер *активируется*.

Триггеры необязательны и определяются в операторе CREATE TRIGGER.

Триггеры можно использовать с реляционными ограничениями и проверочными ограничениями для обеспечения соблюдения правил целостности данных.

Триггеры можно также использовать для изменения других таблиц, для автоматической генерации или изменения значений для вставляемых или изменяемых строк или для запуска функций, выполняющих такие операции, как создание и отправка предупреждений.

Триггеры позволяют задавать и обеспечивать соблюдение *переходных* экономических правил, влияющих на изменение состояния данных (например, заработная плата, которую можно увеличить не более, чем на 10 процентов).

Триггеры реализуют логические связи для применения экономических правил в базе данных. Это позволяет снять ответственность за соблюдение этих правил с прикладных программ. Централизованные правила, применяемые ко всем таблицам, облегчают обслуживание, так как при изменении этих логических правил не требуется вносить изменения в прикладные программы.

Для создания триггера необходимо задать следующие данные:

- *Рабочая таблица* - таблица, для которой определен триггер.
- *Событие триггера* определяет конкретный оператор SQL, вносящий изменения в рабочую таблицу. Событие может быть оператором вставки, изменения или удаления.
- *Время активации триггера* указывает, когда следует активизировать триггер - до или после события триггера.

Оператор, в результате которого активируется триггер, включает *набор изменяемых строк*. Это строки рабочей таблицы, над которыми выполняется операция вставки, изменения или удаления. *Кратность триггера* определяет, выполняются ли операции триггера только один раз для всего оператора или один раз для каждой изменяемой строки.

Действие триггера состоит из необязательного условия поиска и набора операторов SQL, выполняемых при активации триггера. Операторы SQL выполняются только в том случае, если поиск возвратил истинный результат. Если время активации триггера предшествует событию триггера, то действия триггера могут включать операторы, выполняющие выборку, задающие переходные переменные перехода и передающие сигналы SQLstate. Если событие триггера предшествует времени активации триггера, то действия триггера могут включать операторы, выполняющие выборку, вставку, изменение, удаление и передающие сигналы SQLstate.

Действие триггера может ссылаться на значения набора изменяемых строк с помощью *переменных перехода*. Переменные перехода применяют имена столбцов в рабочей таблице, дополненные заданным именем, которое обозначает, используется ли прежнее значение (до изменения) или новое значение (после изменения). Новое значение также можно изменить с помощью оператора SET Variable в триггерах BEFORE, вставки и изменения.

Другим средством обращения к значениям в наборе изменяемых строк являются *таблицы перехода*. В таблицах перехода также используются имена столбцов рабочей таблицы, но в них также задано имя, позволяющее рассматривать весь набор изменяемых строк как одну таблицу. Таблицы перехода могут применяться только в триггерах AFTER. Для прежних и новых значений можно определить отдельные таблицы перехода.

Для одной комбинации таблицы, события и времени активации можно создать несколько триггеров. Триггеры активируются в порядке их создания. Таким образом, триггер, созданный последним, активируется в последнюю очередь.

Активация триггера может вызвать *каскадное срабатывание триггеров*, которое происходит в результате активации триггера, выполняющего операторы SQL, активирующие другие триггеры или даже исходный триггер. Действие триггера может привести к изменениям в результате применения правил реляционной целостности для удаления, которые, в свою очередь, могут активировать дополнительные триггеры. При каскадном срабатывании может быть активирована цепочка триггеров и правил удаления реляционной целостности, в результате чего один оператор INSERT, UPDATE или DELETE может вызвать внесение значительных изменений в базу данных.

Дополнительные особенности структуры базы данных

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

В целях *аудита* может понадобиться записывать все изменения, вносимые в базу данных за определенный период. Например, можно обновлять таблицу аудита каждый раз, когда меняется заработная плата сотрудника. Изменения в эту таблицу можно вносить автоматически, если определен соответствующий триггер. Аудит можно также выполнять с помощью утилиты аудита DB2®.

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

Можно также использовать информацию *сводок*. Например, у вас может быть таблица, содержащая в себе всю информацию о ваших сотрудниках. Однако вам бы хотелось, чтобы эта информация была разбита по отдельным таблицам для

отделов или подразделений. В этом случае для каждого отдела или подразделения можно создать материализованную таблицу запроса, основанную на данных исходной таблицы.

Средства *защиты* тоже надо задавать в вашей структуре. Например, можно обеспечить доступ пользователя к определенным типам данных через таблицы защиты. Можно определить уровни доступа к различным типам данных и задать, кто имеет доступ к этим данным. Конфиденциальные данные, такие как данные о сотруднике и его зарплате, будут иметь строгие ограничения защиты.

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

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

Глава 5. Физическая структура базы данных

После создания логической структуры базы данных встают вопросы о физической среде, где будут находиться база данных и таблицы. Нужно понять, какие файлы будут созданы для поддержки базы данных и управления ей, определить, сколько места потребуется для хранения данных, а также решить, как использовать табличные пространства, необходимые для хранения данных.

Каталоги и файлы базы данных

При создании базы данных информация о ней, в том числе информация по умолчанию, сохраняется в иерархической структуре каталогов. Расположение такой структуры выбирается исходя из информации, указанной в команде CREATE DATABASE. Если при создании базы данных не был задан каталог или диск, то применяется каталог по умолчанию.

Рекомендуется явно объявлять, где вы хотите создать базу данных.

В каталоге, заданном в команде CREATE DATABASE, создается подкаталог с именем экземпляра. Это позволяет создавать разные экземпляры базы данных, задавая при этом один и тот же каталог. В подкаталоге с именем экземпляра создается подкаталог с именем NODE0000. В многораздельной базе данных этот подкаталог соответствует логическому разделу. В подкаталоге с именем узла создается подкаталог с именем SQL00001. Этот подкаталог получает имя, содержащее маркер базы данных, и представляет создаваемую базу данных. При создании первой базы данных ее объекты помещаются в каталог SQL00001. Подкаталоги всех последующих баз данных нумеруются по порядку: SQL00002 и т.д. Эти подкаталоги позволяют различать базы данных, созданные в экземпляре, внутри каталога, заданного в команде CREATE DATABASE.

Структура каталогов выглядит следующим образом:

```
<ваш_каталог>/<ваш_экземпляр>/NODE0000/SQL00001/
```

Команда CREATE DATABASE создает в каталоге базы данных следующие файлы.

- Файлы SQLBP.1 и SQLBP.2 содержат информацию о пуле буферов. Второй файл может применяться как резервная копия первого.
- Файлы SQLSPCS.1 и SQLSPCS.2 содержат информацию о табличном пространстве. Второй файл может применяться как резервная копия первого.
- Файл SQLDBCON содержит информацию о конфигурации базы данных. Не изменяйте этот файл. Для настройки параметров конфигурации

воспользуйтесь графическим интерфейсом Центра управления или операторами командной строки UPDATE DATABASE MANAGER CONFIGURATION и RESET DATABASE MANAGER CONFIGURATION.

- Файл хронологии DB2RHIST.ASC и его резервная копия DB2RHIST.BAK содержат хронологию операций резервного копирования, восстановления, загрузки данных в таблицы, реорганизации таблиц, изменения табличного пространства и прочих операций изменения базы данных.

  Файл DB2TSCHNG.HIS содержит хронологию операций изменения табличного пространства на уровне файла журнала. Для каждого файла журнала DB2TSCHG.HIS содержит информацию о связанном с ним табличном пространстве. Эта информация применяется во время восстановления табличных пространств для определения тех файлов журнала, которые требуется обработать. Содержимое обоих файлов хронологии можно просмотреть в любом текстовом редакторе.

- Файлы управления журналами, SQLOGCTL.LFH и SQLOGMIR.LFH, содержат информацию об активных журналах.

  При восстановлении информация этого файла помогает определить, с какого момента, зафиксированного в журнале, требуется начать восстановление. Подкаталог SQLOGDIR содержит действующие файлы журналов.

  Примечание: Следует задать для подкаталога журналов диск, отличный от дисков ваших данных. В таком случае проблемы с каким-либо диском затронут либо только данные, либо только журналы, но не то и другое одновременно. Кроме того, это дает существенный выигрыш в производительности, так как файлы журнала и контейнеры базы данных не конкурируют за перемещение одних и тех же головок дисков. Расположение подкаталога журналов можно изменить с помощью параметра конфигурации базы данных *newlogpath*.

- Файл SQLINSLK гарантирует, что база данных применяется только одним экземпляром менеджера баз данных.

Дополнительная информация о каталогах базы данных SMS

Подкаталоги SQLT* содержат табличные пространства, управляемые системой (SMS), которые необходимы для рабочей базы данных. По умолчанию создается три табличных пространства:

- Подкаталог SQLT0000.0 содержит табличное пространство каталога с таблицами системного каталога.
- Подкаталог SQLT0001.0 содержит временное табличное пространство по умолчанию.
- Подкаталог SQLT0002.0 содержит табличное пространство пользовательских данных по умолчанию.

В каждом подкаталоге или контейнере создается файл SQLTAG.NAM. Этот файл помечает подкаталог как используемый, чтобы в дальнейшем табличные пространства не создавались в этом подкаталоге.

Кроме того, в файлах SQL*.DAT хранится информация о различных таблицах, содержащихся в этом подкаталоге или контейнере. Вместо звездочки (*) указывается уникальный набор цифр, идентифицирующий таблицу. Для каждого файла SQL*.DAT создается один или несколько следующих файлов, в зависимости от типа таблицы, ее состояния реорганизации и наличия индексов и полей LOB и LONG:

- SQL*.BMP (для таблицы MDC содержит информацию о выделении блоков)
- SQL*.LF (содержит данные LONG VARCHAR или LONG VARGRAPHIC)
- SQL*.LB (содержит данные BLOB, CLOB или DBCLOB)
- SQL*.LBA (содержит информацию о занятой и свободной памяти для файлов SQL*.LB)
- SQL*.INX (содержит данные об индексе таблицы)
- SQL*.DTR (содержит временные данные для реорганизации файла SQL*.DAT)
- SQL*.LFR (содержит временные данные для реорганизации файла SQL*.LF)
- SQL*.RLB (содержит временные данные для реорганизации файла SQL*.LB)
- SQL*.RBA (содержит временные данные для реорганизации файла SQL*.LBA)

Понятия, связанные с данным:

- “Сравнение табличных пространств SMS и DMS” на стр. 146
- “Рекомендации по работе с устройствами DMS” в книге *Руководство администратора: Производительность*
- “Табличные пространства SMS” в книге *Руководство администратора: Производительность*
- “Табличные пространства DMS” в книге *Руководство администратора: Производительность*
- “Пример карты адресов табличного пространства DMS” в книге *Руководство администратора: Производительность*
- “Структура файла восстановления хронологии” в книге *Справочное руководство по восстановлению данных и высокой доступности*

Ссылки, связанные с данной темой:

- “CREATE DATABASE Command” в книге *Command Reference*

Размер объектов базы данных

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

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

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

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

Учитывайте эти утилиты при планировании физических требований баз данных.

При оценке размера базы данных следует также учитывать:

- Таблицы системного каталога
- Данные пользовательских таблиц
- Данные длинных полей
- Данные больших объектов (LOB)
- Индексное пространство
- Пространство файла журнала
- Временное рабочее пространство

Здесь не рассматриваются требования к дисковому пространству, которые предъявляют:

- Локальный файл каталога баз данных

- Системный файл каталога баз данных
- Дополнительное место, требуемое операционной системой для управления файлами, включая:
  - размер файлового блока
  - управляющее пространство каталога

Понятия, связанные с данным:

- “Размер таблиц системного каталога” на стр. 99
- “Размер данных пользовательских таблиц” на стр. 100
- “Размер данных длинных полей” на стр. 101
- “Размер данных больших объектов” на стр. 102
- “Необходимо пространство для индексов” на стр. 103
- “Размер файлов журнала” на стр. 106
- “Размер временных таблиц” на стр. 108

Ссылки, связанные с данной темой:

- “db2look - DB2 Statistics and DDL Extraction Tool Command” в книге *Command Reference*

Размер таблиц системного каталога

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

Размер пространства, выделяемого для таблиц каталога, зависит от типа табличного пространства и размера экстента табличного пространства, содержащего таблицы каталога. Например, если используется табличное пространство DMS с размером экстента 32, для табличного пространства каталога будет выделено 20 Мбайт.

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

Понятия, связанные с данным:

- “Размер объектов базы данных” на стр. 98
- “Определение таблиц системного каталога” в книге *Руководство администратора: Реализация*

Размер данных пользовательских таблиц

По умолчанию табличные данные хранятся на страницах по 4 Кбайта. Каждая страница (независимо от ее размера) содержит 68 дополнительных байтов для менеджера баз данных. Для хранения пользовательских данных (или строк) останется 4028 байт, хотя никакая строка на странице размером 4 Кбайта не может превышать 4005 байт в длину. Строка *не может* занимать несколько страниц. При использовании страниц размером 4 Кбайта допустимо не более 500 столбцов.

Страницы табличных данных *не* содержат данных для столбцов типов LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB и DBCLOB. В строках в странице табличных данных содержится дескриптор для этих столбцов.

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

При вызове оператора ALTER TABLE APPEND ON данные всегда добавляются в конец, и информация о свободном месте на страницах данных не сохраняется.

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

```
число_записей_на_страницу=ОКРУГЛИТЬ В МЕНЬШУЮ СТОРОНУ(4028/(средний размер строки+10))
```

и

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

где средний размер строки - сумма средних размеров столбцов, а множитель ″1,1″ нужен для учета дополнительного места.

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

Вы можете также создавать пулы буферов или табличные пространства с размером страницы 8 Кбайт, 16 Кбайт или 32 Кбайта. Размер страницы у всех таблиц, созданных в табличном пространстве, совпадает с размером страницы табличного пространства. Отдельный объект таблицы или индекса может достигать размера 512 Гбайт (при размере страницы 32 Кбайт). При использовании страниц размером 8 Кбайт, 16 Кбайт или 32 Кбайта допустимо

не более 1012 столбцов. При использовании страниц размером 4 Кбайта максимальное количество столбцов - 500. Максимальные длины строк также зависят от размера страницы:

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

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

Резервную копию нельзя восстановить с другим размером страницы.

Нельзя импортировать файлы данных IXF, содержащие больше 755 столбцов.

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

Понятия, связанные с данным:

- “Размер объектов базы данных” на стр. 98

Размер данных длинных полей

Данные длинных полей хранятся в отдельном табличном объекте, структура которого отличается от других типов данных.

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

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

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

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

Понятия, связанные с данным:

• “Размер объектов базы данных” на стр. 98

Размер данных больших объектов

Данные больших объектов хранятся в двух отдельных табличных объектах, структура которых отличается от остальных типов данных.

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

- **Объекты данных большого объекта**

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

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

Кбайт. Опция COMPACT может уменьшить производительность при добавлении данных в конец значений больших объектов.

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

- **Объекты размещения больших объектов**

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

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

Понятия, связанные с данным:

- “Размер объектов базы данных” на стр. 98

Ссылки, связанные с данной темой:

- “Large objects (LOBs)” в книге *SQL Reference, Том 1*

Необходимо пространство для индексов

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

```
(средний размер ключа индекса + 9) * количество строк * 2
```

где:

- ″Средний размер ключа индекса″ - это число байт в каждом столбце из ключа индекса. (При оценке среднего размера столбца для столбцов VARCHAR и VARGRAPHIC используйте текущий средний размер данных плюс два байта. Не используйте максимальный объявленный размер.)
- Множитель ″2″ отражает дополнительный расход места, например, на промежуточные страницы и на свободное место.

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

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

```
(средний размер ключа индекса + 9) * количество строк * 3.2
```

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

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

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

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

- Грубая оценка среднего числа ключей на конечную страницу:

```
(0,9 * (U - (M*2))) * (D + 1)
---------------------------
      K + 7 + (5 * D)
```

  где:

  – U - доступное место на странице, приблизительно равно размеру страницы минус 100. Для страницы размером 4096 U равно 3996.
  – M = U / (9 + *минимальныйРазмерКлюча*)
  – D = среднее число дубликатов на значение ключа
  – K = *среднийРазмерКлюча*

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

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

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

- Более точная оценка среднего числа ключей на конечную страницу:

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

где X - общее число строк в таблице.

Исходный размер индекса оценивается примерно так:

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

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

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

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

```
(0,9 * (U - (M*2))) * (D + 1)
----------------------------
      K + 13 + (9 * D)
```

где:

- U - доступное место на странице, приблизительно равно размеру страницы минус 100. Для страницы размером 4096 U равно 3996.
- D - среднее число дублей на значение ключа на промежуточных страницах (их будет гораздо меньше, чем на конечных страницах, и для упрощения вычислений можно подставить сюда 0).
- M = U / (9 + *минимальныйРазмерКлюча* для промежуточных страниц)
- K = *среднийРазмерКлюча* для промежуточных страниц

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

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

Число промежуточных страниц можно оценить следующим образом:

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

```
    P = P + Y
    Y = Y / N
   Z++
}
```

где:

– P - число страниц (изначально 0).

– L - число конечных страниц.

– N - число ключей для каждой промежуточной страницы.

– Y = L / N

– Z - количество уровней в дереве индекса (изначально 1).

Общее число страниц равно:

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

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

Необходимый для создания индекса объем места оценивается следующим образом:

```
T * размер_страницы
```

Понятия, связанные с данным:

- “Indexes” в книге *SQL Reference, Том 1*
- “Размер объектов базы данных” на стр. 98
- “Очистка и обслуживание индекса” в книге *Руководство администратора: Производительность*

Размер файлов журнала

Для файлов управления журналом необходимо 32 Кб дискового пространства.

Дополнительно потребуется, по крайней мере, пространство для хранения конфигурации активного журнала, размер которой можно вычислить по формуле:

```
(число_первичных + число_вторичных) * (размер_файла_журнала + 2 ) * 4096
```

где:

- *число_первичных* - число первичных файлов журнала, определенное в файле конфигурации базы данных
- *число_вторичных* - число вторичных файлов журнала, заданное в файле конфигурации базы данных; в этой формуле *число_вторичных* не может быть

равно *-1* (Если *число_вторичных равно* -1, то вычисляется значение для нециклического ведения активного журнала.)

- *размер_файла_журнала* - число страниц в каждом файле журнала, определенное в файле конфигурации базы данных
- 2 - число страниц заголовка, требуемых для каждого файла журнала
- 4096 - число байтов на одной странице.

Если в базе данных включено циклическое ведение журнала, то эта формула позволит найти необходимый объем дискового пространства.

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

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

Если в базе данных настроен нециклических журнал (то есть, параметру *число_вторичных* присвоено значение *-1*), то для применения тех же размеров дискового пространства необходимо включить параметр конфигурации *userexit*. DB2® будет сохранять в каталоге журнала число активных журналов, не превышающее значение *число_первичных*, поэтому не следует применять в указанной выше формуле значение -1 для параметра *число_вторичных*. Зарезервируйте дополнительное дисковое пространство для задержки, которая будет вызвана архивацией файлов журнала.

Если у каталога журнала есть зеркальная пара, то необходимый объем дискового пространства для файла журнала следует удвоить.

Понятия, связанные с данным:

- “Размер объектов базы данных” на стр. 98
- “Структура журналов восстановления” в книге *Справочное руководство по восстановлению данных и высокой доступности*
- “Отображение журнала” в книге *Справочное руководство по восстановлению данных и высокой доступности*

Ссылки, связанные с данной темой:

- “Параметр конфигурации Размер файлов журнала - logfilsiz” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Число первичных файлов журнала - logprimary” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Число вторичных файлов журнала - logsecond” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Путь зеркальной копии журналов - mirrorlogpath” в книге *Руководство администратора: Производительность*

Размер временных таблиц

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

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

Понятия, связанные с данным:

- “Размер объектов базы данных” на стр. 98

Ссылки, связанные с данной темой:

- “sqlbmtsq - Table Space Query” в книге *Administrative API Reference*

Группы разделов базы данных

Группа разделов базы данных - это набор из одного или нескольких разделов базы данных. При создании таблиц для базы данных сначала вы создаете группу разделов базы данных, где будут храниться табличные пространства, а затем создаете табличное пространство, где будут храниться таблицы.

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

На рис. 32 на стр. 109 показан пример базы данных с пятью разделами, в которой:

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

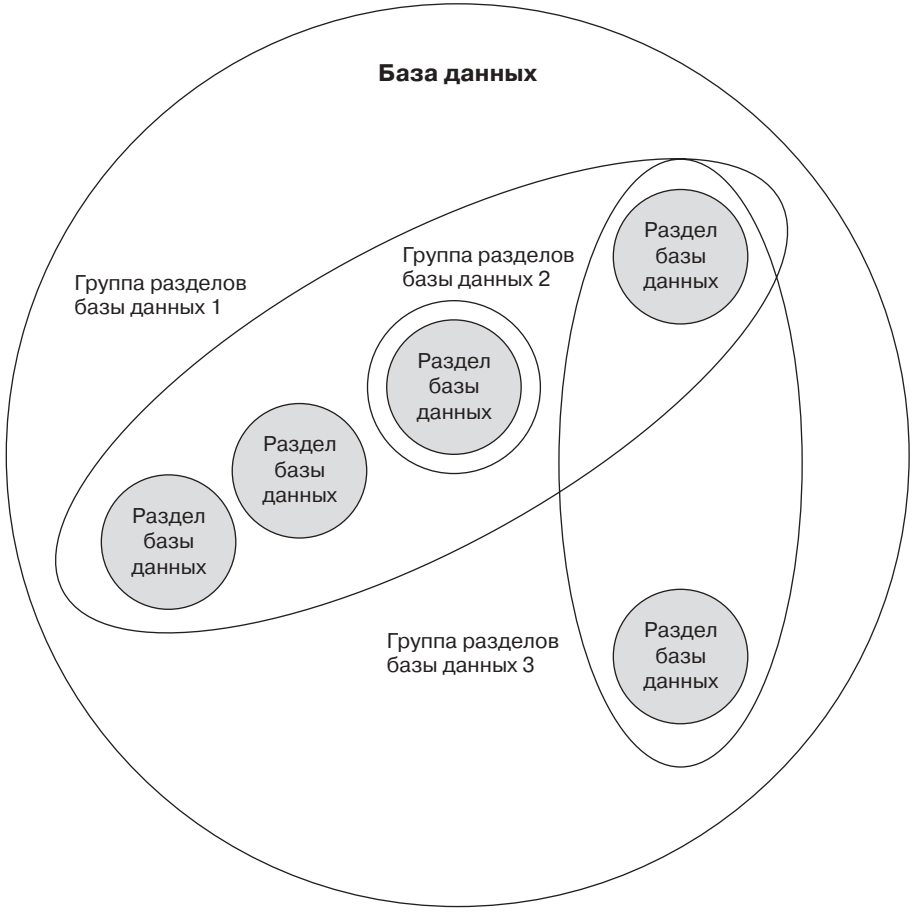

Рисунок 32. Группы разделов в базе данных

Создать новую группу разделов базы данных позволяет оператор CREATE DATABASE PARTITION GROUP. Для ее изменения предназначен оператор

ALTER DATABASE PARTITION GROUP. Данные распределены по всем разделам в группе разделов базы данных, причем можно добавить или отбросить один или несколько разделов базы данных. При использовании многораздельных групп разделов базы данных следует учитывать некоторые особенности их разработки.

Все разделы базы данных, являющиеся частью конфигурации системы базы данных, должны быть заранее определены в *файле конфигурации разделов*, который называется `db2nodes.cfg`. Группа разделов базы данных может содержать от одного раздела базы данных до всего множества разделов базы данных, определенных для системы базы данных.

При создании или изменении группы разделов базы данных с ней связывается *карта разделения*. Карта разделения совместно с *ключом разделения* и алгоритмом хеширования используется менеджером баз данных для определения, на каком разделе базы данных в группе разделов базы данных будет храниться заданная строка данных.

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

Понятия, связанные с данным:

- “Проектирование групп разделов базы данных” на стр. 111
- “Карты разделения” на стр. 112
- “Ключи разделения” на стр. 113

Ссылки, связанные с данной темой:

- “ALTER DATABASE PARTITION GROUP statement” в книге *SQL Reference, Том 2*
- “CREATE DATABASE PARTITION GROUP statement” в книге *SQL Reference, Том 2*

Проектирование групп разделов базы данных

При использовании одноразделной базы данных нет необходимости проектировать группы разделов базы данных.

Если вы используете многораздельную группу разделов базы данных, учтите следующие особенности проектирования:

- В многораздельной группе разделов базы данных индекс уникальности можно создать, только если он является надмножеством ключа разделения.
- В зависимости от числа разделов в базе данных может быть одна или несколько одноразделных групп разделов базы данных и одна или несколько многораздельных групп разделов базы данных.
- Каждому разделу базы данных должен быть назначен уникальный номер раздела. Один раздел базы данных может находиться в одной или в нескольких групп разделов базы данных.
- Для быстрого восстановления раздела базы данных, содержащего таблицы системного каталога, не размещайте на том же разделе базы данных пользовательские таблицы. Этого можно достичь, поместив пользовательские таблицы в группы разделов базы данных, в которые не включен раздел базы данных в группе разделов базы данных IBMCATGROUP.

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

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

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

Понятия, связанные с данным:

- “Группы разделов базы данных” на стр. 108
- “Карты разделения” на стр. 112
- “Ключи разделения” на стр. 113
- “Совместное размещение таблиц” на стр. 116
- “Совместимость с разделами” на стр. 116
- “Реплицируемые материализованные таблицы запросов” на стр. 117

Карты разделения

В среде многораздельной базы данных менеджер баз данных должен иметь способ узнавать, какие строки данных хранятся в каком из разделов базы данных. Менеджер баз данных должен знать, где найти требуемые данные; для этого он использует карту, которая называется *картой разделения*.

Карта разделения - это создаваемый внутри программы массив, содержащий 4096 элементов для многораздельных групп разделов базы данных или один элемент для одноразделных групп разделов базы данных. Для одноразделной группы разделов базы данных в карте разделения есть только один элемент, содержащий номер раздела базы данных, в котором хранятся все строки таблицы базы данных. Для многораздельных групп разделов базы данных циклически задаются номера разделов группы разделов базы данных. Как план города разделяется сеткой на части, так и менеджер баз данных при помощи *ключа разделения* определяет место (раздел базы данных), где хранятся данные.

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

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

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

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

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

На рис. 33 на стр. 113 показана строка со значением ключа разделения (c1, c2, c3), отображаемая в раздел 2, который, в свою очередь, указывает на раздел базы данных n5.

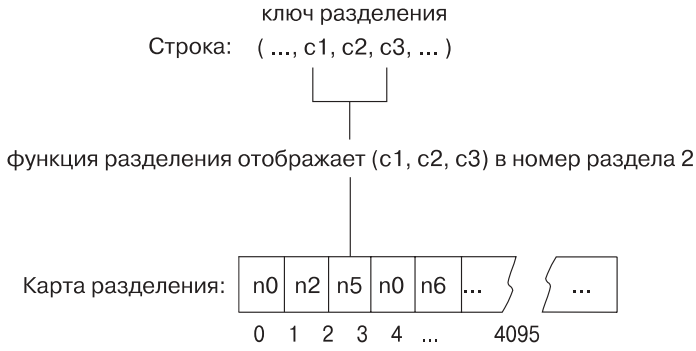

Рисунок 33. Распределение данных при помощи карты разделения

Карта разделения реализует гибкий способ управления хранением данных в многораздельной базе данных. Если в будущем вам потребуется изменить в базе данных распределение данных по разделам базы данных, воспользуйтесь утилитой перераспределения данных. Эта утилита позволяет перебалансировать распределение данных или внести в него асимметрию.

Чтобы получить для просмотра копию карты разделения, можно воспользоваться API получения информации о разделении таблицы (**sqlugtpi**).

Понятия, связанные с данным:

- “Группы разделов базы данных” на стр. 108
- “Проектирование групп разделов базы данных” на стр. 111
- “Ключи разделения” на стр. 113

Ссылки, связанные с данной темой:

- “sqlugtpi - Get Table Partitioning Information” в книге *Administrative API Reference*

Ключи разделения

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

пространстве, связанном с одноразделной группой разделов базы данных, и вам нужен ключ разделения, вы должны задать его в явном виде. Он не будет создан по умолчанию.

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

Важно выбрать правильный ключ разделения. При этом необходимо учесть:

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

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

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

Это гарантирует, что строки совместно размещенных таблиц с одинаковыми значениями ключа разделения расположены на одном и том же разделе.

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

При определении ключей разделения необходимо учитывать следующие особенности:

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

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

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

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

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

Понятия, связанные с данным:

- “Группы разделов базы данных” на стр. 108
- “Проектирование групп разделов базы данных” на стр. 111
- “Карты разделения” на стр. 112

Ссылки, связанные с данной темой:

- “ALTER TABLE statement” в книге *SQL Reference, Том 2*

Совместное размещение таблиц

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

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

При доступе к нескольким таблицам для объединения или подзапроса DB2® может распознать, что объединяемые данные находятся на одном и том же разделе базы данных. Если это происходит, DB2 может провести объединение или подзапрос на том разделе базы данных, где хранятся данные, вместо того, чтобы перемещать данные между разделами базы данных. Эта возможность проводить объединения и подзапросы на том же разделе базы данных дает значительное улучшение производительности.

Понятия, связанные с данным:

- “Группы разделов базы данных” на стр. 108
- “Проектирование групп разделов базы данных” на стр. 111
- “Ключи разделения” на стр. 113
- “Совместимость с разделами” на стр. 116

Совместимость с разделами

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

Совместимость по разделу обладает следующими свойствами:

- Базовый тип данных совместим с пользовательскими типами, производными от него.

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

Понятия, связанные с данным:

- “Группы разделов базы данных” на стр. 108
- “Проектирование групп разделов базы данных” на стр. 111
- “Ключи разделения” на стр. 113

Реплицируемые материализованные таблицы запросов

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

В среде многораздельных баз данных материализованные таблицы запроса можно реплицировать. *Реплицируемые материализованные таблицы запросов* можно использовать для улучшения производительности запросов. Реплицируемая материализованная таблица запроса основана на таблице, которая может быть создана в одноразделной группе разделов базы данных, но вы хотите реплицировать ее по всем разделам базы данных в этой группе

разделов. Чтобы создать реплицируемую материализованную таблицу запроса, укажите в операторе CREATE TABLE ключевое слово REPLICATED.

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

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

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

Понятия, связанные с данным:

- “Проектирование групп разделов базы данных” на стр. 111

Задачи, связанные с данной темой:

- “Создание материализованной таблицы запроса” в книге *Руководство администратора: Реализация*

Ссылки, связанные с данной темой:

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

Проектирование табличного пространства

Табличное пространства представляет собой структуру для хранения таблиц, индексов, больших объектов и длинных данных. Табличные пространства находятся в группах разделов баз данных. Они позволяют напрямую назначать контейнерам местонахождение базы данных и табличных данных. (Контейнером может быть каталог, устройство или файл.) Это может улучшить производительность, а также сделать конфигурацию более гибкой.

Поскольку табличные пространства размещаются в группах разделов базы данных, выбранное для хранения таблицы табличное пространство определяет, как данные для этой таблицы распределяются по разделам базы данных в данной группе разделов базы данных. Одно табличное пространство может

содержать несколько контейнеров. Можно создавать несколько контейнеров (от одного или нескольких табличных пространств) на одном и том же физическом диске. Для улучшения производительности каждый контейнер должен находиться на отдельном диске. На рис. 34 показано, как таблицы и табличные пространства в базе данных связаны с контейнерами, относящимися к этой базе данных.

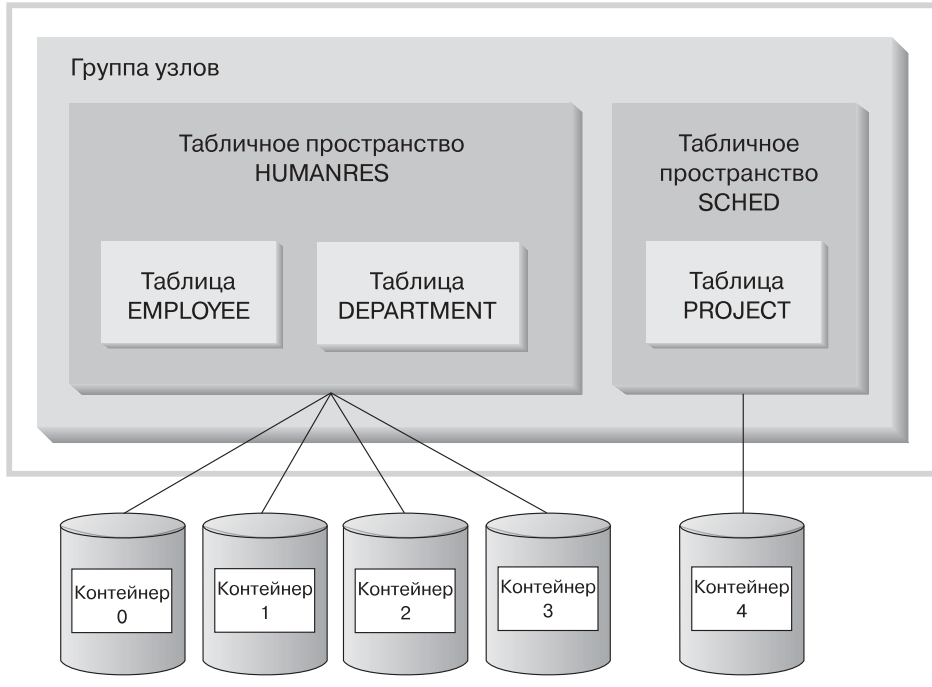

Рисунок 34. Табличные пространства и таблицы в базе данных

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

Менеджер баз данных пытается уравнять загрузку контейнеров данными. В результате для хранения данных используются все контейнеры. Число страниц, записываемых менеджером баз данных в контейнер перед использованием другого, называется *размером экстента*. Менеджер баз данных не обязательно начинает запись данных таблицы с первого контейнера.

На рис. 35 на стр. 120 показано табличное пространство HUMANRES с размером экстента, равным двум страницам по 4 Кбайта, и четыре контейнера, в каждом из которых выделено небольшое количество экстентов. В таблицах

DEPARTMENT и EMPLOYEE по семь страниц; обе они занимают все четыре контейнера.

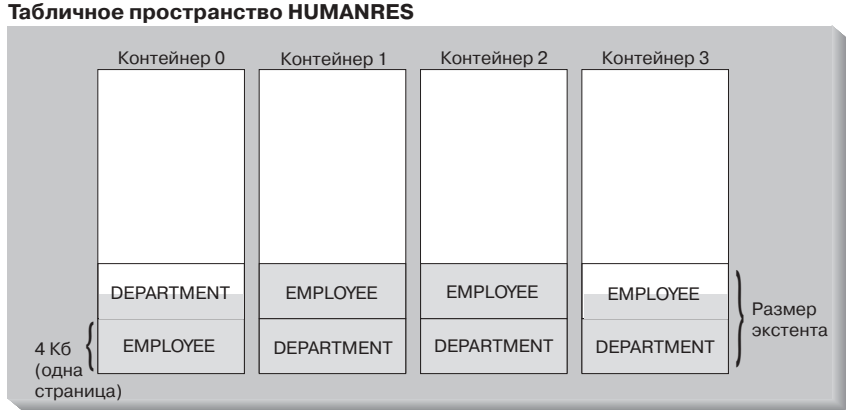

Рисунок 35. Контейнеры и экстенты

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

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

  При создании таблицы необходимо указывать имя табличного пространства, иначе результаты могут отличаться от ожидаемых.

  Размер страницы таблицы определяется либо размером строки, либо количеством столбцов. Максимально допустимая длина строки зависит от размера страницы табличного пространства, в котором создается таблица. Возможные значения размера страницы - 4 Кбайта (по умолчанию), 8 Кбайт, 16 Кбайт и 32 Кбайта. Можно использовать табличное пространство с одним размером страницы для основной таблицы и табличное пространство с другим размером страницы для длинных данных и данных больших объектов. (Напомним, что SMS не поддерживает таблиц, занимающих несколько табличных пространств, а DMS поддерживает.) Если количество столбцов или размер строки превышает пределы для размера страницы табличного пространства, возвращается ошибка (SQLSTATE 42997).
- Одно или несколько *временных табличных пространств*, содержащих временные таблицы. Временные табличные пространства делятся на *системные временные табличные пространства* и *пользовательские временные табличные пространства*. В базе данных должно быть по меньшей мере одно

системное временное табличное пространство; по умолчанию при создании базы данных создается одно системное временное табличное пространство под именем TEMPSPACE1. IBMTEMPGROUP - группа разделов базы данных по умолчанию для этого табличного пространства. Пользовательские временные табличные пространства при создании базы данных по умолчанию *не* создаются.

Если база данных использует несколько временных табличных пространств и нужен новый временный объект, оптимизатор выберет для этого объекта подходящий размер страницы. Объект будет размещен во временном табличном пространство с подходящим размером страницы. Если есть несколько временных табличных пространств с таким размером страницы, табличные пространства будут выбираться циклически. В большинстве случаев, не рекомендуется настраивать временные табличные пространства с одинаковым размером страниц.

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

Нужно определить одно временное табличное пространство SMS с размером страницы, совпадающим с размером страницы большинства пользовательских табличных пространств. Этого должно быть достаточно для типичных состояний среды и рабочей нагрузки.

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

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

- Управляемое системой пространство, в котором менеджер файлов операционной системы управляет пространством хранения.
- Управляемое базой данных пространство, в котором менеджер баз данных управляет пространством хранения.

Понятия, связанные с данным:

- “Table spaces and other storage structures” в книге *SQL Reference, Том 1*
- “Пространство, управляемое системой” на стр. 122

- “Пространство, управляемое базой данных” на стр. 125
- “Сравнение табличных пространств SMS и DMS” на стр. 146
- “Ввод-вывод на диск для табличного пространства” на стр. 147
- “Сведения о рабочей нагрузке при разработке табличных пространств” на стр. 150
- “Размер экстента” на стр. 151
- “Отношения между табличными пространствами и пулами буферов” на стр. 153
- “Отношения между табличными пространствами и группами разделов баз данных” на стр. 154
- “Проектирование временных табличных пространств” на стр. 155
- “Проектирование табличного пространства каталога” на стр. 156

Задачи, связанные с данной темой:

- “Создание табличного пространства” в книге *Руководство администратора: Реализация*
- “Оптимизация производительности табличного пространства при размещении данных на дисковом массиве” на стр. 157

Ссылки, связанные с данной темой:

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

Пространство, управляемое системой

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

С каждой таблицей связан по меньшей мере один физический файл SMS.

В табличном пространстве SMS при увеличении объекта соответствующий файл увеличивается по одной странице за раз. Если вам нужно улучшить производительность вставки, возможно, следует разрешить многостраничное размещение файлов. Это позволяет системе размещать или увеличивать файл

больше, чем по одной странице за раз. Если многомерные таблицы (MDC) будут храниться в табличном пространстве SMS, то для повышения производительности, необходимо включить функцию многостраничного размещения файлов. Чтобы разрешить многостраничное размещение файлов, запустите **db2empfa**. В среде многораздельной базы данных эту утилиту нужно запустить на всех разделах базы данных. После того как многостраничное размещение файлов включено, выключить его невозможно.

Табличные пространства SMS определяются с помощью опции MANAGED BY SYSTEM команды CREATE DATABASE или оператора CREATE TABLESPACE. При проектировании табличных пространств SMS следует учитывать два ключевых фактора:

- Контейнеры для табличного пространства.

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

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

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

  Эта формула предполагает, что каждому контейнеру назначена отдельная файловая система и что в каждой файловой системе свободно максимальное количество места. Практически это не обязательно так, и максимальный размер табличного пространства может быть гораздо меньше. Кроме того, существуют ограничения SQL на размер объектов базы данных, влияющие на максимальный размер табличного пространства.

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

- Размер экстента для табличного пространства.

  Размер экстента можно задать только при создании табличного пространства. Поскольку размер экстента нельзя изменить позже, важно выбрать для него подходящее значение.

  Если при создании табличного пространства не указать размера экстента, менеджер баз данных создаст табличное пространство с размером экстента по умолчанию, заданным параметром конфигурации базы данных *dft_extent_sz*.

Этот параметр конфигурации первоначально задается на основе информации, заданной при создании базы данных. Если в команде CREATE DATABASE не указан параметр *dft_extent_sz*, размер экстента будет по умолчанию равен 32.

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

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

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

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

- Как менеджер баз данных управляет файлами данных и контейнерами, связанными с табличным пространством.

  Первый файл табличных данных (`SQL00001.DAT`) создается в первом контейнере, указанном для табличного пространства, и этот файл может расти, пока не достигнет размера экстента. Когда он достигает этого размера, менеджер баз данных начинает записывать данные в `SQL00001.DAT` в следующем контейнере. Этот процесс продолжается, пока файлы `SQL00001.DAT` не появятся во всех контейнерах, после чего менеджер баз данных вернется к первому контейнеру. Этот процесс (называющийся *чередованием*) продолжает применяться к каталогам контейнеров, пока какой-нибудь контейнер не заполнится (SQL0289N) или операционная система не откажется выделять дополнительное место (ошибка переполнения диска). Чередование используется также для файлов индексов (`SQL`*`nnnnn`*`.INX`), длинных полей (`SQL`*`nnnnn`*`.LF`) и LOB (`SQL`*`nnnnn`*`.LB` и `SQL`*`nnnnn`*`.LBA`).

  Примечание: Табличное пространство SMS заполнено, когда заполнен хотя бы один из его контейнеров. Следовательно, важно выделять для всех контейнеров одинаковое место.

  Чтобы данные распределялись по контейнерам более равномерно, менеджер баз данных определяет, с какого контейнера начать, взяв остаток от деления идентификатора таблицы (в последнем примере - 1) на количество контейнеров. Контейнеры нумеруются последовательно, начиная с 0.

Понятия, связанные с данным:

- “Проектирование табличного пространства” на стр. 118
- “Пространство, управляемое базой данных” на стр. 125
- “Сравнение табличных пространств SMS и DMS” на стр. 146

Ссылки, связанные с данной темой:

- “db2empfa - Enable Multipage File Allocation Command” в книге *Command Reference*

Пространство, управляемое базой данных

В табличном пространстве DMS (Database Managed Space - пространство, управляемое базой данных) пространство хранения управляется менеджером баз данных. Эта модель хранения состоит из ограниченного числа устройств или файлов, пространство которых управляется DB2. Администратор базы данных решает, какие устройства и файлы использовать, и DB2® управляет пространством на этих устройствах и файлах. В сущности, такое табличное пространство - реализация файловой системы специального назначения, разработанной для наилучшего соответствия нуждам менеджера баз данных.

Табличное пространство DMS, содержащие пользовательские таблицы, можно определить, как:

- *Обычное* табличное пространство для хранения данных таблиц и, при необходимости, индексов
- *Большое* табличное пространство для хранения данных длинных полей, больших объектов или индексов.

При проектировании табличных пространств и контейнеров DMS необходимо учитывать следующее:

- Менеджер баз данных использует чередование для равномерного распределения данных по всем контейнерам.
- Максимальный размер обычных табличных пространств - 64 Гбайта для страниц по 4 Кбайта; 128 Гбайт для страниц по 8 Кбайт; 256 Гбайт для страниц по 16 Кбайт; и 512 Гбайт для страниц по 32 Кбайта. Максимальный размер больших табличных пространств - 2 Тбайта.
- В отличие от табличных пространств SMS, контейнеры, из которых состоит табличное пространство DMS, могут быть разного размера, хотя это и не рекомендуется, так как это вызывает неравномерность при чередовании между контейнерами и плохо влияет на производительность. Если контейнер заполнен, табличные пространства DMS используют свободное пространство других контейнеров.
- Поскольку место выделяется заранее, его должно быть достаточно на момент создания табличного пространства. При использовании контейнеров-устройств для определения контейнера необходимо, чтобы существовало достаточно вместительное устройство. На каждом устройстве можно определить только один контейнер. Чтобы избежать потерь пространства, размер устройства должен быть равен размеру контейнера. Например, если на устройстве выделено 5000 страниц, а при определении контейнера выделяется 3000 страниц, 2000 страниц этого устройства нельзя будет использовать.

- По умолчанию, один экстент в каждом контейнере резервируется для дополнительных данных. Используются только целые экстенты, так что для наилучшего управления пространством можно при выделении контейнера определять подходящий размер при помощи следующей формулы:

  ```
  размер_экстента * (n + 1)
  ```

  где *размер_экстента* - размер каждого экстента в этом табличном пространстве, а *n* - число экстентов, которое вы хотите хранить в этом контейнере.
- Минимальный размер табличного пространства DMS - пять экстентов. При попытке создать табличное пространство размером меньше пяти экстентов возникнет ошибка (SQL1422N).
  - В табличном пространстве резервируются три дополнительных экстента.
  - Для хранения любых данных пользовательских таблиц требуются по меньшей мере два экстента. (Эти экстенты нужны для обычных данных таблицы, а не для данных индексов, длинных полей или больших объектов, для которых нужны отдельные экстенты.)
- Контейнеры-устройства должны использовать логические тома с ″символьным специальным интерфейсом″, а не физические тома.
- Для табличных пространств DMS можно использовать файлы вместо устройств. Между файлом и устройством нет функциональной разницы, но файл может быть менее эффективным из-за дополнительного расхода времени при работе через файловую систему. Файлы полезны, если:
  - Устройства не поддерживаются напрямую
  - Устройство недоступно
  - Не требуется максимальная производительность
  - Вы не хотите заниматься конфигурированием устройств.
- Если в рабочую нагрузку входят данные больших объектов или LONG VARCHAR, улучшение производительности можно получить при кэшировании файловой системы. Обратите внимание на то, что большие объекты и LONG VARCHAR не буферизуются пулом буферов DB2.
- Некоторые операционные системы допускают использование физических устройств размером более 2 Гбайт. Возможно, следует разделить такое физическое устройство на несколько логических устройств, чтобы ни один контейнер не превышал размера, допустимого операционной системой.

Понятия, связанные с данным:

- “Проектирование табличного пространства” на стр. 118
- “Пространство, управляемое системой” на стр. 122
- “Сравнение табличных пространств SMS и DMS” на стр. 146
- “Карты табличных пространств” на стр. 127

- “Добавление контейнеров в табличное пространство DMS и их расширение” на стр. 131

Карты табличных пространств

Картой табличного пространства называется внутреннее представление табличного пространства DMS в DB2, которое описывает связь между логическим и физическим расположением страниц в табличном пространстве. Ниже приведена информация о том, для чего предназначена карта табличного пространства, и каким образом она создается.

В базе данных DB2® страницам табличного пространства DMS присвоены логические номера от 0 до (N-1), где N - число доступных страниц в табличном пространстве.

Страницы табличного пространства DMS объединяются в экстенты, и с точки зрения управления табличным пространством память для объектов выделяется в виде экстентов. Это означает, что таблица может занимать только половину страниц экстента, однако будет считаться, что весь экстент используется этой таблицей. По умолчанию в одном экстенте хранится тег контейнера. Страницы из этого экстента не применяются для хранения данных. Однако если включена переменная реестра DB2_USE_PAGE_CONTAINER_TAG, то тег контейнера занимает только одну страницу.

Поскольку память в контейнерах выделяется в виде экстентов, те страницы, которые нельзя объединить в целый экстент, не применяются. Например, если контейнер содержит 205 страниц, а размер экстента равен 10, то в одном экстенте будет хранится тег, 19 экстентов будут доступны для хранения данных, а пять оставшихся страниц не будут использоваться.

Если табличное пространство DMS содержит один контейнер, логический номер страницы напрямую преобразуется в физическое расположение на диске, то есть страницы с номерами 0, 1, 2 располагаются на диске в том же порядке.

Такое преобразование просто выполнить и в том случае, если табличное пространство содержит несколько контейнеров одинакового размера. Первый экстент табличного пространства (содержащий страницы от 0 до (размер экстента - 1)) размещается в первом контейнере, второй экстент - во втором контейнере и т.д. После размещения экстента в последнем контейнере процесс повторяется, начиная с первого контейнера. Таким образом, страницы размещаются карусельным методом.

В том случае, когда табличное пространство содержит контейнеры разных размеров, карусельный метод неприменим, так как он никак не использует дополнительное пространство в больших контейнерах. В этом случае

применяется карта табличного пространства – она указывает, каким образом экстенты должны быть расположены в табличном пространстве, для того чтобы были доступны все экстенты физических контейнеров.

Примечание: В следующих примерах размер контейнеров указан без учета размера тега контейнера. Рассматриваемые контейнеры очень невелики, они приведены только в качестве примера. Такие размеры контейнеров не рекомендуется устанавливать в реальной рабочей среде. В примерах рассматривается табличное пространство с контейнерами различных размеров, однако в рабочей среде рекомендуется использовать контейнеры одного размера.

Пример 1:

Табличное пространство включает 3 контейнера, каждый из которых содержит по 80 доступных страниц. Размер экстента табличного пространства равен 20. Следовательно, каждый контейнер будет содержать по 4 экстента (80 / 20), а всего будет 12 экстентов. Эти экстенты будут расположены на диске так, как показано на рис. 36.

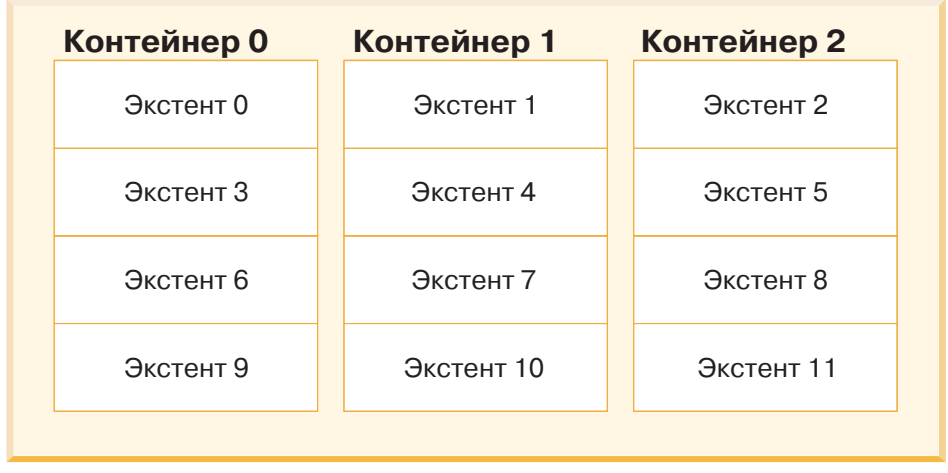

Рисунок 36. Табличное пространство с тремя контейнерами и 12 экстентами

Для того чтобы просмотреть карту табличного пространства, получите снимок табличного пространства с помощью монитора снимков. В примере 1, где рассматривается три контейнера одинакового размера, карта табличного пространства выглядит следующим образом:

```
Номер      Набор   Смещ.   Макс.      Макс.  Нач.    Конечн. Смещ.  Контейнеры
диапазона блоков  блока   экстент    стр.   блок     блок
   [0]       [0]       0      11      239       0         3    0  3 (0, 1, 2)
```

Диапазон - это фрагмент карты, в котором все блоки чередования, расположенные непрерывно, содержат одинаковый набор контейнеров. В примере 1 все блоки чередования (с 0 по 3) содержат одинаковый набор из трех контейнеров (0, 1 и 2), поэтому все они образуют один диапазон.

Карта табличного пространства содержит столбцы Номер диапазона, Набор блоков, Смещение блока, Максимальный номер экстента в диапазоне, Максимальный номер страницы в диапазоне, Начальный блок чередования, Конечный блок чередования, Смещение диапазона и Список контейнеров. Описание этих столбцов приведено в примере 2.

Это табличное пространство можно представить в виде схемы, показанной на рис. 37. Каждая вертикальная линия представляет контейнер, а каждая горизонтальная линия называется *блоком чередования*. Каждый номер ячейки соответствует экстенту.

Рисунок 37. Табличное пространство с тремя контейнерами и 12 экстентами; выделены блоки чередования

Пример 2:

В табличном пространстве расположено два контейнера: первый содержит 100 страниц, а второй - 50 страниц. Размер экстента равен 25. Это означает, что первый контейнер содержит четыре экстента, а второй контейнер - два экстента. Схема табличного пространства показана на рис. 38.

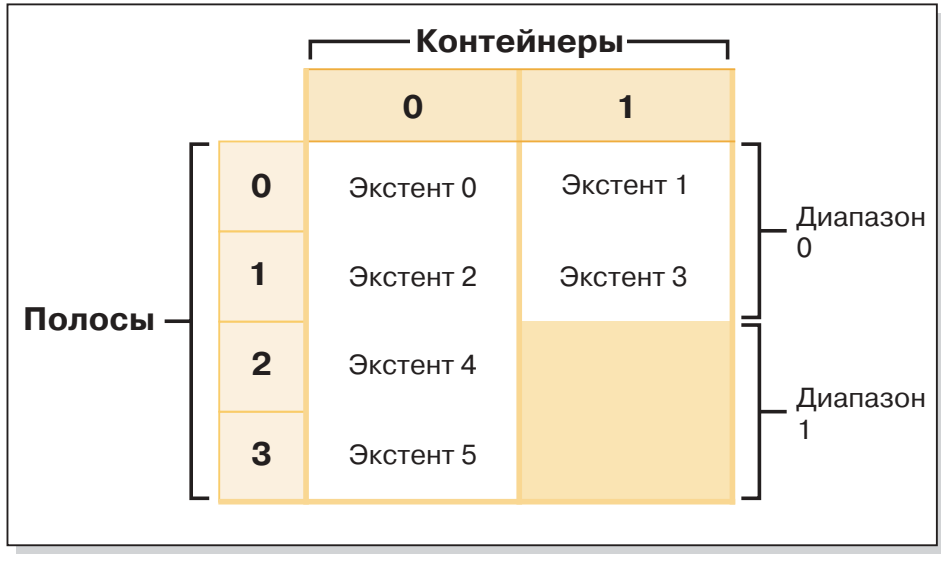

Рисунок 38. Табличное пространство с двумя контейнерами; диапазоны выделены

Блоки чередования 0 и 1 содержат оба контейнера (0 и 1), а блоки чередования 2 и 3 - только первый контейнер (0). Указанные наборы блоков образуют два диапазона. Карта табличного пространства, показанная на снимке, будет выглядеть следующим образом:

```
Номер      Набор   Смещ.   Макс.      Макс.  Нач.     Конечн. Смещ.  Контейнеры
диапазона блоков  блока   экстент    стр.   блок      блок
   [0]       [0]       0       3       99        0         1    0   2 (0, 1)
   [1]       [0]       0       5      149        2         3    0   1 (0)
```

Первый диапазон содержит четыре экстента, поэтому максимальный номер экстента в диапазоне (Макс. экстент) равен 3. Каждый экстент содержит по 25 страниц, поэтому первый диапазон состоит из 100 страниц. Поскольку нумерация страниц начинается с 0, максимальный номер страницы в диапазоне (Макс. стр.) равен 99. Первый блок чередования (Нач. блок) в этом диапазоне - это блок 0, а последний блок чередования (Конечн. блок) - это блок 1. Диапазон содержит два контейнера с номерами 0 и 1. Смещение блока - это первый блок чередования в наборе блоков. В данном случае смещение равно нулю, поскольку есть только один набор блоков чередования. Смещение диапазона (Смещ.) - это

значение, применяемое при перераспределении данных в табличном пространстве. (Перераспределение выполняется при расширении или уменьшении табличного пространства.) До тех пор пока перераспределение не выполнено, это значение всегда равно 0.

Второй диапазон содержит два экстента. Поскольку максимальный номер экстента в предыдущем диапазоне равен 3, максимальный номер экстента в данном диапазоне равен 5. Второй диапазон содержит 50 страниц (2 экстента * 25 страниц). Поскольку максимальный номер страницы в предыдущем диапазоне равен 99, максимальный номер страницы в этом диапазоне равен 149. Этот диапазон начинается с блока чередования 2 и заканчивается блоком чередования 3.

Понятия, связанные с данным:

- “Пространство, управляемое базой данных” на стр. 125
- “Snapshot monitor” в книге *System Monitor Guide and Reference*
- “Добавление контейнеров в табличное пространство DMS и их расширение” на стр. 131
- “Отбрасывание и уменьшение контейнеров в табличном пространстве DMS” на стр. 142

Ссылки, связанные с данной темой:

- “GET SNAPSHOT Command” в книге *Command Reference*

Добавление контейнеров в табличное пространство DMS и их расширение

Вместе с табличным пространством создается его карта, причем все контейнеры выравниваются таким образом, чтобы они начинались в блоке чередования 0. Это означает, что данные будут равномерно распределяться по всем контейнерам табличного пространства до тех пор, пока некоторые из них не заполнятся. (Смотрите раздел “Пример 1” на стр. 133.)

С помощью оператора ALTER TABLESPACE можно добавить новый контейнер или расширить существующий контейнер, чтобы увеличить табличное пространство.

Добавление контейнера, размер которого меньше существующих, приводит к неравномерному распределению данных. Из-за этого операции параллельного ввода/вывода, как, например, предварительная выборка данных, могут выполняться менее эффективно, чем для контейнеров одного размера.

При добавлении контейнеров в табличное пространство или расширении существующих контейнеров *может быть* выполнено перераспределение данных.

Перераспределение данных

Перераспределение данных при добавлении или расширении контейнеров заключается в перемещении экстентов из одного расположения в другое. Целью этого перемещения является равномерное распределение данных в табличном пространстве.

В процессе перераспределения данных доступ к табличному пространству не ограничивается, то есть объекты можно отбрасывать, создавать, заполнять и запрашивать обычным образом. Однако перераспределение данных может оказать заметное влияние на производительность. Если требуется добавить несколько контейнеров, и вы планируете перераспределить данные в контейнерах, добавьте эти контейнеры одновременно с помощью одного оператора ALTER TABLESPACE, для того чтобы менеджеру баз данных не пришлось перераспределять данные несколько раз.

Ключевую роль в процессе перераспределения играет верхняя граница табличного пространства. Верхняя граница задает максимальный номер выделенной страницы в табличном пространстве. Например, если табличное пространство содержит 1000 страниц, и размер экстента равен 10, то всего будет 100 экстентов. Если максимальный номер выделенного экстента в табличном пространстве равен 42, то верхняя граница равна 42 * 10 = 420. Это значение не всегда совпадает с числом занятых страниц, так как некоторые экстенты ниже верхней границы могли быть освобождены для повторного использования.

Перед перераспределением данных создается новая карта табличного пространства, учитывающая те изменения, которые были внесены в контейнеры. В процессе перераспределения экстенты размещаются так, как указано в новой карте табличного пространства. Перераспределение начинается с экстента 0. Затем последовательно перемещаются все остальные экстенты, вплоть до верхней границы. После перемещения каждого экстента в текущую карту вносится соответствующая поправка. После завершения перераспределения текущая карта будет совпадать с новой картой вплоть до блока чередования, содержащего верхнюю границу. После этого в текущую карту вносятся остальные изменения так, чтобы она полностью совпадала с новой картой. На этом процесс перераспределения считается завершенным. Если расположение экстента в текущей карте совпадает с расположением экстента в новой карте, то экстент не перемещается, и операции ввода/вывода не выполняются.

При добавлении контейнера его расположение в карте выбирается в зависимости от его размера и размера остальных контейнеров в том же наборе блоков чередования. Если контейнер достаточно велик для того, чтобы он начинался в первом блоке чередования из набора и заканчивался в последнем блоке чередования (или выходил за его границы), то он размещается именно таким образом (см. “Пример 2” на стр. 134). Если контейнер недостаточно велик, он размещается в карте таким образом, чтобы он заканчивался в

последнем блоке чередования из набора блоков (см. “Пример 4” на стр. 137.) Такое размещение позволяет минимизировать объем перераспределяемых данных.

Примечание: В следующих примерах размер контейнеров указан без учета размера тега контейнера. Рассматриваемые контейнеры очень невелики, они приведены только в качестве примера. Такие размеры контейнеров не рекомендуется устанавливать в реальной рабочей среде. В примерах рассматривается табличное пространство с контейнерами различных размеров, однако в рабочей среде рекомендуется использовать контейнеры одного размера.

Пример 1:

Если в табличном пространстве размещено три контейнера, размер экстента равен 10, и контейнеры содержат 60, 40 и 80 страниц (6, 4 и 8 экстентов), то карта табличного пространства будет выглядеть так, как показано на рис. 39 на стр. 134.

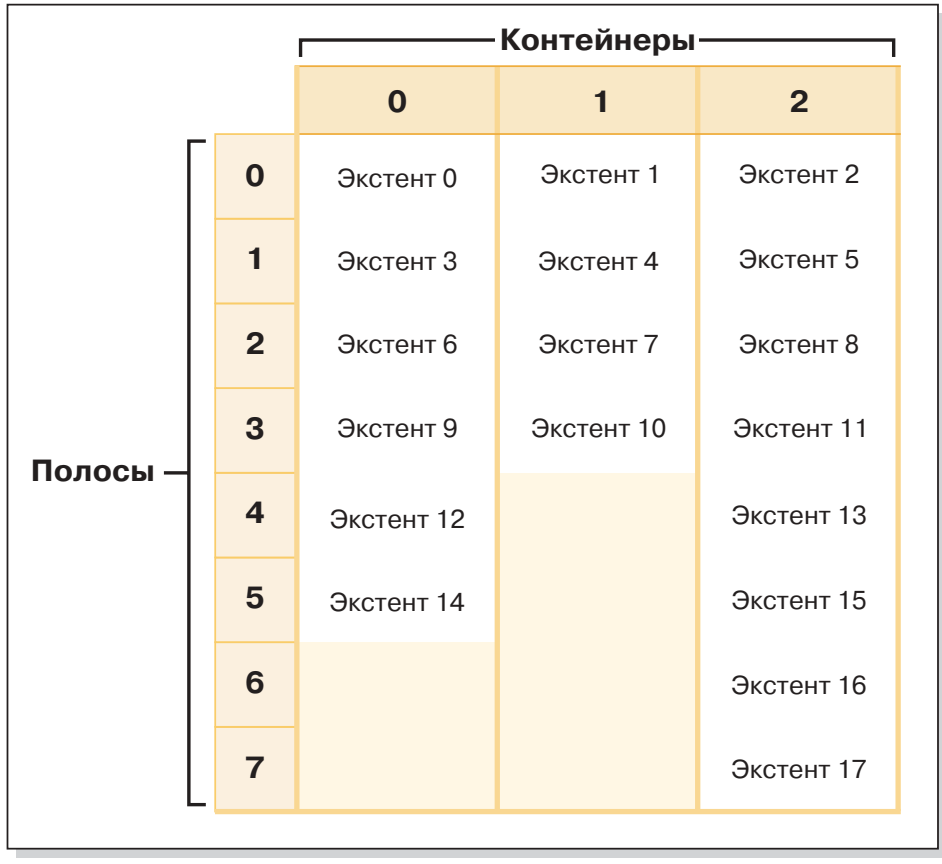

Рисунок 39.

Соответствующая карта табличного пространства, показанная на снимке, будет выглядеть следующим образом:

```
Номер      Набор   Смещ.   Макс.      Макс.  Нач.     Конечн. Смещ.  Контейнеры
диапазона  блоков  блока   экстент    стр.   блок      блок
   [0]       [0]       0       11      119         0         3    0   3 (0, 1, 2)
   [1]       [0]       0       15      159         4         5    0   2 (0, 2)
   [2]       [0]       0       17      179         6         7    0   1 (2)
```

Карта табличного пространства содержит столбцы Номер диапазона, Набор блоков, Смещение блока, Максимальный номер экстента в диапазоне, Максимальный номер страницы в диапазоне, Начальный блок чередования, Конечный блок чередования, Смещение диапазона и Список контейнеров.

Пример 2:

Если в табличное пространство, описанное в примере 1, будет добавлен контейнер, содержащий 80 страниц, то он будет начинаться в первом блоке чередования (блоке 0) и заканчиваться в последнем блоке чередования (блоке 7). Размер контейнера достаточно велик, чтобы его можно было разместить, начиная с первого блока чередования. Схема итогового табличного пространства показана на рис. 40.

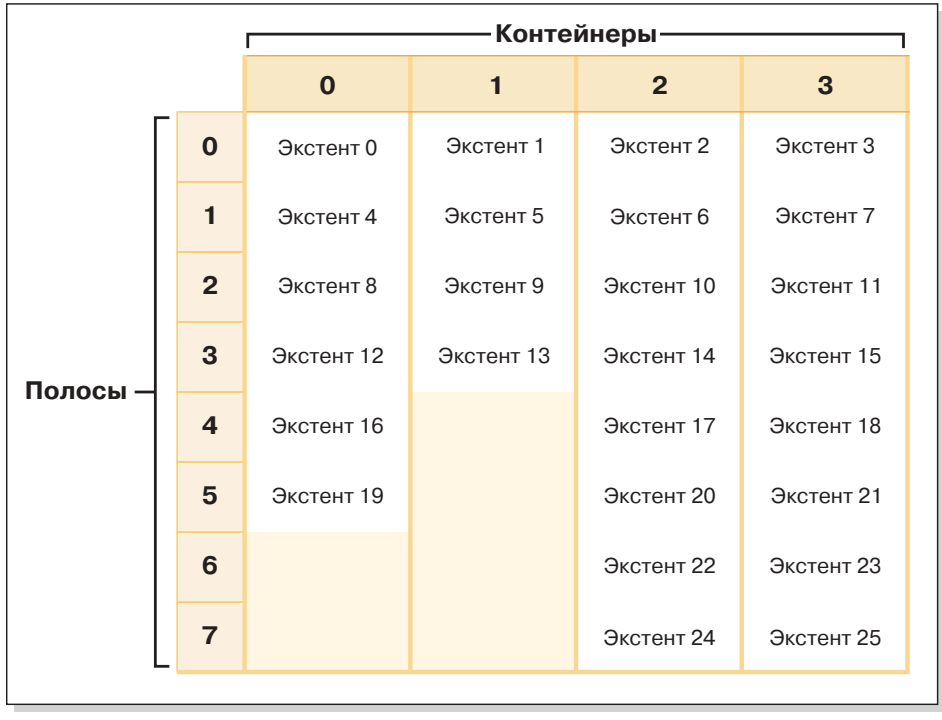

Рисунок 40.

Соответствующая карта табличного пространства, показанная на снимке, будет выглядеть следующим образом:

```
Номер     Набор   Смещ.   Макс.      Макс.  Нач.     Конечн. Смещ.  Контейнеры
диапазона блоков  блока   экстент    стр.   блок      блок
   [0]      [0]       0      15       159       0         3    0   4 (0, 1, 2, 3)
   [1]      [0]       0      21       219       4         5    0   3 (0, 2, 3)
   [2]      [0]       0      25       259       6         7    0   2 (2, 3)
```

Если верхняя граница проходит по экстенту 14, то во время перераспределения данных будут перемещены все экстенты, начиная с 0 и заканчивая 14. Расположение экстента 0 в обеих картах одинаковое, поэтому его перемещать не

нужно. То же самое относится к экстентам 1 и 2. Экстент 3 необходимо переместить, поэтому он считывается из старого расположения (второй экстент в контейнере 0) и записывается в новое расположение (первый экстент в контейнере 3). Все остальные экстенты, вплоть до 14, будут перемещены. После перемещения экстента 14 текущая карта будет полностью совпадать с новой картой, поэтому на этом процедура перераспределения будет завершена.

Если карта изменяется так, что все новое пространство размещается после верхней границы, то перераспределение данных выполняться не будет, и все пространство станет немедленно доступно для использования. Если карта изменяется так, что часть пространства размещается выше верхней границы, а часть - ниже верхней границы, то то пространство, которое находится в блоках чередования выше верхней границы, станет сразу доступно для использования. Остальное пространство станет доступно только после завершения перераспределения данных.

При расширении контейнера перераспределение данных выполняется аналогичным образом. Если после расширения контейнер будет выходить за рамки последнего блока чередования из набора блоков, то набор блоков чередования будет расширен на недостающую величину, а следующие наборы блоков чередования будут сдвинуты на ту же величину. В результате контейнер не будет размещаться в следующих наборах блоках расширения.

Пример 3:

Рассмотрим табличное пространство, описанное в примере 1. Если контейнер 1 будет расширен с 40 страниц до 80 страниц, то новое табличное пространство будет выглядеть так, как показано на рис. 41 на стр. 137.

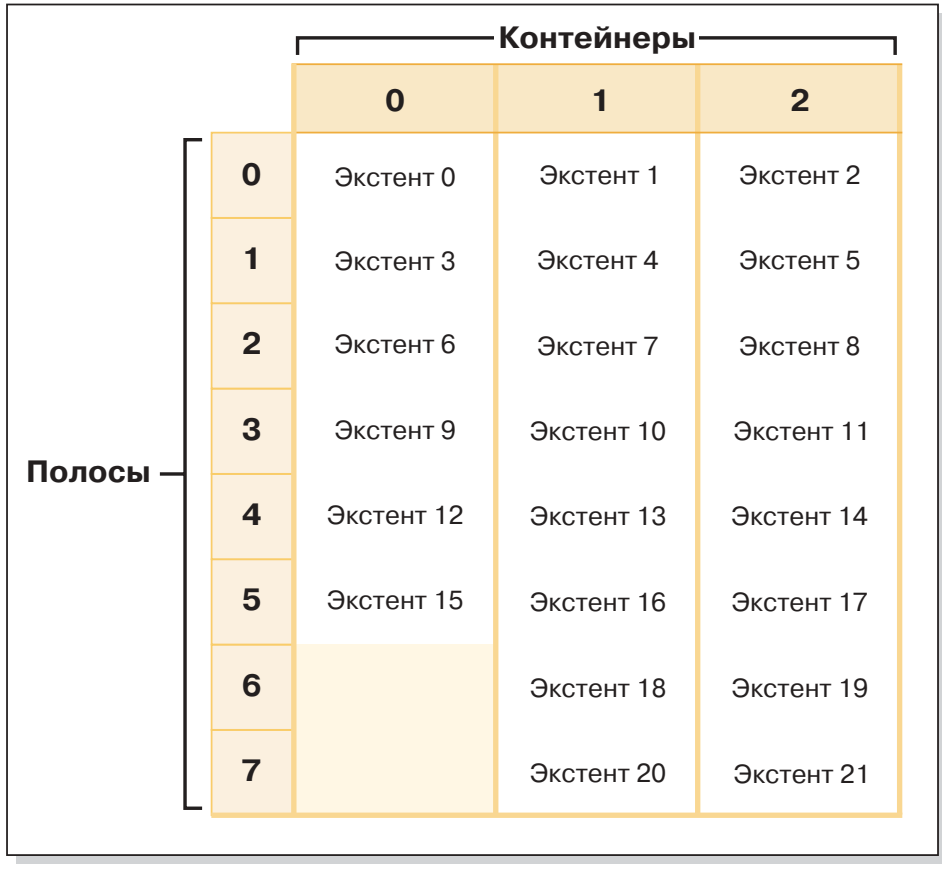

Рисунок 41.

Соответствующая карта табличного пространства, показанная на снимке, будет выглядеть следующим образом:

```
Номер      Набор   Смещ.   Макс.      Макс.  Нач.     Конечн. Смещ.  Контейнеры
диапазона  блоков  блока   экстент    стр.   блок      блок
   [0]       [0]       0       17      179         0        5    0   3 (0, 1, 2)
   [1]       [0]       0       21      219         6        7    0   2 (1, 2)
```

Пример 4:

Рассмотрим табличное пространство, показанное на “Пример 1” на стр. 133. Если в него будет добавлен контейнер размером 50 страниц (5 экстентов), то этот контейнер будет размещен в новой карте следующим образом. Контейнер недостаточно велик, чтобы его можно было разместить, начиная с первого

блока чередования (блока 0) и заканчивая последним блоком чередования (блоком 7), поэтому контейнер размещается таким образом, чтобы он заканчивался в последнем блоке чередования. (Смотрите раздел рис. 42.)

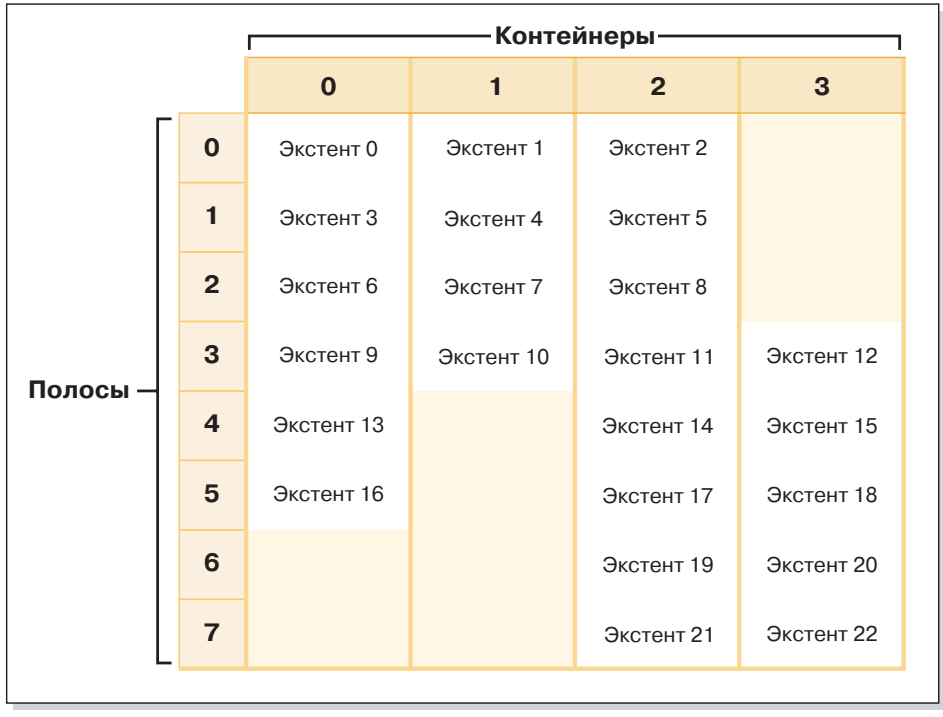

Рисунок 42.

Соответствующая карта табличного пространства, показанная на снимке, будет выглядеть следующим образом:

```
Номер     Набор   Смещ.   Макс.      Макс.  Нач.    Конечн. Смещ.  Контейнеры
диапазона блоков  блока   экстент    стр.   блок     блок
   [0]      [0]      0        8       89        0        2    0   3 (0, 1, 2)
   [1]      [0]      0       12      129        3        3    0   4 (0, 1, 2, 3)
   [2]      [0]      0       18      189        4        5    0   3 (0, 2, 3)
   [3]      [0]      0       22      229        6        7    0   2 (2, 3)
```

Для расширения контейнера укажите опцию EXTEND или RESIZE в операторе ALTER TABLESPACE. Для добавления контейнера и перераспределения данных укажите опцию ADD в операторе ALTER TABLESPACE. Если контейнер добавляется в табличное пространство, содержащее несколько наборов блоков чередования, вы можете явно указать, в каком наборе нужно разместить

контейнер. Для этого применяется опция ADD TO STRIPE SET оператора ALTER TABLESPACE. Если набор блоков чередования не задан, то по умолчанию контейнер добавляется в текущий набор блоков. Текущим считается тот набор блоков чередования, который был создан последним, а не тот, который был расширен последним.

Изменение набора блоков чередования может повлечь за собой перераспределение данных в этом наборе и последующих наборах.

Вы можете контролировать ход перераспределения данных с помощью снимков табличного пространства. Снимок табличного пространства содержит такую информацию о перераспределении, как время начала операции, число перемещенных экстентов и общее число экстентов, которые требуется переместить.

Без перераспределения (с помощью наборов блоков чередования)

Если при добавлении или расширении контейнера новое пространство будет размещено выше верхней границы табличного пространства, то перераспределение данных выполняться не будет.

При добавлении контейнера почти всегда часть пространства размещается ниже верхней границы. Это означает, что обычно при добавлении контейнера выполняется перераспределение данных. Существует возможность принудительно добавить контейнер выше верхней границы и не выполнять перераспределение данных табличного пространства. Преимущество этого способа заключается в том, что новый контейнер сразу станет доступен для использования. Отменить перераспределение данных можно только при добавлении контейнеров, но не при расширении существующих контейнеров. Избежать перераспределения данных при расширении контейнеров можно лишь в том случае, если добавляемое пространство будет размещено выше верхней границы. Например, если есть несколько контейнеров одинакового размера, и все они увеличиваются на одну и ту же величину, то относительное расположение экстентов не изменится, поэтому данные перераспределяться не будут.

Для добавления контейнеров без перераспределения данных применяется новый *набор блоков чередования*. Набор блоков чередования - это набор контейнеров табличного пространства, в которых данные распределены независимо от остальных контейнеров табличного пространства. Новый набор блоков чередования создается в том случае, если необходимо добавить контейнеры, не перераспределяя данные. Контейнеры в существующих наборах блоков чередования не будут затронуты этой операций, и добавляемые контейнеры станут частью нового набора блоков чередования.

Для добавления контейнеров без перераспределения данных укажите опцию BEGIN NEW STRIPE SET в операторе ALTER TABLESPACE.

Пример 5:

Если в табличном пространстве размещено три контейнера, размер экстента равен 10, и контейнеры содержат 30, 40 и 40 страниц (3, 4 и 4 экстента, соответственно), то схема табличного пространства будет выглядеть так, как показано на рис. 43.

Рисунок 43.

Соответствующая карта табличного пространства, показанная на снимке, будет выглядеть следующим образом:

```
Номер     Набор   Смещ.   Макс.      Макс.  Нач.     Конечн. Смещ.  Контейнеры
диапазона блоков  блока   экстент    стр.   блок      блок
   [0]      [0]       0       8       89        0        2    0   3 (0, 1, 2)
   [1]      [0]       0      10      109        3        3    0   2 (1, 2)
```

Пример 6:

При добавлении двух контейнеров размером 30 и 40 страниц (3 и 4 экстента, соответственно) с помощью опции BEGIN NEW STRIPE SET существующие диапазоны не изменяются, а вместо этого создается новый набор диапазонов. Этот набор представляет собой набор блоков чередования. Последний созданный набор называется текущим. После добавления двух контейнеров табличное пространство будет выглядеть так, как показано на рис. 44 на стр. 141.

Рисунок 44.

Соответствующая карта табличного пространства, показанная на снимке, будет выглядеть следующим образом:

```
Номер     Набор   Смещ.   Макс.      Макс.  Нач.     Конечн. Смещ.  Контейнеры
диапазона блоков  блока   экстент    стр.   блок      блок
   [0]       [0]       0       8       89        0        2    0   3 (0, 1, 2)
   [1]       [0]       0      10      109        3        3    0   2 (1, 2)
   [2]       [1]       4      16      169        4        6    0   2 (3, 4)
   [3]       [1]       4      17      179        7        7    0   1 (4)
```

Если при добавлении контейнеров в табличное пространства в условии ADD *не* будет задана опция TO STRIPE SET, контейнеры будут добавлены в текущий набор блоков чередования (самый верхний набор). Вы можете добавить контейнеры в любой другой набор, указав опцию ADD TO STRIPE SET. В этой опции можно задать любой допустимый набор блоков чередования.

DB2® хранит информацию о наборах блоков чередования в карте табличного пространства. При добавлении контейнеров без перераспределения данных размер карты увеличивается быстрее, чем в случае перераспределения данных в контейнерах. Когда размер карты табличного пространства становится слишком большим, при очередной попытке добавить контейнеры отправляется сообщение об ошибке SQL0259N.

Понятия, связанные с данным:

- “Карты табличных пространств” на стр. 127

Задачи, связанные с данной темой:

- “Добавление контейнера в табличное пространство DMS” в книге *Руководство администратора: Реализация*
- “Изменение контейнеров в табличном пространстве DMS” в книге *Руководство администратора: Реализация*

Ссылки, связанные с данной темой:

- “ALTER TABLESPACE statement” в книге *SQL Reference, Том 2*
- “GET SNAPSHOT Command” в книге *Command Reference*
- “Элементы данных активности табличных пространств” в книге *System Monitor Guide and Reference*

Отбрасывание и уменьшение контейнеров в табличном пространстве DMS

Вы можете отбросить контейнер, содержащийся в табличном пространстве DMS, или уменьшить его размер. Для этого служит оператор ALTER TABLESPACE.

Отбрасывание и уменьшение контейнеров допустимо лишь в том случае, если число экстентов, отбрасываемых в ходе операции, не больше числа свободных экстентов, расположенных в табличном пространстве выше верхней границы. Это ограничение связано с тем, что при выполнении операции нельзя изменить номера страниц, поэтому логическая позиция всех экстентов табличного пространства вплоть до верхней границы должна совпадать. В итоговом табличном пространстве ниже верхней границы должно быть достаточно места для хранения всех данных. Если свободного места недостаточно, сразу после выполнения оператора будет выдано сообщение об ошибке.

Верхняя граница задает максимальный номер выделенной страницы в табличном пространстве. Например, если табличное пространство содержит 1000 страниц, и размер экстента равен 10, то всего будет 100 экстентов. Если максимальный номер выделенного экстента в табличном пространстве равен 42, то верхняя граница равна 42 * 10 = 420. Это значение не всегда совпадает с числом занятых страниц, так как некоторые экстенты ниже верхней границы могли быть освобождены для повторного использования.

При отбрасывании или уменьшении контейнеров выполняется перераспределение данных, если в отбрасываемом пространстве содержатся данные. Перед перераспределением данных создается новая карта табличного пространства, учитывающая те изменения, которые были внесены в контейнеры. В процессе перераспределения экстенты размещаются так, как указано в новой карте табличного пространства. Перераспределение начинается с экстента,

содержащего верхнюю границу. Затем последовательно перемещаются остальные экстенты вплоть до экстента 0. После перемещения каждого экстента в текущую карту вносится соответствующая поправка. Если расположение экстента в текущей карте совпадает с расположением экстента в новой карте, то экстент не перемещается, и операции ввода/вывода не выполняются. Поскольку перераспределение выполняется начиная с максимального номера выделенного экстента и заканчивая первым экстентом табличного пространства, эта процедура называется *обратным перераспределением* (в отличие от *прямого перераспределения*, которое выполняется при добавлении и расширении контейнеров табличного пространства).

После отбрасывания некоторых контейнеров всем остальным контейнерам заново присваиваются номера, начиная с 0. Если были отброшены все контейнеры из набора блоков чередования, то этот набор удаляется из карты, а все последующие наборы блоков чередования смещаются вниз и заново нумеруются таким образом, чтобы в номерах наборов не было пропусков.

Примечание: В следующих примерах размер контейнеров указан без учета размера тега контейнера. Рассматриваемые контейнеры очень невелики, они приведены только в качестве примера. Такие размеры контейнеров не рекомендуется устанавливать в реальной рабочей среде. В примерах рассматривается табличное пространство с контейнерами различных размеров, однако в рабочей среде рекомендуется использовать контейнеры одного размера.

Например, рассмотрим табличное пространство с тремя контейнерами, в котором размер экстента равен 10. Контейнеры содержат 20, 50 и 50 страниц, соответственно (2, 5 и 5 экстентов). Схема табличного пространства показана на рис. 45 на стр. 144.

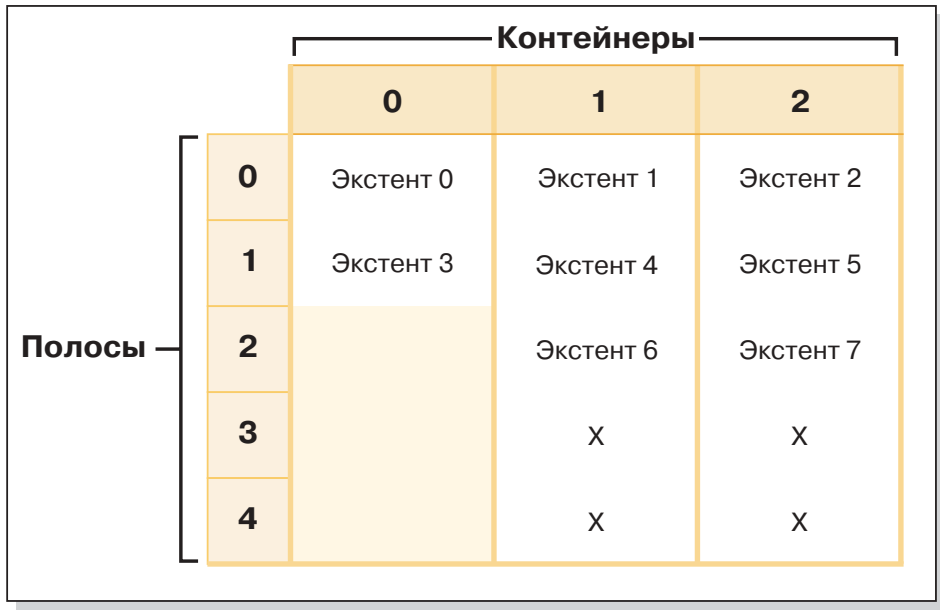

Рисунок 45.

Символом X обозначены экстенты, не содержащие данных.

Для того чтобы отбросить контейнер 0, содержащий два экстента, ниже верхней границы должно быть по крайней мере два свободных экстента. Верхняя граница расположена в экстенте 7. Ниже нее расположено четыре свободных экстента, поэтому можно отбросить контейнер 0.

Соответствующая карта табличного пространства, показанная на снимке, будет выглядеть следующим образом:

```
Номер      Набор   Смещ.   Макс.      Макс.  Нач.    Конечн. Смещ.  Контейнеры
диапазона блоков  блока   экстент    стр.   блок     блок
   [0]       [0]       0       5       59        0        1    0   3 (0, 1, 2)
   [1]       [0]       0      11      119        2        4    0   2 (1, 2)
```

После отбрасывания контейнера в табличном пространстве останутся контейнеры 0 и 1. Новая схема табличного пространства показана на рис. 46 на стр. 145.

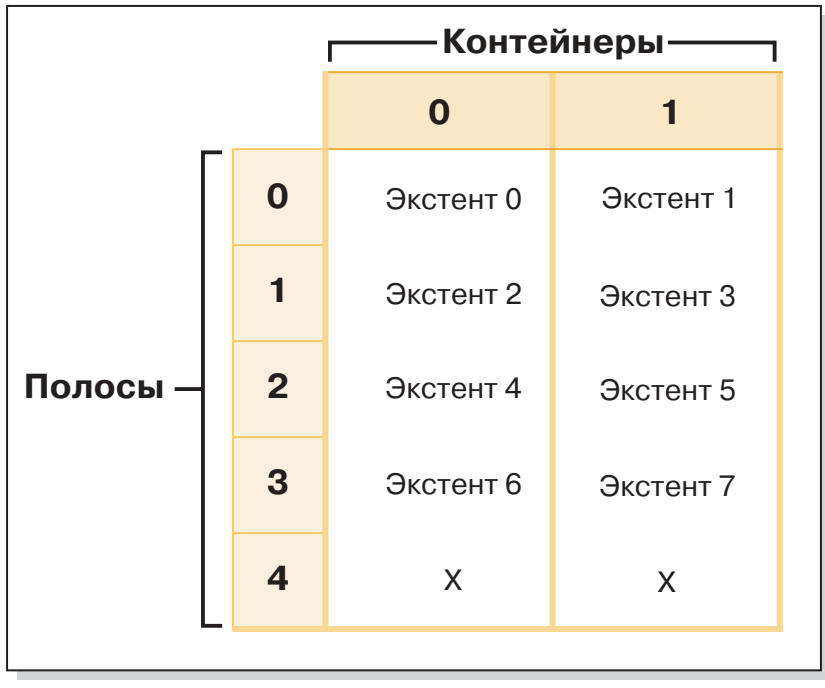

Рисунок 46.

Соответствующая карта табличного пространства, показанная на снимке, будет выглядеть следующим образом:

```
Номер      Набор   Смещ.   Макс.      Макс.  Нач.     Конечн. Смещ.  Контейнеры
диапазона  блоков  блока   экстент    стр.   блок      блок
   [0]       [0]       0       9       99        0         4    0   2 (0, 1)
```

При уменьшении контейнера процедура перераспределения данных выполняется аналогичным образом.

Для уменьшения контейнера в операторе ALTER TABLESPACE нужно указать опцию REDUCE или RESIZE. Для отбрасывания контейнера в операторе ALTER TABLESPACE нужно указать опцию DROP.

Понятия, связанные с данным:

- “Карты табличных пространств” на стр. 127

Задачи, связанные с данной темой:

- “Изменение контейнеров в табличном пространстве DMS” в книге *Руководство администратора: Реализация*

Ссылки, связанные с данной темой:

- “ALTER TABLESPACE statement” в книге *SQL Reference, Том 2*
- “GET SNAPSHOT Command” в книге *Command Reference*
- “Элементы данных активности табличных пространств” в книге *System Monitor Guide and Reference*

Сравнение табличных пространств SMS и DMS

Выбирая, в табличное пространство какого типа поместить данные, надо принять во внимание следующее:

Достоинства табличного пространства SMS:

- Ресурсы выделяются системой только при необходимости.
- Для создания табличного пространства требуется меньше подготовительных усилий, так как вам не требуется определять контейнеры заранее.

Достоинства табличного пространства DMS:

- Табличное пространство можно расширять, добавляя или расширяя контейнеры с помощью оператора ALTER TABLESPACE. Существующие данные могу автоматически перебалансироваться по новому набору контейнеров, что сохраняет максимальную эффективность ввода/вывода.
- Таблицу можно разбить на несколько табличных пространств в зависимости от типа хранимых данных:
  - Длинные поля и типы данных больших объектов
  - Индексы
  - Обычные табличные данные

  Табличные данные можно разделить для повышения производительности или для увеличения объема данных, хранящихся в таблице. Например, в таблице можно хранить 64 Гбайта обычных данных, 64 Гбайта индексов и 2 Тбайта длинных данных. Если вы используете страницы по 8 Кбайт, табличные данные и данные индексов могут занять до 128 Гбайт. Если вы используете страницы по 16 Кбайт, они могут достигать 256 Гбайт. Если вы используете страницы по 32 Кбайта, табличные данные и данные индексов могут достигать 512 Гбайт.
- Если операционная система позволяет, можно управлять размещением данных на диске.
- Если все табличные данные находятся в одном табличном пространстве, его можно отбросить и переопределить с меньшими затратами, чем потребовалось бы для отбрасывания и переопределения таблицы.
- У хорошо настроенного набора табличных пространств DMS производительность в целом выше, чем у табличных пространств SMS.

Примечание: В Solaris для нагрузок, где важна производительность, настоятельно рекомендуются табличные пространства DMS с непосредственными устройствами.

Небольшие персональные базы данных обычно проще вести, используя табличные пространства SMS. С другой стороны, в больших, растущих базах данных табличные пространства SMS скорее всего будут использованы лишь как временные пространства и пространства каталогов, а для всех прочих таблиц будут выделены отдельные табличные пространства DMS, включающие по нескольку контейнеров. Кроме того, вероятно, вы захотите хранить данные длинных полей и индексы в отдельных табличных пространствах.

Если вы выбрали табличные пространства DMS с контейнерами-устройствами, вам стоит настроить вашу среду и управлять ей.

Понятия, связанные с данным:

- “Проектирование табличного пространства” на стр. 118
- “Пространство, управляемое системой” на стр. 122
- “Пространство, управляемое базой данных” на стр. 125

Ввод-вывод на диск для табличного пространства

Эффективность работы ввода/вывода для табличного пространства определяется типом и структурой этого табличного пространства. Ниже изложены основные понятия, которые необходимо знать до того, как продолжить изучение проектирования структуры табличного пространства и его использования:

Чтение больших блоков
Операция чтения, при которой по одному требованию получается несколько страниц (обычно - экстент). Чтение сразу нескольких страниц более эффективно, чем чтение отдельных страниц.

Предварительная выборка
Чтение страниц до того, как их потребует запрос. Основная цель - уменьшить время ответа. Этого можно достичь, если предварительная выборка страниц может выполняться асинхронно с выполнением запроса. Наилучшее время ответа достигается, если либо процессор, либо подсистема ввода/вывода работают на максимальной мощности.

Очистка страниц
По мере чтения и изменения страниц они накапливаются в пуле буферов базы данных. При считывании страницы она считывается в страницу пула буферов. Если пул буферов

заполнен измененными страницами, до того, как можно будет считать новую страницу, надо записать на диск одну из измененных страниц. Чтобы пул буферов не заполнялся, средства очистки страниц записывают на диск измененные страницы, чтобы гарантировать доступность страниц пула буферов для дальнейших требований чтения.

Во всех случаях, когда это дает преимущество, DB2® выполняет чтение больших блоков. Обычно это происходит при получении данных, которые по своей природе последовательны или частично последовательны. Объем данных, считываемых за одну операцию чтения, зависит от размера экстента – чем больше размер экстента, тем больше страниц можно считать за раз.

Производительность последовательной предварительной выборки можно повысить, если читать страницы с диска в идущие подряд страницы в пуле буферов. Так как, по умолчанию, все пулы буферов создаются на основе страниц, то смежные страницы не всегда расположены на диске подряд. Для этого могут применяться блочные пулы буферов, содержащие не только область страниц, но и область блоков для наборов смежных страниц. Каждый набор смежных страниц называется блоком, а каждый блок содержит определенное число страниц, называемое размером блока. Размеры области страниц и области блоков, а также число страниц в каждом блоке можно настроить.

Способ хранения экстента на диске влияет на эффективность ввода/вывода. В табличном пространстве DMS, использующем контейнеры-устройства, данные имеют тенденцию к последовательному расположению на диске, и их можно прочесть с минимальным временем доступа и работы диска. Но при использовании файлов файловая система может разбить данные и хранить их в нескольких местах диска. Наиболее часто это происходит при использовании табличных пространств SMS, в которых файлы увеличиваются по одной странице за раз, что увеличивает вероятность фрагментации. Большой файл, выделенный под табличное пространство DMS, обычно хранится на диске непрерывно и последовательно, в особенности, если он был выделен в чистой файловой системе.

Степенью предварительной выборки можно управлять при помощи параметра PREFETCHSIZE операторов CREATE TABLESPACE и ALTER TABLESPACE. (Значение по умолчанию для всех табличных пространств в базе данных задается параметром конфигурации баз данных *dft_prefetch_sz*.) Параметр PREFETCHSIZE задает число страниц, которое DB2 считывает при каждой предварительной выборке. Если задать значение PREFETCHSIZE, кратное значению параметра EXTENTSIZE оператора CREATE TABLESPACE, можно параллельно считывать несколько экстентов. (Значение по умолчанию для всех табличных пространств в базе данных задается параметром конфигурации баз

данных *dft_extent_sz*.) Параметр EXTENTSIZE задает число страниц по 4 Кбайта, которое записывается в контейнер перед тем, как перейти к следующему контейнеру.

Допустим, что у вас было табличное пространство, использующее три устройства. Если задать PREFETCHSIZE, в три раза большее, чем EXTENTSIZE, DB2 может параллельно считать большие блоки со всех устройств, таким образом значительно увеличив пропускную способность ввода/вывода. Здесь предполагается, что все устройства являются отдельными физическими устройствами и что пропускная способность контроллера достаточна для обработки потока от каждого устройства. Учтите, что при работе DB2 значения параметров предварительной выборки могут быть динамически изменены в зависимости от скорости запроса, использования пула буферов и других факторов.

Некоторые файловые системы используют собственный метод предварительной выборки (например, файловая система JFS - Journaled File System - в AIX). В некоторых случаях предварительная выборка файловой системы работает более агрессивно, чем предварительная выборка DB2. Это может вызвать видимость превосходства производительности предварительной выборки для табличных пространств SMS и DMS с контейнерами-файлами над предварительной выборкой для табличных пространств DMS с устройствами. Это заблуждение, так как, скорее всего, это результат еще одного уровня предварительной выборки, происходящей в файловой системе. Табличные пространства DMS должны превосходить по производительности любую такую конфигурацию.

Для эффективной предварительной выборки (и даже чтения) нужно, чтобы в буферном пуле было достаточно чистых страниц. Возьмем, например, требование параллельной предварительной выборки, считывающее из табличного пространства три экстента, и при считывании каждой страницы из пула буферов записывается на диск одна измененная страница. Требование предварительной выборки может замедлиться до точки, когда оно перестает успевать за запросом. Чтобы требование предварительной выборки было удовлетворено, нужно сконфигурировать достаточное число чистильщиков страниц.

Понятия, связанные с данным:

- “Проектирование табличного пространства” на стр. 118
- “Предварительная выборка в пул буферов” в книге *Руководство администратора: Производительность*

Ссылки, связанные с данной темой:

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

Сведения о рабочей нагрузке при разработке табличных пространств

На выбор типа табличных пространств и размера страницы может повлиять преимущественный тип нагрузки DB2® в вашей среде. Нагрузка оперативной обработки транзакций (OLTP) характеризуется транзакциями, которым нужен нерегулярный доступ к данным, которые содержат множество операций вставки и изменения и, обычно, возвращают небольшие наборы данных. Если доступ нерегулярный и запрашивается одна или несколько страниц, предварительная выборка происходит с меньшей вероятностью.

В такой ситуации лучше всего производительность у табличных пространств DMS, использующих контейнеры-устройства. Если максимальная производительность не важна, для нагрузок OLTP можно использовать табличные пространства DMS с контейнерами-файлами или табличные пространства SMS. Если ожидается малый объем последовательного ввода/вывода или же его не ожидается вовсе, значения параметров EXTENTSIZE и PREFETCHSIZE оператора CREATE TABLESPACE не влияют на эффективность ввода/вывода. Однако важно задать в параметре конфигурации *chngpgs_thresh* необходимое число процессов очистки страниц.

Нагрузка запросов характеризуется транзакциями, которым нужен последовательный или частично последовательный доступ к данным и которые обычно возвращают большие наборы данных. Табличное пространство DMS, использующее несколько контейнеров-устройств (где каждый контейнер находится на отдельном диске), дает наибольшие возможности эффективной параллельной предварительной выборки. Значение параметра PREFETCHSIZE оператора CREATE TABLESPACE должно быть равно произведению значения параметра EXTENTSIZE и числа контейнеров-устройств. Это позволяет DB2 параллельно выполнять предварительную выборку изо всех контейнеров. При изменении числа контейнеров или при необходимости сделать предварительную выборку более или менее агрессивной можно изменить значение PREFETCHSIZE с помощью оператора ALTER TABLESPACE.

Если файловая система выполняет собственную предварительную выборку, разумная альтернатива для нагрузки запросов - использовать файлы. Файлы могут содержать как контейнеры типа DMS, так и контейнеры типа SMS. Имейте в виду, что если вы используете SMS, для достижения параллельности ввода/вывода нужно, чтобы контейнеры каталогов отображались на разные физические диски.

Для смешанной нагрузки ваша задача - сделать как можно более эффективными одиночные требования ввода/вывода для нагрузки OLTP и повысить эффективность параллельного ввода/вывода для нагрузки запросов.

При определении размера страницы для рабочего пространства нужно учитывать следующие особенности:

- Для программ OLTP, выполняющих нерегулярные операции чтения и записи строк, обычно предпочтителен меньший размер страницы, чтобы тратить меньше места в пуле буферов на ненужные строки.
- При работе с программами DSS, обращающимися сразу к большому числу последовательных строк, обычно лучше использовать больший размер страницы, так как это уменьшает количество требований ввода/вывода, необходимых для считывания заданного числа строк. Но здесь есть одно исключение. Если ваш размер строки меньше, чем:

  ```
  размер_страницы / 255
  ```

  на всех страницах окажется неиспользуемое место (максимальное число строк на страницу - 255). В этой ситуации, возможно, лучше использовать меньший размер страницы.
- Большие размеры страниц позволяют уменьшить число уровней в индексе.
- Большие страницы поддерживают строки большей длины.
- На страницах по умолчанию размером 4 Кбайта в таблицах может быть максимум 500 столбцов, в то время как большие размеры страниц (8 Кбайт, 16 Кбайт и 32 Кбайта) поддерживают 1012 столбцов.
- Максимальный размер табличного пространства прямо пропорционален размеру страницы в этом табличном пространстве.

Понятия, связанные с данным:

- “Пространство, управляемое системой” на стр. 122
- “Пространство, управляемое базой данных” на стр. 125

Ссылки, связанные с данной темой:

- “Параметр конфигурации Порог числа измененных страниц - chngpgs_thresh” в книге *Руководство администратора: Производительность*
- “ALTER TABLESPACE statement” в книге *SQL Reference, Том 2*
- “CREATE TABLESPACE statement” в книге *SQL Reference, Том 2*
- “SQL limits” в книге *SQL Reference, Том 1*

Размер экстента

Размер экстента для табличного пространства отражает число страниц табличных данных, которое записывается в контейнер перед тем, как начать записывать данные в следующий контейнер. При выборе размера экстента следует учитывать:

- Размер и тип таблиц в табличном пространстве.

  В табличных пространствах DMS место для таблицы выделяется по одному экстенту за раз. Когда таблица и экстент заполняются, создается новый

экстент. Если включено многостраничное размещение файлов, то пространство в табличных пространствах SMS выделяется одному экстенту.

Таблица состоит из следующих отдельных табличных объектов:

– Объект данных. Здесь хранятся обычные данные столбцов.
– Объект индексов. Здесь хранятся все индексы, определенные для этой таблицы.
– Объект длинных полей. Здесь, если в таблице есть столбцы LONG, хранятся данные длинных полей.
– Два объекта LOB. Если в таблице есть столбцы больших объектов, они хранятся в этих двух табличных объектах:
  - Один табличный объект для данных больших объектов
  - Второй табличный объект для метаданных, описывающих эти данные.
– Объект карты блоков для многомерных таблиц.

Все табличные объекты хранятся отдельно, и для каждого объекта по мере надобности выделяются новые экстенты. Для каждого табличного объекта имеется объект метаданных, называющийся *картой экстентов*, который описывает все экстенты в этом табличном пространстве, принадлежащие объекту таблицы. Место для карт экстентов также выделяется по одному экстенту за раз.

Таким образом, изначально для таблицы выделяется два экстента для каждого табличного объекта. Если в табличном пространстве слишком много маленьких таблиц, для хранения относительно небольшого количества данных выделяется относительно большое пространство. В этом случае следует задать маленький размер экстента или использовать табличное пространство SMS, которое выделяет по одной странице за раз.

С другой стороны, если у вас есть очень большая таблица, которая быстро растет, и вы используете табличное пространство DMS с маленьким размером экстента, в связи с частым выделением дополнительных экстентов может образоваться ненужное дополнительное место.

- Тип доступа к таблицам.

  Если при доступе к таблицам выполняется много запросов или транзакций, обрабатывающих большие объемы данных, значительное улучшение производительности достигается предварительной выборкой данных.

- Минимально необходимое число экстентов.

  Если в контейнерах не хватает места для пяти экстентов табличного пространства, табличное пространство не создается.

Понятия, связанные с данным:

- “Проектирование табличного пространства” на стр. 118

Ссылки, связанные с данной темой:

- “CREATE TABLESPACE statement” в книге *SQL Reference, Том 2*
- “db2empfa - Enable Multipage File Allocation Command” в книге *Command Reference*

Отношения между табличными пространствами и пулами буферов

Каждое табличное пространство связано с некоторым пулом буферов. Пул буферов по умолчанию - IBMDEFAULTBP. Если табличное пространство надо связать с другим пулом буферов, этот пул буферов должен существовать (определяется оператором CREATE BUFFERPOOL) и иметь тот же размер страницы; тогда связь определяется при создании табличного пространства (при помощи оператора CREATE TABLESPACE). Связь между табличным пространством и пулом буферов можно изменить при помощи оператора ALTER TABLESPACE.

Наличие нескольких пулов буферов позволяет конфигурировать память, используемую базой данных, для улучшения общей производительности. Например, для табличных пространств с одной или несколькими большими таблицами (размер которых превышает объем доступной памяти), к которым нерегулярно обращаются пользователи, можно ограничить размер пула буферов, так как кэширование страниц данных может быть бесполезным. Табличное пространство для программы сетевых транзакций можно связать с большим пулом буферов, чтобы используемые программой страницы данных дольше хранились в кэше, что ускорит время ответа. При конфигурировании новых пулов буферов следует соблюдать осторожность.

Примечание: Если вы определили, что базе данных требуются страницы размером 8 Кбайт, 16 Кбайт или 32 Кбайта, все табличные пространства с такими размерами страницы должны отображаться в пулы буферов с тем же размером страницы.

При запуске базы данных менеджеру баз данных должна быть доступна память, необходимая для всех пулов буферов. Если DB2® не может получить требуемую память, менеджер баз данных будет запущен с пулами буферов по умолчанию (по одному с размерами страницы 4 Кбайта, 8 Кбайт, 16 Кбайт и 32 Кбайта) и выдаст предупреждение.

В среде многораздельных баз данных можно создать пул буферов одного и того же размера для всех разделов базы данных. Можно также создавать пулы буферов разных размеров на разных разделах.

Понятия, связанные с данным:

- “Table spaces and other storage structures” в книге *SQL Reference, Том 1*

Ссылки, связанные с данной темой:

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

Отношения между табличными пространствами и группами разделов баз данных

В среде многораздельных баз данных каждое табличное пространство связано с некоторой группой разделов базы данных. Это позволяет применить характеристики табличного пространства к каждому разделу в этой группе разделов базы данных. Группа разделов базы данных должна существовать (определяется оператором CREATE DATABASE PARTITION GROUP); тогда связь между табличным пространством и группой разделов базы данных определяется при создании табличного пространства оператором CREATE TABLESPACE.

Связь между табличным пространством и группой разделов базы данных нельзя изменить при помощи оператора ALTER TABLESPACE. Можно только изменять задание табличного пространства для отдельных разделов этой группы разделов базы данных. В среде одноразделных баз данных все табличные пространства связаны с группой разделов базы данных по умолчанию. Группа разделов базы данных по умолчанию при определении табличного пространства называется IBMDEFAULTGROUP, если определяется не системное временное табличное пространство; в противном случае используется IBMTEMPGROUP.

Понятия, связанные с данным:

- “Table spaces and other storage structures” в книге *SQL Reference, Том 1*
- “Группы разделов базы данных” на стр. 108
- “Проектирование табличного пространства” на стр. 118

Ссылки, связанные с данной темой:

- “CREATE DATABASE PARTITION GROUP statement” в книге *SQL Reference, Том 2*
- “CREATE TABLESPACE statement” в книге *SQL Reference, Том 2*

Проектирование временных табличных пространств

Рекомендуется определить одно временное табличное пространство SMS с размером страницы, совпадающим с размером страницы большинства обычных табличных пространств. Это должно подходить для типичных состояний среды и рабочей нагрузки. Но, возможно, экспериментирование с различными конфигурациями и нагрузками временных табличных пространств может дать лучшие результаты. Необходимо учитывать следующие особенности:

- В большинстве случаев доступ к временным таблицам выполняется последовательно в виде пакетов. Это значит, что вставляется пакет строк или производится чтение пакета последовательных строк. Таким образом, обычно увеличение размера страницы улучшает производительность, так как для считывания заданного количества данных нужно меньше логических или физических требований страничного ввода/вывода. Это не обязательно так, если средний размер строки временной таблицы меньше размера страницы, деленного на 255. На любой странице, независимо от размера страницы, может быть до 255 строк. Например, для запроса, требующего временную таблицу со строками по 15 байт, лучше использовать временное табличное пространство с размером страницы 4 Кбайта, поскольку 255 таких строк все помещаются в странице размером 4 Кбайта. Размер страницы 8 Кбайт и более вызовет потерю по меньшей мере 4 Кбайт на каждой странице временной таблицы и не уменьшит достаточного числа требований ввода/вывода.
- Если в базе данных больше пятидесяти процентов обычных табличных пространств используют один и тот же размер страницы, вероятно, стоит задать тот же размер страницы для временных табличных пространств. Это позволяет временному табличному пространству совместно использовать один пул буферов вместе с большинством или всеми обычными табличными пространствами. Это, в свою очередь, упрощает настройку пулов буферов.
- При реорганизации таблицы с использованием временного табличного пространства размеры страниц временного табличного пространства и данной таблицы должны совпадать. Поэтому необходимо, чтобы для всех размеров страниц, используемых существующими таблицами, которые могут быть реорганизованы с использованием временного табличного пространства, были определены временные табличные пространства.

  Другой вариант - реорганизация без временного табличного пространства; таблица реорганизуется прямо в табличном пространстве назначения. Конечно, для этого типа реорганизации необходимо, чтобы в табличном пространстве назначения было достаточно места.
- Обычно, если существуют временные табличные пространства с разным размером страницы, оптимизатор выбирает временное табличное пространство с наибольшим пулом буферов. Часто в таких случаях имеет смысл назначить одному из временных табличных пространств большой пул буферов, а остальным назначить меньший пул буферов. Такое назначение

пулов буферов обеспечивает эффективное использование основной памяти. Например, если табличное пространство каталога использует страницы по 4 Кбайта, а остальные табличные пространства используют страницы по 8 Кбайт, возможно, что лучшая конфигурация временных табличных пространств - одно временное табличное пространство размером 8 Кбайт с большим пулом буферов и одно табличное пространство размером 4 Кбайта с маленьким пулом буферов.

Примечание: Табличные пространства каталога могут использовать только размер страницы 4 Кбайта. По существу, менеджер баз данных всегда требует существования временного системного табличного пространства 4 Кбайта, чтобы выполнять реорганизацию таблиц каталога.

- Как правило, определение нескольких временных табличных пространств с одним размером страницы не дает никаких преимуществ.
- Для временных табличных пространств почти всегда лучше выбирать SMS, чем DMS, поскольку:
  - При создании временной таблицы с применением DMS возникает большая дополнительная нагрузка, по сравнению с SMS.
  - В SMS дисковое пространство выбирается по требованию, а в DMS оно должно быть выделено заранее. Предварительное выделение может оказаться сложным, поскольку во временных табличных пространствах хранятся кратковременные данные, для которых в пиковые моменты может потребоваться гораздо больше памяти, чем в среднем. В DMS пиковую необходимую память нужно выделить заранее, а в SMS в периоды средней нагрузки эту память можно использовать для других целей.
  - Менеджер баз данных пытается хранить страницы временных таблиц в памяти, а не записывать их на диск. В результате преимущество DMS в производительности становится менее значительным.

Понятия, связанные с данным:

- “Проектирование табличного пространства” на стр. 118
- “Пространство, управляемое системой” на стр. 122

Ссылки, связанные с данной темой:

- “REORG INDEXES/TABLE Command” в книге *Command Reference*

Проектирование табличного пространства каталога

Для каталогов баз данных рекомендуется табличное пространство SMS, поскольку:

- Каталог баз данных состоит из множества таблиц разных размеров. При использовании табличного пространства DMS для каждого табличного

объекта выделяется минимум два экстента. В зависимости от выбранного размера экстента можно получить значительный объем выделенного, но неиспользуемого места. При использовании табличного пространства DMS следует выбирать маленький размер экстента (от двух до четырех страниц); в противном случае используйте табличное пространство SMS.

- В таблицах каталога есть столбцы больших объектов (LOB). Данные больших объектов не хранятся в пуле буферов вместе с остальными данными, а при каждом обращении считываются с диска. Чтение больших объектов с диска уменьшает производительность. Поскольку обычно у файловой системы есть собственный кэш, использование табличного пространства SMS или табличного пространства DMS, построенного на контейнерах-файлах, дает возможность избежать ввода/вывода, если к этому большому объекту уже были обращения.

Из-за этих особенностей для каталогов лучше выбирать табличное пространство SMS.

Еще один фактор, который нужно учесть - потребуется ли увеличивать табличное пространство каталога. Некоторые платформы поддерживают увеличение места, выделенного для контейнеров SMS, и для увеличения табличного пространства SMS можно воспользоваться восстановлением с перенаправлением, однако добавлять новые контейнеры проще при использовании табличного пространства DMS.

Понятия, связанные с данным:

- “Определение таблиц системного каталога” в книге *Руководство администратора: Реализация*
- “Проектирование табличного пространства” на стр. 118
- “Пространство, управляемое системой” на стр. 122
- “Пространство, управляемое базой данных” на стр. 125

Оптимизация производительности табличного пространства при размещении данных на дисковом массиве

В этом разделе описывается, как повысить производительность, если данные находятся на устройствах RAID (Redundant Array of Independent Disks - дублированный массив независимых дисков).

Порядок действий:

Для каждого табличного пространства, использующего устройство RAID, нужно:

- Определить для табличного пространства один контейнер (устройство RAID).

- Сделать EXTENTSIZE табличного пространства равным или кратным размеру полосы RAID.
- Убедитесь, что значение PREFETCHSIZE табличного пространства:
  - равно или кратно произведению размера полосы RAID и числа параллельных устройств RAID;
  - кратно EXTENTSIZE.
- Чтобы разрешить для табличного пространства параллельный ввод/вывод, используйте переменную реестра DB2_PARALLEL_IO.

DB2_PARALLEL_IO:

При чтении данных из контейнеров табличного пространства или записи в них, если в базе данных несколько контейнеров, DB2 может использовать параллельный ввод/вывод. Однако есть ситуации, когда имеет смысл включать параллельный ввод/вывод для табличных пространств с одним контейнером. Например, если контейнер создан на одном устройстве RAID, состоящем из нескольких физических дисков, имеет смысл выполнять параллельные требования чтения и записи.

Чтобы принудительно разрешить параллельный ввод/вывод для табличного пространства с одним контейнером, используйте переменную реестра DB2_PARALLEL_IO. Для этой переменной можно задать значение "*" (звездочка), что означает все табличные пространства, или список ID табличных пространств через запятую. Например:

```
db2set DB2_PARALLEL_IO=* {включает паралл. ввод/вывод для всех табл. пр-в}
db2set DB2_PARALLEL_IO=1,2,4,8 {включает паралл. ввод/вывод для табл. пр-в 1, 2, 4 и 8}
```

Чтобы изменения вступили в силу, после задания переменной реестра нужно остановить DB2 (**db2stop**) и запустить снова (**db2start**).

Переменная реестра DB2_PARALLEL_IO влияет также на табличные пространства с несколькими контейнерами. Если эта переменная реестра не задана, степень параллелизма ввода-вывода равна числу контейнеров в табличном пространстве. Если эта переменная реестра задана, степень параллелизма ввода-вывода равна результату деления размера предварительной выборки на размер экстента. Эту переменную реестра можно задать, если отдельные контейнеры табличного пространства хранятся на дисковом массиве RAID, где данные распределяются по нескольким физическим дискам.

Например, табличное пространство содержит два контейнера, а размер предварительной выборки в четыре раза больше размера экстента. Если эта переменная реестра не задана, запрос предварительной выборки для этого табличного пространства будет разбит на два запроса (каждый для двух экстентов). Если доступна предварительная выборка, два таких устройства могут выполнять эти запросы в параллельном режиме. Если эта переменная

реестра задана, запрос предварительной выборки для этого табличного пространства будет разбит на четыре запроса (каждый для одного экстента) и запросы могут выполняться четырьмя процессами предварительной выборки в параллельном режиме.

Если в этом примере для каждого из двух контейнеров выделен отдельный диск и задана эта переменная реестра, это может привести к конфликтам при обращении к дискам, так как два процесса предварительной выборки будут одновременно обращаться к одному диску. Однако если эти каждый из этих двух контейнеров распределен по нескольким дисках дискового массива, задание этой переменной реестра сделает возможным одновременное обращение к четырем разным дискам.

DB2_USE_PAGE_CONTAINER_TAG:

По умолчанию, DB2 хранит в первом экстенте каждого контейнера DMS (файла или устройства) тег контейнера. Тег контейнера - это метаданные DB2 для контейнера. В предыдущих версиях DB2 для хранения тега контейнера применялась первая страница, а не экстент, в результате чего для хранения тега в контейнере использовалось меньше места. (В предыдущих версиях DB2 переменная реестра DB2_STRIPED_CONTAINERS применялась для создания табличных пространств с тегами, размер которых был равен экстенту. Однако так как теперь это выполняется по умолчанию, данная переменная реестра больше не действует.)

Если переменной реестра DB2_USE_PAGE_CONTAINER_TAG присвоено значение ON, все вновь создаваемые контейнеры DMS создаются с тегом, размер которого равен странице, вместо тега, занимающего экстент (который задан по умолчанию). Действие этого значения не распространяется на существующие контейнеры, которые были созданы до изменения значения этой переменной реестра.

Присваивать этой переменной значение ON рекомендуется только при острой нехватке свободного пространства, либо для обеспечения совместимости с базами данных предыдущих версий.

Присвоение этой переменной значения ON может негативно отразиться на производительности, если в качестве контейнеров табличного пространства применяются устройства RAID. При использовании в качестве контейнеров табличных пространств устройств RAID рекомендуется создавать табличное пространство с размером экстентов, равным или кратным размеру полосы RAID. Если присвоить переменной реестра значение ON, то будет применяться тег контейнера размером в одну, и экстенты не будут совпадать с блоками чередования RAID. При выполнении запроса ввода/вывода может понадобиться

обращение к большему числу физических дисков, чем было бы оптимально. В этой связи пользователям настоятельно рекомендуется не задавать эту переменную реестра.

Для создания контейнеров с тегом, размер которого равен странице, присвойте этой переменной реестра значение ON, а затем перезапустите экземпляр:

```
db2set DB2_USE_PAGE_CONTAINER_TAG=ON
db2stop
db2start
```

Для того чтобы прекратить создание контейнеров с тегами, размер которых равен странице, сбросьте значение этой переменной реестра, а затем перезапустите экземпляр:

```
db2set DB2_USE_PAGE_CONTAINER_TAG=
db2stop
db2start
```

Центр управления и команды LIST TABLESPACE CONTAINERS и GET SNAPSHOT FOR TABLESPACES не показывают размер тега, с которым был создан контейнер. Они показывают только метку ″файл″ или ″устройство″, в зависимости от того, как был создан контейнер. Чтобы проверить, с каким размером тега был создан контейнер, можно воспользоваться опцией /DTSF команды DB2DART, чтобы вывести данные о табличных пространствах и контейнерах, а затем посмотреть поле ″тип″ интересующего вас контейнера. При помощи API опроса контейнера (sqlbftcq и sqlbtcq) можно создать простую программу, выводящую тип контейнера.

Понятия, связанные с данным:

- “Проектирование табличного пространства” на стр. 118

Ссылки, связанные с данной темой:

- “Переменные системной среды” в книге *Руководство администратора: Производительность*

Рекомендации по выбору табличных пространств для таблиц

При определении, как отображать таблицы в табличные пространства, необходимо учитывать:

- Разделение ваших таблиц.

  Как минимум, следует убедиться, что выбранное табличное пространство находится в группе разделов базы данных с нужным разделением.

- Количество данных в таблице.

  Если вы собираетесь хранить в табличном пространстве много маленьких таблиц, подумайте об использовании табличного пространства SMS.

Преимущества DMS при вводе/выводе и управлении пространством не настолько важны с маленькими таблицами. Такие преимущества SMS, как выделение места по одной странице за раз и только по мере надобности, более привлекательны для маленьких таблиц. Если одна из таблиц большая, или же вам нужен ускоренный доступ к данным в таблицах, рекомендуется использовать табличное пространство DMS с маленьким размером экстента.

Можно использовать по отдельному табличному пространству для всех очень больших таблиц, а все маленькие таблицы сгруппировать в одном табличном пространстве. Такое разделение позволяет также выбрать подходящий размер экстента в зависимости от использования табличного пространства.

- Тип данных в таблице.

  Допустим, у вас есть таблицы с редко используемыми хронологическими данными; возможно, конечный пользователь готов мириться с большим временем ответа для запросов к этим данных. В этом случае для хронологических таблиц можно использовать отдельное табличное пространство и назначить этому пространству менее дорогие физические устройства с более медленным временем доступа.

  Другой вариант - выбрать несколько важных таблиц, для данных которых требуется высокая доступность и короткое время ответа. Эти таблицы можно поместить в табличное пространство, назначенное быстрому физическому устройству, которое сможет поддерживать эти требования для важных данных.

  Используя табличные пространства DMS, можно распределить табличные данные по трем разным табличным пространствам: одно для индексных данных, одно для данных больших объектов и длинных полей и одно для обычных табличных данных. Это позволяет выбрать характеристики табличных пространств и физические устройства, на которых они будут размещаться, лучше всего подходящие для конкретных данных. Например, можно поместить данные индексов на самых быстрых устройствах из имеющихся и в результате получить существенное улучшение производительности. Если таблица разбита по табличным пространствам DMS и включено восстановление с повтором транзакций, надо предусмотреть для этих табличных пространств совместное резервное копирование и восстановление. Табличные пространства SMS не поддерживают этого типа распределения данных по табличным пространствам.

- Управление.

  Некоторые функции управления можно выполнять на уровне табличного пространства вместо уровня базы данных или таблицы. Например, можно сэкономить время и ресурсы, если выполнять резервное копирование табличного пространства, а не базы данных. Это позволяет часто создавать резервные копии табличных пространств с большим объемом изменений, а резервные копии табличных пространств с небольшим объемом изменений создавать только изредка.

Можно восстановить базу данных или табличное пространство. Если в табличных пространствах нет лишних таблиц, вы можете восстановить только нужную часть базы данных за меньшее время.

Хорошим подходом является группировка связанных таблиц в набор табличных пространств. Эти таблицы могут быть связаны реляционными ограничениями или другими определенными ограничениями.

Если вам часто приходится отбрасывать и переопределять некоторую таблицу, возможно, ее следует определить в отдельном табличном пространстве, так как отбрасывать табличное пространство DMS эффективнее, чем отбрасывать таблицу.

Понятия, связанные с данным:

- “Группы разделов базы данных” на стр. 108
- “Пространство, управляемое системой” на стр. 122
- “Пространство, управляемое базой данных” на стр. 125
- “Сравнение табличных пространств SMS и DMS” на стр. 146

Глава 6. Проектирование распределенных баз данных

Единица работы

В DB2® транзакции обычно называют *единицами работы*. Единица работы - это восстановимая последовательность операций процесса прикладной программы. Единица работы используется менеджером баз данных для обеспечения согласованного состояния базы данных. Все операции чтения и записи для базы данных выполняются внутри единицы работы.

Например, банковская транзакция может включать в себя перевод денег с накопительного счета на расходный счет. После того, как прикладная программа вычтет величину перевода с накопительного счета, эти два счета будут в несогласованном состоянии до тех пор, пока эта же величина не будет добавлена к расходному счету. После выполнения *обоих* шагов достигается точка согласованности. Изменения могут быть приняты и стать доступными для других прикладных программ.

Единица работы начинается при передаче базе данных первого оператора SQL. Прикладная программа должна завершить эту единицу работы с помощью оператора COMMIT или ROLLBACK. Оператор COMMIT делает постоянными все изменения, сделанные в единице работы. Оператор ROLLBACK удаляет эти изменения из базы данных. Если прикладная программа завершается нормально, но не выполняет явно ни один из этих операторов, для единицы работы автоматически выполняется принятие. Если прикладная программа завершается аварийно в середине единицы работы, для этой единицы работы выполняется откат. Если операция COMMIT или ROLLBACK запущена, остановить ее нельзя. Для некоторых многопоточных прикладных программ и некоторых операционных систем (например, Windows), если прикладная программа завершается нормально, но не выполняет явно ни один из этих операторов, для единицы работы автоматически выполняется откат. Рекомендуется, чтобы прикладные программы всегда явно выполняли принятие или откат в конце единицы работы. Если часть единицы работы не выполнена успешно, для изменений выполняется откат и использовавшиеся таблицы остаются в том состоянии, в котором они были перед началом транзакции. Это гарантирует, что требования не будут потеряны или выполнены дважды.

Ссылки, связанные с данной темой:

- “COMMIT statement” в книге *SQL Reference, Том 2*
- “ROLLBACK statement” в книге *SQL Reference, Том 2*

Изменение в транзакции одной базы данных

Самая простая форма транзакции выполняет чтение или запись только для одной базы данных в одной единице работы. Такой тип доступа к базе данных называется *удаленной единицей работы*.

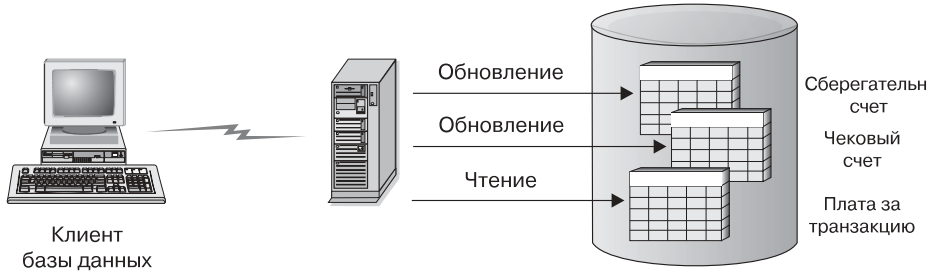

Рисунок 47. Использование в транзакции одной базы данных

На рис. 47 показан клиент базы данных, на котором выполняется прикладная программа денежных переводов; она обращается к базе данных, содержащей таблицы накопительных и расходных счетов, а также план оплаты банковских услуг. Эта программа должна:

- Получить значение переводимой суммы от пользовательского интерфейса
- Вычесть эту величину с накопительного счета и определить новый остаток
- Прочитать план оплаты, чтобы определить стоимость перевода денег для накопительного счета с данным остатком
- Вычесть стоимость перевода с накопительного счета
- Добавить переводимую сумму к расчетному счету
- Выполнить принятие транзакции (единицы работы).

Порядок действий:

Для конфигурирования такой программы надо:

1. Создать таблицы для накопительного счета, расчетного счета и плана оплаты в одной базе данных
2. Для физически удаленной базы данных: задать для сервера баз данных использование подходящего протокола связи
3. Для физически удаленной базы данных - внести в каталог узел и базу данных, чтобы определить эту базу для сервера
4. Прекомпилировать прикладную программу, задав соединение типа 1; для этого надо задать CONNECT 1 (значение по умолчанию) в команде PRECOMPILE PROGRAM.

Понятия, связанные с данным:

- “Единица работы” на стр. 163

Задачи, связанные с данной темой:

- “Изменение одной базы данных в транзакции с несколькими базами данных” на стр. 165
- “Изменение в транзакции нескольких баз данных” на стр. 166

Ссылки, связанные с данной темой:

- “PRECOMPILE Command” в книге *Command Reference*

Использование в транзакции нескольких баз данных

При использовании в одной транзакции нескольких баз данных требования к настройке и управлению средой будут различными в зависимости от числа изменяемых в транзакции баз данных.

Изменение одной базы данных в транзакции с несколькими базами данных

Когда данные распределены по нескольким базам данных, может понадобиться читать данные из одной или нескольких баз данных и изменять данные в одной базе данных. Этот тип доступа может выполняться в одной единице работы (транзакции).

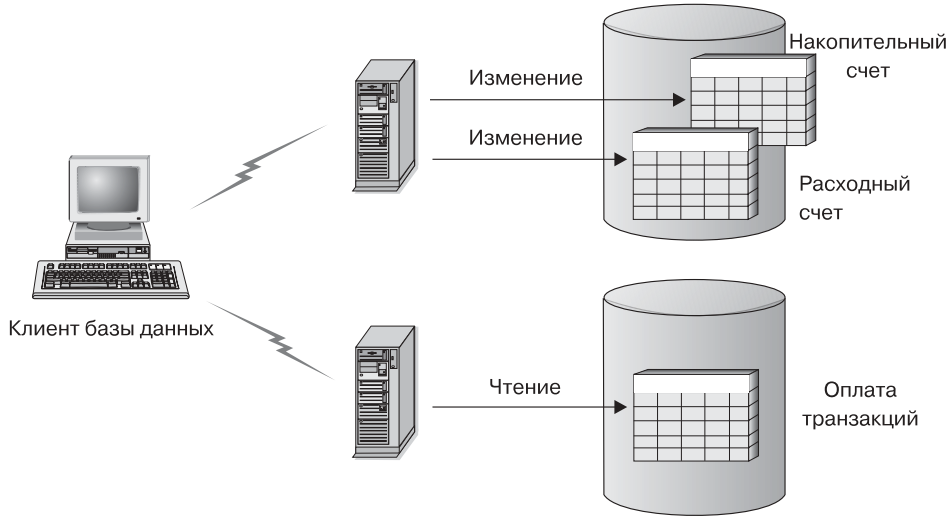

Рисунок 48. Использование в транзакции нескольких баз данных

На рис. 48 на стр. 165 показан клиент базы данных, на котором выполняется прикладная программа денежных переводов; она обращается к двум серверам баз данных: на одном хранятся накопительные и расходные счета, а на другом - план оплаты банковских услуг.

Порядок действий:

Чтобы сконфигурировать прикладную программу денежных переводов для этой среды, нужно:

1. Создайте необходимые таблицы в подходящих базах данных
2. Для физически удаленных баз данных: задать для серверов баз данных использование подходящих протоколов связи
3. Для физически удаленных баз данных: внести в каталог узлы и базы данных, чтобы определить эти базы для серверов баз данных
4. Прекомпилировать прикладную программу, задав соединение типа 2 (для этого надо задать CONNECT 2 в команде PRECOMPILE PROGRAM) и однофазное принятие (для этого надо задать SYNCPOINT ONEPHASE в команде PRECOMPILE PROGRAM)

Если база данных находится на сервере баз данных хоста или iSeries, для связи с этим сервером требуется DB2 Connect.

Понятия, связанные с данным:

- “Единица работы” на стр. 163

Задачи, связанные с данной темой:

- “Изменение в транзакции одной базы данных” на стр. 164
- “Изменение в транзакции нескольких баз данных” на стр. 166

Ссылки, связанные с данной темой:

- “PRECOMPILE Command” в книге *Command Reference*

Изменение в транзакции нескольких баз данных

Когда данные распределены по нескольким базам данных, может понадобиться читать и изменять данные в нескольких базах данных в одной транзакции. Такой тип доступа к базе данных называется *многоузловым изменением*.

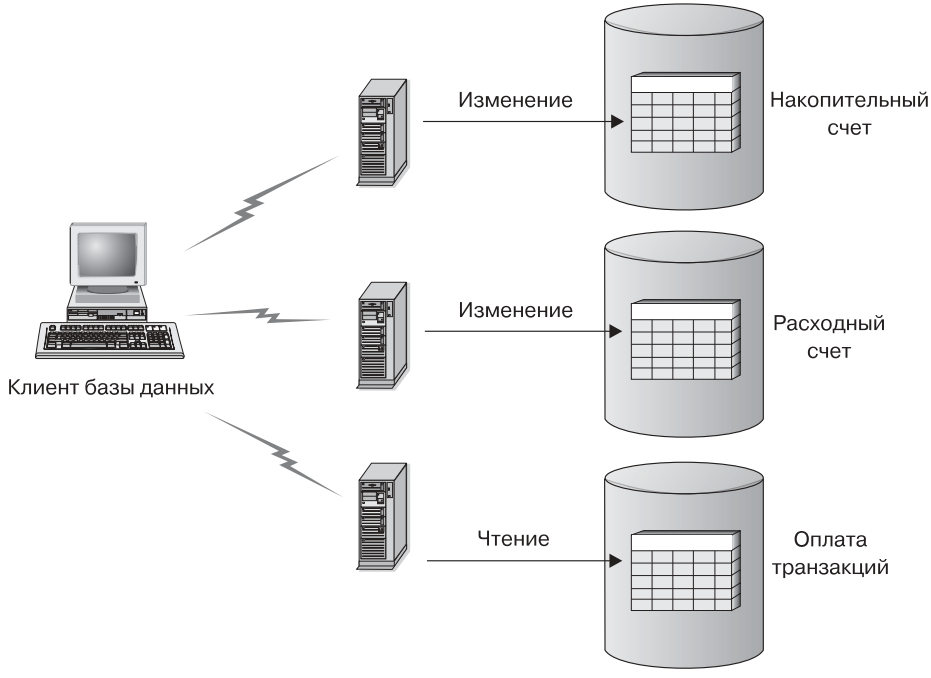

Рисунок 49. Изменение нескольких баз данных в одной транзакции

На рис. 49 показан клиент базы данных, на котором выполняется прикладная программа денежных переводов; она обращается к трем серверам баз данных: на одном хранятся накопительные счета, на другом - расходные счета, а на третьем - план оплаты банковских услуг.

Порядок действий:

Чтобы сконфигурировать прикладную программу денежных переводов для этой среды, есть две возможности:

1. С помощью менеджера транзакций DB2 (TM):
   a. Создайте необходимые таблицы в подходящих базах данных
   b. Для физически удаленных баз данных: задать для серверов баз данных использование подходящих протоколов связи
   c. Для физически удаленных баз данных: внести в каталог узлы и базы данных, чтобы определить эти базы для серверов баз данных
   d. Прекомпилировать прикладную программу, задав соединение типа 2 (для этого надо задать CONNECT 2 в команде PRECOMPILE PROGRAM) и двухфазное принятие (для этого надо задать SYNCPOINT TWOPHASE в команде PRECOMPILE PROGRAM)
   e. Настройте менеджер транзакций DB2 (TM).

2. С помощью XA-совместимого менеджера транзакций:
   a. Создайте необходимые таблицы в подходящих базах данных
   b. Для физически удаленных баз данных: задать для серверов баз данных использование подходящих протоколов связи
   c. Для физически удаленных баз данных: внести в каталог узлы и базы данных, чтобы определить эти базы для серверов баз данных
   d. Прекомпилировать прикладную программу, задав соединение типа 2 (для этого надо задать CONNECT 2 в команде PRECOMPILE PROGRAM) и однофазное принятие (для этого надо задать SYNCPOINT ONEPHASE в команде PRECOMPILE PROGRAM)
   e. Настройте XA-совместимый менеджер транзакций для работы с базами данных DB2.

Понятия, связанные с данным:

- “Единица работы” на стр. 163
- “Менеджер транзакций DB2” на стр. 168

Задачи, связанные с данной темой:

- “Изменение в транзакции одной базы данных” на стр. 164
- “Изменение одной базы данных в транзакции с несколькими базами данных” на стр. 165

Ссылки, связанные с данной темой:

- “PRECOMPILE Command” в книге *Command Reference*

Менеджер транзакций DB2

Менеджер транзакций DB2® приписывает транзакциям идентификаторы, отслеживает их ход и несет ответственность за их успех или неудачу. Менеджер транзакций входит в состав DB2 Universal Database™ (UDB) и DB2 Connect™. Менеджер транзакций DB2 хранит информацию о транзакциях в заданной базе данных TM.

Менеджер баз данных поддерживает функции менеджера транзакций, которые могут использоваться для координации изменений нескольких баз данных в одной единице работы. Клиент базы данных автоматически координирует единицу работы и использует *базу данных менеджера транзакций* для регистрации каждой транзакции и отслеживания состояния ее выполнения.

С базами данных DB2 может применяться менеджер транзакций DB2. Если кроме баз данных DB2 у вас есть и другие ресурсы, которые вы хотите использовать в двухфазном принятии транзакций, можно использовать для баз данных XA-совместимый менеджер транзакций.

Понятия, связанные с данным:

- “Единица работы” на стр. 163
- “Конфигурация менеджера транзакций DB2” на стр. 169
- “Двухфазное принятие (Two-phase commit)” на стр. 175

Конфигурация менеджера транзакций DB2

Если используется менеджер транзакций, совместимый с XA (например, IBM® WebSphere, BEA Tuxedo или Microsoft® Transaction Server), то выполните инструкции по настройке для данного продукта.

Если для координации транзакций используется DB2® UDB для систем на основе UNIX или операционных систем Windows®, то должны быть удовлетворены некоторые требования к конфигурации. Если для связи используется только протокол TCP/IP и в транзакциях участвуют только серверы баз данных DB2 UDB for Unix, Windows, iSeries™ V5, z/OS™ или OS/390®, конфигурирование будет несложным.

DB2 для Unix и Windows и DB2 для z/OS, OS/390 и iSeries V5 с применением TCP/IP

Если в вашей среде выполняются следующие условия, конфигурирование многоузлового изменения будет несложным.

- Для связи со всеми удаленными серверами баз данных (включая DB2 UDB для z/OS, OS/390 и iSeries V5) используется только TCP/IP.
- В транзакции используются только серверы баз данных DB2® UDB для систем на основе UNIX, операционных систем Windows, z/OS, OS/390 и iSeries V5.
- Не сконфигурирован менеджер точек синхронизации (SPM) DB2 Connect™.

  Менеджер точек синхронизации DB2 Connect конфигурируется автоматически во время создания экземпляра DB2; он требуется, если:

  – Для операций многоузлового изменения используется связь SNA с серверами баз данных хоста или iSeries.

    Примечание: Возможно, лучше перейти на TCP/IP, так поддержка SNA может отсутствовать в следующих выпусках DB2. Для настройки SNA необходимо глубокое знание конфигурации, а в самом процесс настройки часто возникают ошибки. TCP/IP проще настроить, требует меньше затрат на содержание и обеспечивает более высокую производительность.

  – Двухфазное принятие координируется менеджером транзакций, совместимым с XA (например, IBM WebSphere).

    Это верно как для связи SNA, так и для связи TCP/IP с серверами баз данных хоста или iSeries. Если вашей среде не требуется менеджер точек синхронизации DB2 Connect, его можно отключить, выполнив команду

`db2 update dbm cfg using spm_name NULL` сервере DB2 Connect. Затем остановите и снова запустите менеджер баз данных.

База данных, которая будет использоваться в качестве базы данных менеджера транзакций, задается на клиенте базы данных параметром конфигурации *tm_database*. При задании этого параметра конфигурации учтите следующие факторы:

- Менеджер транзакций может быть:
  - База данных DB2 UDB для UNIX или Windows версии 8
  - База данных DB2 для z/OS и OS/390 версии 7 или DB2 для OS/390 версий 5 и 6
  - База данных DB2 для iSeries V5

    Серверы DB2 для z/OS, OS/390 и iSeries V5 рекомендуется использовать в качестве баз данных менеджеров транзакций. Системы z/OS, OS/390 и iSeries V5, в целом, более надежны, чем серверы рабочих станций, и на них меньше вероятность случайных отключений питания, перезагрузок и так далее. Поэтому будут более надежными и журналы восстановления, используемые при ресинхронизации событий.
- Если для параметра конфигурации *tm_database* задано значение `1ST_CONN`, в качестве базы данных менеджера транзакций будет использоваться первая база данных, с которой соединится прикладная программа.

  При использовании значения `1ST_CONN` необходима осторожность. Используйте его, только если легко обеспечить правильное внесение в каталог всех используемых баз данных; то есть если:
  - Клиент базы данных, запускающий транзакцию, принадлежит тому же экземпляру, что и участвующие в транзакции базы данных, включая базу данных менеджера транзакций.

  Учтите, что если прикладная программа попытается отсоединиться от базы данных, используемой в качестве базы данных менеджера транзакций, вы получите предупреждение, а соединение будет удерживаться до принятия единицы работы.

Другие среды

Если в вашей среде:

- Для связи с удаленными серверами баз данных используется не только TCP/IP (например, NETBIOS)
- Выполняются обращения к DB2 for AS/400® V4 или DB2 for VM&VSE
- Выполняются обращения с помощью SNA к DB2 for z/OS, OS/390 или iSeries V5
- Для доступа к серверам баз данных хоста или iSeries используется менеджер точек синхронизации DB2 Connect,

то шаги конфигурирования многоузлового изменения будут сложнее.

База данных, которая будет использоваться в качестве базы данных менеджера транзакций, задается на клиенте базы данных параметром конфигурации *tm_database*. При задании этого параметра конфигурации учтите следующие факторы, если применяется клиент времени выполнения DB2 версии 8:

- Базой данных менеджера транзакций может быть база данных DB2 UDB для UNIX, Windows, z/OS, OS/390 или iSeries V5. Базой данных менеджера транзакций не может быть DB2 UDB для Unix, Windows или OS/2® версии 7.
- Будут применяться ID пользователя и пароль первой базы данных.
- Если для параметра конфигурации *tm_database* задано значение `1ST_CONN`, в качестве базы данных менеджера транзакций будет использоваться первая база данных, с которой соединится прикладная программа.

  При использовании значения `1ST_CONN` необходима осторожность. Используйте его, только если легко обеспечить правильное внесение в каталог всех используемых баз данных; то есть если:

  – Клиент базы данных, запускающий транзакцию, принадлежит тому же экземпляру, что и участвующие в транзакции базы данных, включая базу данных менеджера транзакций.

  – Соединение должно устанавливаться в первую очередь с базой данных, которая может применяться в качестве базы данных менеджера транзакций.

  Учтите, что если прикладная программа попытается отсоединиться от базы данных, используемой в качестве базы данных менеджера транзакций, вы получите предупреждение, а соединение будет удерживаться до принятия единицы работы.

Параметры конфигурации

При конфигурировании среды надо рассмотреть следующие параметры конфигурации.

Параметры конфигурации менеджера баз данных

- *tm_database*

  Этот параметр задает имя базы данных Transaction Manager (TM) для каждого экземпляра DB2.

- *spm_name*

  Этот параметр задает имя экземпляра менеджера точек синхронизации DB2 Connect для менеджера баз данных. Для успешной ресинхронизации это имя должно быть уникальным в сети.

- *resync_interval*

  Этот параметр задает интервал (в секундах), после которого менеджер транзакций DB2, менеджер баз данных сервера DB2 и менеджер точек

синхронизации DB2 Connect или менеджер точек синхронизации DB2 UDB должны попытаться восстановить все незавершенные неоднозначные транзакции.

- *spm_log_file_sz*

  Этот параметр задает размер (в 4-Кбайтных страницах) файла журнала SPM.

- *spm_max_resync*

  Этот параметр задает число агентов, которые могут одновременно выполнять операции ресинхронизации.

- *spm_log_path*

  Этот параметр задает путь журналов для файлов журналов SPM.

Параметры конфигурации базы данных

- *maxappls*

  Этот параметр задает максимальное разрешенное число активных прикладных программ. Его значение должно быть больше или равно числу соединенных прикладных программ, плюс число таких прикладных программ, для которых может одновременно выполняться двухфазное принятие или откат, плюс ожидаемое число возможных неоднозначных транзакций в любой момент времени.

- *autorestart*

  Этот параметр конфигурации определяет, будет ли при необходимости автоматически выполняться процедура перезапуска базы данных (RESTART DATABASE). Значение по умолчанию - YES (перезапуск разрешен).

  Чтобы можно было использовать базу данных, содержащую неоднозначные транзакции, для нее должна быть выполнена операция перезапуска. Если параметр *autorestart* имеет значение NO, после прекращения последнего соединения с базой данных следующее соединение будет неудачным и потребуется явно запустить команду RESTART DATABASE. Это условие сохраняется, пока неоднозначные транзакции не будут удалены операцией ресинхронизации менеджера транзакций или запущенной администратором эвристической операцией. Если при запуске команды RESTART DATABASE в базе данных есть неоднозначные транзакции, выдается сообщение. Администратор может затем использовать команду LIST INDOUBT TRANSACTIONS и другие команды процессора командной строки, чтобы получить информацию об этих неоднозначных транзакциях.

Понятия, связанные с данным:

- “Менеджер транзакций DB2” на стр. 168

Задачи, связанные с данной темой:

- “Конфигурирование IBM TXSeries CICS” на стр. 203
- “Конфигурирование IBM TXSeries Encina” на стр. 203

- “Конфигурирование BEA Tuxedo” на стр. 205
- “Настройка IBM WebSphere Application Server” на стр. 203

Ссылки, связанные с данной темой:

- “Параметр конфигурации Путь файлов журнала менеджера точек синхронизации - spm_log_path” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Разрешение автоматического перезапуска - autorestart” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Максимальное число активных прикладных программ - maxappls” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Интервал ресинхронизации транзакций - resync_interval” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Имя базы данных менеджера транзакций - tm_database” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Имя менеджера точек синхронизации - spm_name” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Размер файлов журнала менеджера точек синхронизации - spm_log_file_sz” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Предельное число агентов ресинхронизации менеджера точек синхронизации - spm_max_resync” в книге *Руководство администратора: Производительность*

Изменение базы данных с хоста или клиента iSeries

Прикладные программы, выполняемые в системе хоста или iSeries могут обращаться к данным, находящимся на серверах базы данных DB2 UDB для Unix и Windows. Для работы с этими данными может применяться как TCP/IP, так и SNA. Для применения SNA на сервере DB2 UDB необходимо настроить менеджер точек синхронизации (SPM). Для применения TCP/IP SPM не нужен.

Примечание: Если вы пользуетесь SNA, то возможно, лучше перейти на TCP/IP, так поддержка SNA может отсутствовать в следующих выпусках DB2. Для настройки SNA необходимо глубокое знание конфигурации, а в самом процесс настройки часто возникают ошибки. TCP/IP проще настроить, требует меньше затрат на содержание и обеспечивает более высокую производительность.

Сервер баз данных, к которому обращаются с клиента баз данных хоста или iSeries, не обязан быть локальным для рабочей станции, на которой установлен менеджер точек синхронизации DB2. Клиент баз данных хоста или iSeries может

соединяться с сервером DB2 UDB, используя рабочую станцию менеджера точек синхронизации DB2 в качестве временного шлюза. Это позволяет установить рабочую станцию менеджера точек синхронизации DB2 в защищенной среде, в то время как сами серверы DB2 UDB будут удаленными. Это также позволяет использовать базу данных DB2 Common Server Версии 2 в многоузловом изменении с клиентов баз данных хоста или iSeries.

Порядок действий:

На рабочей станции, к которой будут напрямую обращаться прикладные программы хоста или iSeries:

1. Для поддержки многоузлового изменения с клиентов баз данных хоста или iSeries установите DB2 Universal Database Server Edition.
2. Создайте экземпляр базы данных на той же системе. Можно, например, использовать экземпляр по умолчанию, DB2, или создать новый экземпляр с помощью команды:

   ```
   db2icrt мой_экземпляр
   ```

3. Задайте требуемую лицензионную информацию.
4. Убедитесь, что в значении реестра DB2COMM указано APPC и/или TCP/IP.
5. Для применения SNA на хосте или клиенте iSeries необходимо настроить менеджер точки синхронизации:
   a. Правильно сконфигурируйте связь SNA. Если используются поддерживаемые IBM продукты SNA, необходимые для менеджера точек синхронизации DB2 профили SNA создаются автоматически на основе значения параметра конфигурации менеджера баз данных *spm_name*. Для других поддерживаемых стеков SNA требуется ручная настройка.
   b. Определите значение, которое нужно задать для параметра конфигурации менеджера баз данных *spm_name*. При создании экземпляра DB2 для этого параметра задается значение, полученное на основе имени хоста TCP/IP для этого компьютера. Если это значение подходит и уникально в вашей среде, не меняйте его.
   c. При необходимости измените значение параметра *spm_name* на сервере DB2 Universal Database, используя команду UPDATE DATABASE MANAGER CONFIGURATION.
6. Сконфигурируйте связь, чтобы эта рабочая станция DB2 могла в случае необходимости соединяться с удаленными серверами DB2 UDB.
7. Сконфигурируйте связь, чтобы удаленные серверы DB2 UDB могли соединяться с этим сервером DB2.
8. Остановите и снова запустите менеджер баз данных на сервере DB2 Universal Database.

   Необходимо, чтобы с этой рабочей станции DB2 UDB можно было соединяться с удаленными серверами DB2 UDB.

На каждом удаленном сервере DB2 UDB, к которому будет обращаться клиент базы данных хоста или iSeries:

1. Сконфигурируйте связь, чтобы рабочая станция удаленного сервера DB2 UDB с менеджером точек синхронизации DB2 могла соединяться с этим сервером DB2 UDB.
2. Остановите и снова запустите менеджер баз данных.

Ссылки, связанные с данной темой:

- "Параметр конфигурации Имя менеджера точек синхронизации - spm_name" в книге *Руководство администратора: Производительность*
- "Переменные связи" в книге *Руководство администратора: Производительность*

Двухфазное принятие (Two-phase commit)

На рис. 50 на стр. 176 показаны шаги многоузлового изменения. Понимание того, как выполняется управление транзакциями, поможет вам исправлять ошибки, возникшие в процессе двухфазного принятия.

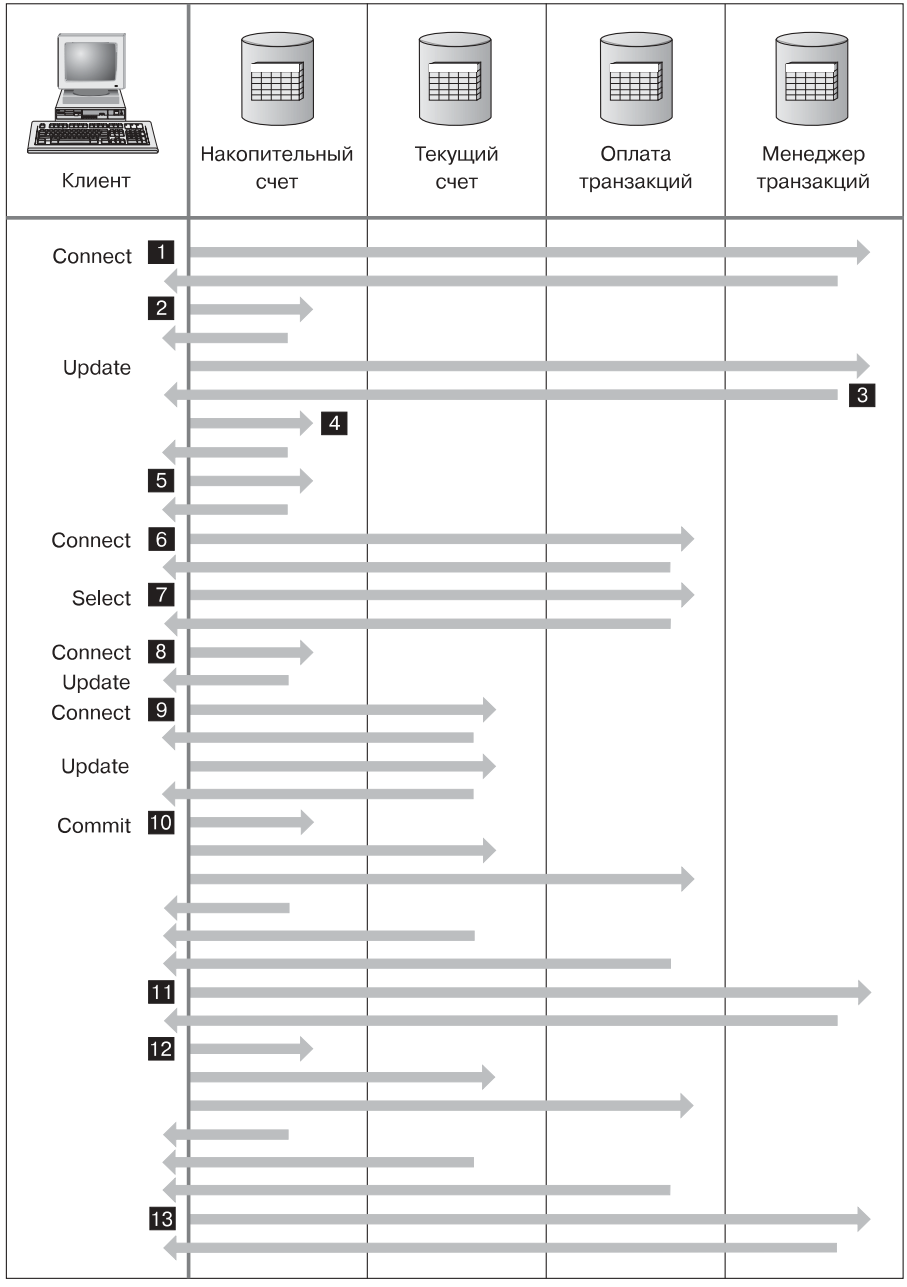

Рисунок 50. Изменение нескольких баз данных

0 Прикладная программа подготавливается для двухфазного принятия.

Для этого могут использоваться опции прекомпиляции. Для этого также может использоваться конфигурация CLI (интерфейс уровня вызовов) DB2®.

1 Когда клиенту базы данных нужно соединиться с базой данных SAVINGS_DB, он сначала соединяется с базой данных менеджера транзакций. База данных менеджера транзакций возвращает подтверждение клиенту баз данных. Если параметр конфигурации менеджера баз данных *tm_database* имеет значение `1ST_CONN`, на время работы этого экземпляра прикладной программы в качестве SAVINGS_DB используется база данных менеджера транзакций.

2 Установлено соединение с базой данных SAVINGS_DB и получено подтверждение.

3 Клиент базы данных начал изменять таблицу SAVINGS_ACCOUNT. Это начинает единицу работы. База данных менеджера транзакций отвечает клиенту базы данных, сообщая ID транзакции для этой единицы работы. Обратите внимание на то, что регистрация единицы работы происходит при выполнении первого оператора SQL в рабочей единице, а не при установлении соединения.

4 Получив ID транзакции, клиент базы данных регистрирует эту единицу работы в базе данных, содержащей таблицу SAVINGS_ACCOUNT. Клиенту посылается ответ, сообщающий об успешной регистрации единицы работы.

5 Выдается оператор SQL для базы данных SAVINGS_DB; он обрабатывается обычным образом. При работе с встроенными в программу операторами SQL ответ для каждого оператора возвращается в SQLCA.

6 ID транзакции регистрируется в базе данных FEE_DB, содержащей таблицу TRANSACTION_FEE, при первом обращении к этой базе данных в данной единице работы.

7 Все операторы SQL для базы данных FEE_DB обрабатываются обычным образом.

8 При необходимости при установлении соединения для базы данных SAVINGS_DB могут выполняться дополнительные операторы SQL. Поскольку единица работы уже зарегистрирована в базе данных SAVINGS_DB **4**, клиенту базы данных не нужно опять выполнять шаг регистрации.

9 Для соединения с базой данных CHECKING_DB и ее использования применяются те же правила, описанные в **6** и **7**.

10 Когда клиент базы данных запрашивает принятие единицы работы, всем базам данных, принимающим участие в этой транзакции,

посылается сообщение *prepare*. Каждая база данных помещает запись ″PREPARED″ в свой файл журнала и отвечает клиенту базы данных.

11 Получив положительные ответы от всех баз данных, клиент базы данных посылает сообщение базе данных менеджера транзакций, информируя ее, что единица работы теперь готова к выполнению принятия (PREPARED). База данных менеджера транзакций помещает запись ″PREPARED″ в свой файл журнала и отвечает клиенту, сообщая ему, что может быть начата вторая фаза процесса принятия.

12 Во время второй фазы процесса принятия клиент базы данных посылает сообщение всем базам данных, принимающим участие в этой транзакции, указывая им, что нужно выполнить принятие. Каждая база данных помещает запись ″COMMITTED″ в свой файл журнала и освобождает все блокировки, удерживаемые для этой единицы работы. Завершив принятие изменений, база данных посылает ответ клиенту.

13 Получив положительные ответы от всех баз данных, принимающих участие в этой транзакции, клиент базы данных посылает сообщение базе данных менеджера транзакций, информируя ее, что единица работы завершена. Затем база данных менеджера транзакций помещает запись ″COMMITTED″ в свой файл журнала, указывающую, что единица работы завершена, и отвечает клиенту, сообщая ему, что операция завершена.

Понятия, связанные с данным:

- “Единица работы” на стр. 163
- “Менеджер транзакций DB2” на стр. 168

Восстановление после ошибок при двухфазном принятии

Восстановление после ошибок - это обычная задача, связанная с созданием прикладных программ, управлением системой, управлением базой данных и управлением системой. При распределении баз данных по нескольким удаленным серверам возрастает вероятность ошибок, вызванных ошибками сети или ошибками связи. Для обеспечения целостности данных менеджер баз данных использует двухфазный процесс принятия. Далее объясняется, как менеджер баз данных обрабатывает ошибки, возникшие в процессе двухфазного принятия:

- **Ошибка на первой фазе**

  Если база данных сообщает, что она не смогла подготовиться к принятию единицы работы, на второй фазе принятия клиент базы данных выполняет откат этой единицы работы. В этом случае базе данных менеджера транзакций *не* посылается сообщение о подготовке.

На второй фазе клиент посылает сообщение об откате всем принимающим участие в транзакции базам данных, которые успешно подготовили принятие на первой фазе. Затем каждая база данных помещает запись ″ABORT″ в свой файл журнала и освобождает все блокировки, удерживаемые для этой единицы работы.

- **Ошибка на второй фазе**

  Обработка ошибки на этом этапе зависит от операции, выполняемой для транзакции на второй фазе (принятие или откат). Если на первой фазе возникла ошибка, на второй фазе для транзакции будет выполняться откат.

  Если одной из принимающих участие в транзакции баз данных не удалось выполнить принятие единицы работы (возможно, из-за ошибки связи), база данных менеджера транзакций попытается еще раз выполнить принятие в базе данных, в которой возникла ошибка. Однако прикладной программе в SQLCA будет передана информация о том, что принятие было успешным. DB2® обеспечит принятие непринятых транзакций на сервере баз данных. Для задания срока ожидания между попытками базы данных менеджера транзакций выполнить принятие единицы работы используется параметр конфигурации менеджера баз данных *resync_interval*. Сервер баз данных удерживает все блокировки, пока единица работы не будет принята.

  Если возникла ошибка базы данных менеджера транзакций, она ресинхронизирует единицу работы при своем перезапуске. Процесс ресинхронизации попытается завершить все *неоднозначные транзакции*, то есть транзакции, для которых была завершена первая, но не завершена вторая фаза процесса принятия. Для выполнения ресинхронизации менеджер баз данных, связанный с базой данных менеджера транзакций, выполняет следующие операции:

  1. Соединяется с базами данных, которые были подготовлены к принятию (″PREPARED″) на первой фазе процесса принятия.
  2. Пытается выполнить принятие неоднозначных транзакций в этих базах данных. (Если неоднозначные транзакции не удалось найти, менеджер баз данных предполагает, что база данных успешно выполнила принятие транзакций на второй фазе процесса принятия.)
  3. Выполняет принятие неоднозначных транзакций в базе данных менеджера транзакций после того, как все неоднозначные транзакции были приняты в базах данных, принимающих участие в транзакции.

  Если в одной из принимающих участие в транзакции баз данных возникла ошибка и эта база данных была перезапущена, менеджер баз данных для этой базы данных запросит у базы данных менеджера транзакций информацию о состоянии этой транзакции, чтобы узнать, нужно ли выполнить откат этой транзакции. Если транзакция не найдена в журнале, менеджер баз данных предполагает, что для транзакции был выполнен откат, и выполняет откат этой неоднозначной транзакции в этой базе данных. В противном случае база данных ожидает запроса принятия от базы данных менеджера транзакций.

Если транзакция координируется монитором обработки транзакций (менеджером транзакций, совместимым с XA), база данных будет всегда зависеть от монитора обработки транзакций, инициализирующего ресинхронизацию.

Если по каким-то причинам вы не можете ждать, пока менеджер транзакций автоматически разрешит неоднозначные транзакции, это можно сделать вручную. Этот процесс восстановления иногда называют ″принятием эвристического решения″.

Восстановление после ошибок, если autorestart=off

Если параметр конфигурации базы данных *autorestart* имеет значение OFF и в базах данных менеджера транзакций или менеджера восстановления есть неоднозначные транзакции, для запуска процесса ресинхронизации требуется команда RESTART DATABASE. Для ввода команды RESTART DATABASE из процессора командной строки используйте разные сеансы. Если из того же сеанса перезапустить другую базу данных, соединение, установленное предыдущей командой, будет прекращено и его придется перезапустить еще раз. Когда команда LIST INDOUBT TRANSACTIONS более не сообщает о неоднозначных транзакциях, завершите соединение с помощью команды TERMINATE.

Понятия, связанные с данным:

- “Двухфазное принятие (Two-phase commit)” на стр. 175

Задачи, связанные с данной темой:

- “Разрешение неоднозначных транзакций вручную” на стр. 194

Ссылки, связанные с данной темой:

- “Параметр конфигурации Разрешение автоматического перезапуска - autorestart” в книге *Руководство администратора: Производительность*
- “LIST INDOUBT TRANSACTIONS Command” в книге *Command Reference*
- “TERMINATE Command” в книге *Command Reference*
- “RESTART DATABASE Command” в книге *Command Reference*

Глава 7. Проектирование базы данных для применения XA-совместимых менеджеров транзакций

Если кроме баз данных DB2 у вас есть и другие ресурсы, которые вы хотите использовать в двухфазном принятии транзакций, можно использовать для баз данных XA-совместимый менеджер транзакций. Если в транзакциях принимают участие только базы данных DB2, надо использовать менеджер транзакций DB2, описанный в разделе “Изменение в транзакции нескольких баз данных” на стр. 166.

Следующие темы помогут вам освоить использование менеджера баз данных с такими XA-совместимыми менеджерами транзакций, как IBM WebSphere и BEA Tuxedo.

Информация о сервере Microsoft Transaction Server приведена в разделе *CLI Guide and Reference, Volume 1*.

Если вы используете или собираетесь использовать XA-совместимый менеджер транзакций, дополнительную информацию вы сможете найти на Web-сайте технической поддержки IBM:

`http://www.ibm.com/software/data/db2/udb/winos2unix/support`

На этой странице выберите ″DB2 Universal Database″, а затем выполните поиск по ключевому слову ″XA″, чтобы найти самую свежую информацию о XA-совместимых менеджерах транзакций.

Модель распределенной обработки транзакций X/Open

Модель распределенной обработки транзакций X/Open (Distributed Transaction Processing - DTP) включает в себя три взаимосвязанных компонента:

- Прикладная программа (AP)
- Менеджер транзакций (TM)
- Менеджеры ресурсов (RM)

Эта модель и отношения между ее компонентами показаны на рис. 51.

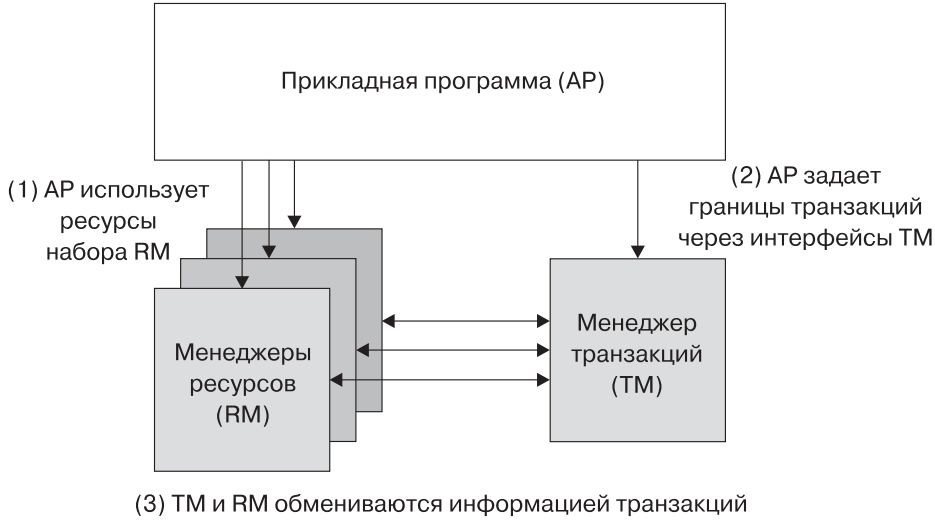

Рисунок 51. Модель распределенной обработки транзакций X/Open (DTP)

Прикладная программа (AP)

Прикладная программа определяет границы транзакции и входящие в состав транзакции действия, требуемые для выполнения этой прикладной программы.

Например, прикладная программа CICS* может обращаться к менеджерам ресурсов, таким, как база данных и очередь временных данных CICS®, и обрабатывать данные при помощи своей внутренней логики. Каждое требование доступа передается соответствующим менеджерам ресурсов при помощи вызовов функций, специфичных для конкретных менеджеров. В DB2 это могут быть вызовы функций, генерируемые для каждого оператора SQL прекомпилятором DB2®, или вызовы баз данных, кодируемые непосредственно программистом с использованием API.

Менеджер транзакций (TM) обычно включает в себя монитор транзакций (TP) для запуска пользовательских программ. Монитор транзакций предоставляет API, позволяющие программе начать и закончить транзакцию, а также включить программу в расписание и сбалансировать нагрузку для большого числа пользователей, которые запускают программу. Реально прикладная программа в среде распределенной обработки транзакций - это сочетание пользовательской программы и монитора транзакций.

Для упрощения работы среды оперативной обработки транзакций (OLTP) монитор транзакций размещает несколько процессов сервера при запуске, а затем планирует и использует их многократно во многих пользовательских транзакциях. Это экономит системные ресурсы, позволяя меньшему числу процессов серверов и соответствующих им процессов менеджера ресурсов поддерживать большее число одновременно работающих пользователей. Кроме того, многократное использование процессов позволяет избежать расходов, связанных с запуском процесса в менеджере транзакций и в менеджерах ресурсов для каждой пользовательской транзакции или программы. (Программа вызывает одну или несколько транзакций.) Это означает также, что для менеджера транзакций и менеджеров ресурсов процессы серверов являются ″пользовательскими процессами″. Это имеет значение для управления защитой и для написания программ.

Для монитора транзакций возможны следующие типы транзакций:

- Транзакции без XA

  Эти транзакции используют менеджеры ресурсов, которые не определены для менеджера транзакций и поэтому не координируются протоколом двухфазного принятия TM. Это необходимо, если программа должна обратиться к менеджеру ресурсов, который не поддерживает интерфейс XA. Монитор транзакций в этом случае просто обеспечивает расписание программ и балансировку нагрузки. Поскольку менеджер транзакций не ″открывает″ явно менеджеры ресурсов для обработки XA, менеджер ресурсов рассматривает эту программу, как и любую другую программу, выполняемую не в среде DTP.

- Глобальные транзакции

  Эти транзакции используют менеджеры ресурсов, которые определены для менеджера транзакций и координируются протоколом двухфазного принятия TM. Глобальная транзакция - это единица работы, которая может вовлекать один или несколько менеджеров ресурсов. Часть единицы работы между менеджером транзакций и менеджером ресурсов, поддерживающим глобальную транзакцию, называется *ветвью транзакции*. У глобальной транзакции может быть несколько ветвей транзакции, если один или несколько процессов программ, координируемых TM, обращаются к нескольким менеджерам ресурсов.

  В свободно связанных глобальных транзакциях каждый из процессов прикладной программы обращается к менеджерам ресурсов, как если бы он

был в отдельной глобальной транзакции, хотя эти программы и координируются менеджером транзакций. В менеджере ресурсов у каждого процесса прикладной программы будет своя собственная ветвь транзакции. Когда прикладная программа, менеджер транзакций или менеджер ресурсов требуют принятия или отката, ветви транзакции полностью завершаются. Ответственность за тупиковые ситуации между ветвями транзакции несет программа. (Заметим, что координация транзакций, выполняемая менеджером транзакций DB2 для программ, подготовленных с опцией SYNCPOINT(TWOPHASE), примерно эквивалентна таким свободно связанным глобальным транзакциям.

В тесно связанной глобальной транзакции несколько процессов прикладной программы запускаются по очереди под одной ветвью транзакции в менеджере ресурсов. Для менеджера ресурсов эти процессы прикладной программы являются одним объектом. Ответственность за тупиковые ситуации в этой ветви транзакции несет менеджер ресурсов.

Менеджер транзакций (TM)

Менеджер транзакций приписывает транзакциям идентификаторы, отслеживает ход транзакций и несет ответственность за их успешное или неудачное завершение. Идентификаторы ветвей транзакций (XID) назначаются менеджером транзакций для идентификации как глобальных транзакций, так и их отдельных ветвей в менеджере ресурсов. Они используются как метки соответствия между журналом менеджера транзакций и журналом менеджера ресурсов. XID используются в двухфазном принятии или откате для выполнения операции *ресинхронизации* при запуске системы, или чтобы в случае необходимости администратор мог выполнить *эвристическую* операцию (*ручное вмешательство*).

После запуска монитора транзакций он просит менеджер транзакций открыть все менеджеры ресурсов, определенные набором серверов прикладных программ. Менеджер транзакций передает менеджерам ресурсов вызовы **xa_open**, чтобы инициировать их для обработки DTP. В ходе процедуры запуска менеджер транзакций выполняет ресинхронизацию, чтобы восстановить *неоднозначные транзакции*. Неоднозначная транзакция - это глобальная транзакция, которая осталась в неопределенном состоянии. Такое случается, когда менеджер транзакций (или как минимум один менеджер ресурсов) стал недоступным после успешного завершения первой фазы (фазы подготовки) протокола двухфазного принятия. Менеджер ресурсов не будет знать, выполнять ему принятие или откат своей ветви транзакции, пока менеджер транзакций не согласует свой журнал с журналами менеджеров ресурсов, когда они снова станут доступными. Для выполнения ресинхронизации менеджер транзакций один или несколько раз передает вызов **xa_recover** каждому менеджеру ресурсов, чтобы идентифицировать все неоднозначные транзакции. Менеджер транзакций сравнивает ответы с соответствующей информацией в своем журнале и определяет, какой вызов надо послать менеджерам ресурсов: **xa_commit** или **xa_rollback**. Если какой-то менеджер ресурсов для своей ветви

неоднозначной транзакции уже выполнил принятие или откат в результате эвристической операции своего администратора, менеджер транзакций для ресинхронизации посылает этому менеджеру ресурсов вызов **xa_forget**.

Когда программа пользователя требует принятия или отката, она должна использовать API, предоставляемые монитором транзакций или менеджером транзакций, чтобы менеджер транзакций мог координировать принятия и откаты по всем вовлеченным менеджерам ресурсов. Например, если программа CICS выдает для принятия транзакции требование CICS SYNCPOINT, менеджер транзакций CICS XA (реализованный в сервере Encina® Server) в свою очередь выдаст вызовы XA **xa_end**, **xa_prepare**, **xa_commit** или **xa_rollback** в качестве требования менеджеру ресурсов выполнить принятие или откат данной транзакции. Если в транзакции участвует только один менеджер ресурсов, или менеджер ресурсов отвечает, что в его ветви транзакции выполнялось Н только чтение, менеджер транзакций вместо двухфазного принятия может Н использовать однофазное.

Менеджеры ресурсов (RM)

Менеджер ресурсов обеспечивает доступ к совместно используемым ресурсам, например, к базам данных.

DB2, как менеджер ресурсов базы данных, может участвовать в *глобальной транзакции*, которая координируется XA-совместимым менеджером транзакций. Как того требует интерфейс XA, менеджер баз данных обеспечивает внешнюю переменную *db2xa_switch* языка C типа xa_switch_t для возврата менеджеру транзакций структуры переключателя XA. Эта структура данных содержит адреса различных подпрограмм XA, вызываемых менеджером транзакций, и операционные характеристики менеджера ресурсов.

Есть два метода регистрации участия менеджера ресурсов в каждой глобальной транзакции: *статическая регистрация* и *динамическая регистрация*:

- При статической регистрации требуется, чтобы менеджер транзакций выдавал (для каждой транзакции) серии вызовов **xa_start**, **xa_end** и **xa_prepare** всем менеджерам ресурсов, определенным для программы сервера, независимо от того, используется ли данный менеджер ресурсов транзакцией. Если не все менеджеры ресурсов участвуют в каждой транзакции, это неэффективно, причем чем больше определено менеджеров транзакций, тем сильнее эта неэффективность.
- Динамическая регистрация (используемая DB2) гибче и эффективней. Менеджер ресурсов регистрируется с менеджером транзакций, используя вызов **ax_reg**, только когда он получает требование на свой ресурс. Заметим, что в этом методе производительность не ухудшается, даже если определен только один менеджер ресурсов или если каждый менеджер ресурсов используется всеми транзакциями, так как в менеджере транзакций вызовы **ax_reg** и **xa_start** обрабатываются одинаково.

Интерфейс XA обеспечивает двухстороннюю связь между менеджером транзакций и менеджером ресурсов. Это интерфейс системного уровня между двумя программными компонентами DTP, а не обычный программный интерфейс, код которого разрабатывают прикладные программисты. Однако программисты должны хорошо знать программные ограничения, которые налагаются компонентами программного обеспечения DTP.

Хотя интерфейс XA един, у каждого XA-совместимого менеджера транзакций могут быть свои способы интеграции менеджеров ресурсов. Информацию об интеграции вашего продукта DB2 в качестве менеджера ресурсов с H конкретным менеджером транзакций смотрите в документации соответствующего H менеджера транзакций.

Понятия, связанные с данным:

- “Сведения о защите для менеджеров транзакций XA” на стр. 197
- “Функции XA, поддерживаемые DB2 UDB” на стр. 200
- “X/Open XA Interface Programming Considerations” в книге *Application Development Guide: Programming Client Applications*

Задачи, связанные с данной темой:

- “Изменение в транзакции нескольких баз данных” на стр. 166

Настройка менеджера ресурсов

Каждая база данных определяется для менеджера транзакций как отдельный менеджер ресурсов; база данных должна идентифицироваться строкой xa_open.

При задании базы данных в качестве менеджера ресурсов строка xa_close необязательна. Если ее задать, менеджер баз данных будет игнорировать ее.

Сведения о соединении с базой данных

Транзакции, обращающиеся к многораздельным базам данных

В среде многораздельных баз данных пользовательские данные могут быть распределены по разделам базы данных. Программа, обращаясь к базе данных, соединяется с одним из ее разделов (узел координатора) и посылает ему требования. Разные программы могут соединяться с разными разделами базы данных, а одна и та же программа может выбирать в разных соединениях разные разделы базы данных.

Для транзакций в среде многораздельных баз данных весь доступ должен осуществляться через *один* раздел базы данных. Это значит, что с момента запуска транзакции и до момента ее принятия (включительно) должен использоваться один и тот же раздел.

Любая транзакция в многораздельной базе данных должна быть принята до того, как произойдет разъединение.

Понятия, связанные с данным:

- “Модель распределенной обработки транзакций X/Open” на стр. 182

Ссылки, связанные с данной темой:

- “Строковые форматы xa_open” на стр. 187

Строковые форматы xa_open

Строковые форматы xa_open для DB2 версии 7 более поздних версий

Ниже приведен формат строки xa_open:

```
id1_параметра = <значение параметра>,id2_параметра = <значение параметра>, ...
```

Порядок этих параметров несуществен. Допустимые значения для *id_параметра* описаны в следующей таблице.

Таблица 21. Допустимые значения для parm_id

Имя параметра	Значение	Обязательность	Регистрозависим?	Значение по умолчанию
DB	Алиас базы данных	Да	Нет	Отсутствуют
Алиас базы данных используется программой для обращения к базе данных.				
UID	ID пользователя	Нет	Да	Отсутствуют
ID пользователя, имеющего полномочия соединяться с этой базой данных. Обязательный, если задается пароль.				
PWD	Пароль	Нет	Да	Отсутствуют
Пароль, соответствующий данному ID пользователя. Обязателен, если задается ID пользователя.				
TPM	Имя монитора транзакций	Нет	Нет	Отсутствуют
Имя используемого монитора транзакций. Информация о поддерживаемых значениях приведена в следующей таблице. Этот параметр можно задать, чтобы несколько мониторов транзакций могли использовать один экземпляр DB2. Задаваемое значение заменяет значение, заданное параметром конфигурации менеджера баз данных *tp_mon_name*.				

Таблица 21. Допустимые значения для parm_id (продолжение)

Имя параметра	Значение	Обязательность	Регистрозависим?	Значение по умолчанию
AXLIB	Библиотека, содержащая функции монитора транзакций **ax_reg** и **ax_unreg**.	Нет	Да	Отсутствуют
Это значение DB2 использует для получения адресов требуемых функций **ax_reg** и **ax_unreg**. Его можно использовать для переопределения значений, основанных на значении параметра TPM, его могут использовать также мониторы транзакций, которых нет в списке TPM.				
CHAIN_END	Флаг цепочки xa_end. Допустимые значения - T, F или нет значения.	Нет	Нет	Ф
Цепочка XA_END - это оптимизация, которую DB2 может использовать для сокращения сетевых потоков. Если среда монитора транзакций гарантирует, что **xa_prepare** будет вызван в том же потоке или процессе немедленно после вызова **xa_end**, и если CHAIN_END включен, флаг xa_end будет связан с вызовом **xa_prepare**, исключая тем самым одну операцию передачи по сети. Значение T указывает, что CHAIN_END включен; значение F указывает, что CHAIN_END выключен; если значение не задано, предполагается, что CHAIN_END включен. Этот параметр может быть использован для переопределения значения, полученного из заданного значения TPM.				
SUSPEND_CURSOR	Задает, должны ли храниться указатели, если поток транзакции управления приостанавливается. Допустимые значения - T, F или нет значения.	Нет	Нет	Ф

Таблица 21. Допустимые значения для parm_id (продолжение)

Имя параметра	Значение	Обязательность	Регистрозависим?	Значение по умолчанию
Мониторы транзакций, которые приостановили выполнение ветви транзакции, могут повторно использовать приостановленный поток или процесс для других транзакций. Если опция SUSPEND_CURSOR выключена, то будут закрыты все указатели, кроме тех, для которых установлены атрибуты блокировки. При возобновлении приостановленной транзакции программа должна получить эти указатели снова. Если опция SUSPEND_CURSOR включена, то открытые указатели не закрываются и будут доступны приостановленной транзакции после восстановления. Значение T указывает, что SUSPEND_CURSOR включен; значение F указывает, что SUSPEND_CURSOR выключен; если значение не задано, предполагается, что SUSPEND_CURSOR включен. Этот параметр может быть использован для переопределения значения, полученного из заданного значения TPM.				
HOLD_CURSOR	Задает, удерживаются ли указатели после принятия транзакции. Допустимые значения - T, F или нет значения.	Нет	Нет	Ф
Мониторы транзакций обычно многократно используют потоки или процессы для нескольких программ. Чтобы гарантировать, что вновь загружаемая программа не унаследует указатели, открытые предыдущей программой, указатели закрываются после принятия. Если опция HOLD_CURSORS включена, то указатели с атрибутами блокировки не закрываются и сохраняются на протяжении нескольких транзакций. При использовании этой опции откат и повтор глобальных транзакций должен выполняться из той же управляющей нити. Если опция HOLD_CURSOR выключена, то в операции открытия указателей с атрибутами блокировки будет отказано. Значение T указывает, что HOLD_CURSOR включен; значение F указывает, что HOLD_CURSOR выключен; если значение не задано, предполагается, что HOLD_CURSOR включен. Этот параметр может быть использован для переопределения значения, полученного из заданного значения TPM.				

Значения TPM и tp_mon_name

Параметр TPM строки xa_open и параметр конфигурации *tp_mon_name* менеджера баз данных указывают для DB2, какой монитор транзакций используется. Параметр *tp_mon_name* применяется для всего экземпляра DB2. Параметр TPM применяется только для заданного менеджера ресурсов XA. Значение TPM переопределяет значение параметра *tp_mon_name*. Для параметров TPM и *tp_mon_name* допустимы следующие значения:

Таблица 22. Допустимые значения для параметров TPM и tp_mon_name

Значение TPM	Монитор транзакций	Внутренние параметры
CICS	IBM TxSeries CICS	AXLIB=libEncServer (для Windows) =/usr/lpp/encina/lib/libEncServer (для систем на основе UNIX) HOLD_CURSOR=T CHAIN_END=T SUSPEND_CURSOR=F
ENCINA	IBM TxSeries Encina Monitor	AXLIB=libEncServer (для Windows) =/usr/lpp/encina/lib/libEncServer (для систем на основе UNIX) HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
MQ	IBM MQSeries	AXLIB=mqmax (для Windows) =/usr/mqm/lib/libmqmax.a (для AIX) =/opt/mqm/lib/libmqmax.a (для Solaris) HOLD_CURSOR=F CHAIN_END=F SUSPEND_CURSOR=F
CB	IBM Component Broker	AXLIB=somtrx1i (для Windows) =libsomtrx1 (для систем на основе UNIX) HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
SF	IBM San Francisco	AXLIB=ibmsfDB2 HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
TUXEDO	BEA Tuxedo	AXLIB=libtux HOLD_CURSOR=F CHAIN_END=F SUSPEND_CURSOR=F
MTS	Microsoft Transaction Server	Конфигурировать DB2 для MTS необязательно. MTS определяется автоматически драйвером DB2 ODBC.

Таблица 22. Допустимые значения для параметров TPM и tp_mon_name (продолжение)

Значение TPM	Монитор транзакций	Внутренние параметры
JTA	Java Transaction API	Конфигурировать DB2 для серверов Enterprise Java Server (EJS), например для сервера IBM WebSphere, необязательно. Драйвер DB2 JDBC определяет эту среду автоматически. Поэтому значение TPM игнорируется.

Строковые форматы xa_open для более ранних версий

Ниже описывается формат строки xa_open, который использовался в прежних версиях DB2. Этот формат по-прежнему поддерживается из соображений совместимости. По возможности программы следует перенастроить на новый формат.

Каждая база данных определяется для менеджера транзакций как отдельный менеджер ресурсов и должна идентифицироваться строкой xa_open:

```
"алиас_базы_данных<,id_пользователя,пароль>"
```

Алиас_базы_данных нужен для задания алиаса базы данных. Алиас совпадает с именем базы данных, если после создания базы данных не внести в каталог алиас явно. Имя пользователя и пароль необязательны и, в зависимости от метода аутентификации, используются для задания информации аутентификации.

Примеры

1. Вы используете IBM TxSeries CICS в Windows NT. В документации TxSeries говорится, что для параметра *tp_mon_name* надо задать значение `libEncServer:C`. Это приемлемый формат, однако в DB2 UDB или DB2 Connect Версии 7 есть выбор:
   - Задать для *tp_mon_name* значение `CICS` (это рекомендуется для данного сценария):

     ```
     db2 update dbm cfg using tp_mon_name CICS
     ```

     Для каждой базы данных, определенной для CICS, в строке инициализации Region—> Resources—> Product—> XAD—> Resource manager (Регион—> Ресурсы—> Продукт—> XAD—> Менеджер ресурсов) задайте:

```
db=алиас_базы_данных,uid=id_пользователя,pwd=пароль
```

- Для каждой базы данных, определенной для CICS, в строке инициализации Region—> Resources—> Product—> XAD—> Resource manager (Регион—> Ресурсы—> Продукт—> XAD—> Менеджер ресурсов) задайте:

  ```
  db=алиас_базы_данных,uid=id_пользователя,pwd=пароль,tpm=cics
  ```

2. Вы используете IBM MQSeries CICS в Windows NT. В документации MQSeries говорится, что для параметра *tp_mon_name* надо задать значение `mqmax`. Это приемлемый формат, однако в DB2 UDB или DB2 Connect Версии 7 есть выбор:

   - Задать для *tp_mon_name* значение `MQ` (это рекомендуется для данного сценария):

     ```
     db2 update dbm cfg using tp_mon_name MQ
     ```

     Для каждой базы данных, определенной для CICS, в строке инициализации Region—> Resources—> Product—> XAD—> Resource manager (Регион—> Ресурсы—> Продукт—> XAD—> Менеджер ресурсов) задайте:

     ```
     uid=id_пользователя,db=алиас_базы_данных,pwd=пароль
     ```

   - Для каждой базы данных, определенной для CICS, в строке инициализации Region—> Resources—> Product—> XAD—> Resource manager (Регион—> Ресурсы—> Продукт—> XAD—> Менеджер ресурсов) задайте:

     ```
     uid=id_пользователя,db=алиас_базы_данных,pwd=пароль,tpm=mq
     ```

3. Вы используете в Windows NT и IBM TxSeries CICS, и IBM MQSeries. Используется единственный экземпляр DB2. В этом сценарии следует создать следующую конфигурацию:

   a. Для каждой базы данных, определенной для CICS, в строке инициализации Region—> Resources—> Product—> XAD—> Resource manager (Регион—> Ресурсы—> Продукт—> XAD—> Менеджер ресурсов) задайте:

      ```
      pwd=пароль,uid=id_пользователя,tpm=cics,db=алиас_базы_данных
      ```

   b. Для каждой базы данных, определенной в качестве ресурса в свойствах менеджера очередей, задайте XaOpenString как:

      ```
      db=алиас_базы_данных,uid=id_пользователя,pwd=пароль,tpm=mq
      ```

4. Вы разрабатываете свой собственный XA-совместимый менеджер транзакций (XA TM) в Windows NT и хотите указать DB2, что в библиотеке ″myaxlib″ есть требуемые функции **ax_reg** и **ax_unreg**. Библиотека ″maxlib″ находится в каталоге, заданном в операторе PATH. Есть выбор:

   - Задать *tp_mon_name* для библиотеки `myaxlib`:

     ```
     db2 update dbm cfg using tp_mon_name myaxlib
     ```

     и для каждой базы данных, определенной для XA TM, задать строку xa_open:

     ```
     db=алиас_базы_данных,uid=id_пользователя,pwd=пароль
     ```

- Для каждой базы данных, определенной для XA TM, задать строку xa_open:

  db=*алиас_базы_данных*,uid=*id_пользователя*,pwd=*пароль*,axlib=myaxlib

5. Вы разрабатываете свой собственный XA-совместимый менеджер транзакций (XA TM) в Windows NT и хотите указать DB2, что в библиотеке ″myaxlib″ есть требуемые функции **ax_reg** и **ax_unreg**. Библиотека ″maxlib″ находится в каталоге, заданном в операторе PATH. Кроме того, вы хотите задать цепочку XA END. Есть выбор:
   - Для каждой базы данных, определенной для XA TM, задать строку xa_open:

     db=*алиас_б_д*,uid=*id_польз*,pwd=*пароль*,axlib=myaxlib,chain_end=T
   - Для каждой базы данных, определенной для XA TM, задать строку xa_open:

     db=*алиас_б_д*,uid=*id_польз*,pwd=*пароль*,axlib=myaxlib,chain_end

Понятия, связанные с данным:

- “Модель распределенной обработки транзакций X/Open” на стр. 182

Ссылки, связанные с данной темой:

- “Параметр конфигурации Имя монитора процессора транзакций - tp_mon_name” в книге *Руководство администратора: Производительность*

Обновление серверов баз данных хоста и iSeries с помощью менеджера транзакций XA

Возможность обновления серверов баз данных хоста и iSeries зависит от архитектуры менеджера транзакций XA.

Процедура:

Для того чтобы поддерживались последовательности принятия от различных процессов, должен быть включен концентратор соединений DB2 Connect. Для того чтобы включить концентратор соединений DB2 Connect, присвойте параметру конфигурации менеджера баз данных *max_connections* значение, превосходящее значение *maxagents*. Обратите внимание, что для поддержки последовательностей принятия XA от различных процессов концентратору соединений DB2 Connect требуется клиент версии 7.1 или выше.

Кроме того, вам понадобиться DB2 Connect с настроенным менеджером точек синхронизации (SPM) DB2.

Ссылки, связанные с данной темой:

- “Параметр конфигурации Максимальное число агентов - maxagents” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Максимальное число одновременных соединений - max_connections” в книге *Руководство администратора: Производительность*

Разрешение неоднозначных транзакций вручную

XA-совместимый менеджер транзакций (монитор обработки транзакций) использует H процесс двухфазного принятия, сходный с процессом, который использует менеджер транзакций DB2. Принципиальное отличие между этими двумя средами заключается в том, что вместо менеджера транзакций DB2 и базы данных менеджера транзакций функцию регистрации и управления транзакциями обеспечивает монитор транзакций.

При использовании XA-совместимого менеджера транзакций могут случаться ошибки, сходные с ошибками, которые возникают при работе с менеджером транзакций DB2. Так же, как и менеджер транзакций DB2, XA-совместимый менеджер транзакций будет пытаться ресинхронизировать неоднозначные транзакции.

Если по каким-то причинам вы не можете ждать, пока менеджер транзакций автоматически разрешит неоднозначные транзакции, это можно сделать вручную. Этот процесс восстановления иногда называют ″принятием эвристического решения″.

Команда LIST INDOUBT TRANSACTIONS (с использованием опции WITH PROMPTING) или соответствующий ей набор API позволяет выполнить запрос, принятие или откат неоднозначных транзакций. Кроме того, она позволяет ″забыть″ транзакции, для которых было выполнено эвристическое принятие или откат, удалив записи журнала и освободив место.

Ограничения:

Пользуйтесь этими командами (или соответствующими API) с *чрезвычайной осторожностью* и только в крайнем случае. Лучшая стратегия - это подождать, пока ресинхронизацию проведет менеджер транзакций. Если выполнить вручную принятие или откат транзакции в одной из участвующих баз данных, могут возникнуть ошибки, связанные с нарушением целостности данных. Исправление таких ошибок требует от вас понимания логики программы, чтобы идентифицировать данные, которые были изменены или для которых был выполнен откат, а затем выполнить восстановление базы данных до указанного момента времени или вручную отменить или применить изменения повторно.

Если вы не можете ждать, пока менеджер транзакций инициирует процесс ресинхронизации, а нужно освободить ресурсы, связанные неоднозначной

транзакцией, необходимы эвристические операции. Такая ситуация может случиться, если менеджер транзакций не в состоянии выполнить ресинхронизацию в течение длительного времени, и неоднозначная транзакция связывает ресурсы, которые срочно требуются. Неоднозначной транзакцией оказываются связаны те ресурсы, которые были назначены ей до того, как стал недоступен менеджер транзакций или менеджер ресурсов. Для менеджера баз данных к таким ресурсам относятся блокировки в таблицах и индексах, пространство журнала и память, задействованная транзакцией. Кроме того, каждая транзакция сокращает (на единицу) максимальное число параллельных транзакций, которые может обработать база данных. Кроме того, резервную копию невозможно создать, пока все неоднозначные транзакции не будут обработаны.

Эвристическая функция forget необходима в следующих ситуациях:

- если эвристическое принятие или откат транзакции приведет к переполнению журнала, которое указывается в выводе команды INDOUBT TRANSACTIONS.
- если необходимо создать резервную копию

Эвристическая функция forget освобождает пространство журнала, занятое неоднозначной транзакцией. Это значит, что если менеджер транзакций в конечном счете выполнит операцию ресинхронизации для данной неоднозначной транзакции, потенциально он может принять неправильное решение для другого менеджера ресурсов, так как в этом менеджере ресурсов будет отсутствовать запись журнала для данной транзакции. Обычно ″пропущенная″ запись журнала подразумевает выполнение менеджером ресурсов отката транзакции.

Порядок действий:

Хотя универсального рецепта выполнения эвристических операций не существует, приведем несколько общих указаний:

1. Соединитесь с базой данных, для которой требуется выполнение всех транзакций.
2. Для вывода неоднозначных транзакций воспользуйтесь командой LIST INDOUBT TRANSACTIONS. *xid* - это ID глобальной транзакции; он идентичен *xid*, использовавшемуся менеджером транзакций и остальными менеджерами ресурсов, участвующими в транзакции.
3. Для каждой неоднозначной транзакции используйте известную информацию о программе и операционной среде, чтобы определить другие участвующие менеджеры ресурсов.
4. Определите доступность менеджера ресурсов:
   - Если менеджер ресурсов доступен, а неоднозначная транзакция была вызвана менеджером ресурсов, недоступным во второй фазе двухфазного принятия либо в предыдущем процессе ресинхронизации, следует

проверить журнал менеджера ресурсов, чтобы определить, какое действие предпринималось другими менеджерами ресурсов. Затем следует выполнить то же действие на этой базе данных, то есть, воспользовавшись командой LIST INDOUBT TRANSACTIONS, выполнить либо эвристическое принятие транзакции, либо ее эвристический откат.

- Если менеджер транзакций *недоступен*, необходимо будет посмотреть состояние транзакции в других участвующих менеджерах ресурсов, чтобы определить, какое действие вы должны выполнить:
  - Если по крайней мере один из остальных менеджеров ресурсов выполнил принятие данной транзакции, в остальных менеджерах ресурсов следует выполнить эвристическое принятие транзакции.
  - Если по крайней мере один из остальных менеджеров ресурсов выполнил откат данной транзакции, следует выполнить эвристический откат транзакции.
  - Если транзакция находится в ″подготовленном″ (неоднозначном) состоянии во всех участвующих менеджерах ресурсов, следует выполнить эвристический откат транзакции.
  - Если один или несколько из других менеджеров ресурсов недоступны, следует выполнить эвристический откат транзакции.

Для получения информации о неоднозначных транзакциях из DB2 UDB в системе UNIX или Windows соединитесь с базой данных и введите команду LIST INDOUBT TRANSACTIONS WITH PROMPTING или вызовите эквивалентный API.

Информацию о неоднозначности транзакций, относящуюся к серверам баз данных хоста или iSeries можно получить двумя способами:

- Информацию о неоднозначности можно получить непосредственно с сервера хоста или iSeries.

  Чтобы получить информацию о неоднозначности непосредственно от DB2 for z/OS and OS/390, введите команду DISPLAY THREAD TYPE(INDOUBT). Для принятия эвристического решения используйте команду RECOVER. Чтобы получить информацию о неоднозначности непосредственно от DB2 for iSeries, введите команду **wrkcmtdfn**.
- Информацию о неоднозначности можно получить от сервера DB2 Connect, использовавшегося для доступа к серверу баз данных хоста или iSeries.

  Чтобы получить информацию о неоднозначности от сервера DB2 Connect, сначала соединитесь с менеджером точек синхронизации DB2, соединившись с экземпляром DB2, который задается значением параметра конфигурации менеджера баз данных *spm_name*. Затем введите команду LIST DRDA INDOUBT TRANSACTIONS WITH PROMPTING для вывода неоднозначных транзакций и принятия эвристических решений.

Понятия, связанные с данным:

- “Двухфазное принятие (Two-phase commit)” на стр. 175

Ссылки, связанные с данной темой:

- “LIST INDOUBT TRANSACTIONS Command” в книге *Command Reference*
- “LIST DRDA INDOUBT TRANSACTIONS Command” в книге *Command Reference*

Сведения о защите для менеджеров транзакций XA

Монитор транзакций заранее размещает набор процессов сервера и запускает транзакции от различных пользователей с ID процессов сервера. Для базы данных каждый процесс сервера выглядит как большая программа, состоящая из множества единиц работы, которые все запускаются под одним ID, относящимся к данному процессу сервера.

Например, в среде AIX® с использованием CICS, где запускается регион TXSeries™ CICS®, этот ID соответствует имени пользователя AIX, под которым он определен. Все процессы CICS Application Server также выполняются с этим ″главным″ ID TXSeries CICS, который обычно задается как ″cics″. Пользователи CICS могут вызывать транзакции CICS под своими ID регистрации DCE, а находясь в CICS, могут также изменять свои ID с использованием транзакции регистрации CESN. В любом случае ID конечного пользователя недоступен для менеджера ресурсов. Таким образом, CICS Application Process может выполнять транзакции с несколькими именами пользователей, но для менеджера ресурсов они будут выглядеть как единая программа с несколькими единицами работы, но одним ID ″cics″. Вы можете, если хотите, задать ID пользователя и пароль в строке xa_open; тогда для соединения с базой данных вместо ID ″cics″ будет использоваться этот ID пользователя.

Это не повлияет существенно на статические операторы SQL, так как для доступа к базе данных используются не привилегии конечных пользователей, а привилегии связывающего. Однако это означает, что привилегия пакетов базы данных EXECUTE должна быть предоставлена ID сервера, а не ID конечного пользователя.

Для динамических операторов, аутентификация которых производится во время выполнения, привилегии доступа к объектам базы данных должны быть предоставлены ID сервера, а не реальным пользователям этих объектов. Вместо того, чтобы доверить управление доступом к базе данных заданным пользователям, вы должны положиться на систему монитора транзакций для определения, какие пользователи смогут запускать конкретные программы. ID сервера нужно предоставить все привилегии, которые требуются пользователям SQL.

Чтобы определить, у кого есть доступ к таблице или производной таблице базы данных, можно выполнить следующие действия:

1. Из производной таблицы каталога SYSCAT.PACKAGEDEP получите список всех пакетов, зависимых от данной таблицы или производной таблицы.
2. Определите имена программ сервера (например, программ CICS), соответствующие этим пакетам по соглашению об именах, использующемуся в вашей системе.
3. Определите программы клиентов (например, ID транзакций CICS), которые могут вызывать эти программы, а затем с помощью журнала монитора транзакций (например, журнала CICS) определите, кто и когда запускал эти транзакции или программы.

Понятия, связанные с данным:

• “Модель распределенной обработки транзакций X/Open” на стр. 182

Сведения о настройке для менеджеров транзакций XA

При настройке среды монитора транзакций следует рассмотреть следующие параметры конфигурации:

- *tp_mon_name*

  Этот параметр конфигурации идентифицирует имя используемого продукта монитора транзакций (например, ″CICS″ или ″ENCINA″).

- *tpname*

  Этот параметр конфигурации менеджера баз данных определяет имя удаленной программы транзакций, которую должен использовать клиент базы данных при выдаче требования на выделение серверу баз данных с использованием протокола связи APPC. Значение этого параметра устанавливается в файле конфигурации на сервере, и должно совпадать с именем процессора транзакций, сконфигурированным в программе транзакций SNA.

- *tm_database*

  Так как DB2® *не* координирует транзакции в среде XA, этот параметр конфигурации менеджера баз данных для XA-координируемых транзакций не используется.

- *maxappls*

  Этот параметр конфигурации базы данных задает максимальное число допустимых активных программ. Значение этого параметра должно быть равно или больше суммы числа соединяющихся программ плюс число части из этих программ, которые смогут работать параллельно в процессе двухфазного принятия или отката. Затем эту сумму следует увеличить на ожидаемое число неоднозначных транзакций, которые могут существовать в любое время.

Для среды монитора TP (например, TXSeries™ CICS) значение параметра *maxappls*, возможно, придется увеличить. Это должно помочь запустить все процессы монитора транзакций.

- *autorestart*

  Этот параметр конфигурации определяет, будет ли при необходимости автоматически выполняться процедура перезапуска базы данных (RESTART DATABASE). Значение по умолчанию - YES (перезапуск разрешен).

  Чтобы можно было использовать базу данных, содержащую неоднозначные транзакции, для нее должна быть выполнена операция перезапуска. Если параметр *autorestart* имеет значение NO, после прекращения последнего соединения с базой данных следующее соединение будет неудачным и потребуется явно запустить команду RESTART DATABASE. Это состояние будет существовать, пока неоднозначные транзакции не будут удалены либо операцией ресинхронизации менеджера транзакций, либо с помощью эвристической операции, инициированной администратором. Если при запуске команды RESTART DATABASE в базе данных есть неоднозначные транзакции, выдается сообщение. Администратор может затем использовать команду LIST INDOUBT TRANSACTIONS и другие команды процессора командной строки, чтобы получить информацию об этих неоднозначных транзакциях.

Понятия, связанные с данным:

- “Модель распределенной обработки транзакций X/Open” на стр. 182

Ссылки, связанные с данной темой:

- “Параметр конфигурации Разрешение автоматического перезапуска - autorestart” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Имя программы транзакций APPC - tpname” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Максимальное число активных прикладных программ - maxappls” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Имя базы данных менеджера транзакций - tm_database” в книге *Руководство администратора: Производительность*
- “Параметр конфигурации Имя монитора процессора транзакций - tp_mon_name” в книге *Руководство администратора: Производительность*
- “LIST INDOUBT TRANSACTIONS Command” в книге *Command Reference*
- “RESTART DATABASE Command” в книге *Command Reference*

Функции XA, поддерживаемые DB2 UDB

DB2® Universal Database поддерживает спецификацию XA91, определенную в документе *X/Open CAE Specification Distributed Transaction Processing: The XA Specification*, со следующими исключениями:

- Асинхронные службы

  Спецификация XA позволяет интерфейсу использовать асинхронные службы, поэтому результат требования может быть проверен позднее. Менеджер баз данных требует, чтобы требования вызывались в синхронном режиме.

- Статическая регистрация

  Интерфейс XA предусматривает два способа регистрации менеджера ресурсов: статическую регистрацию и динамическую регистрацию. DB2 Universal Database™ поддерживает только динамическую регистрацию, которая более совершенна и эффективна.

- Передача транзакций

  DB2 Universal Database не поддерживает передачу транзакций от одного управляющего потока другому.

Использование и положение переключателя XA

Как того требует интерфейс XA, менеджер баз данных обеспечивает внешнюю переменную *db2xa_switch* языка C типа xa_switch_t для возврата менеджеру транзакций структуры переключателя XA. Кроме адресов различных функций XA, возвращаются следующие поля:

Поле	Значение
name	Имя продукта менеджера баз данных. Например, DB2 for AIX.
flags	TMREGISTER | TMNOMIGRATE Явно устанавливает, что DB2 Universal Database использует динамическую регистрацию и что менеджер транзакций не должен использовать передачу связей. Неявно устанавливает, что асинхронная работа не поддерживается.
version	Должна равняться нулю.

Использование переключателя XA в DB2 Universal Database

Архитектура XA требует, чтобы менеджер ресурсов задал *переключатель*, который бы предоставил менеджеру транзакций XA доступ к программам **xa_** менеджера ресурсов. В переключателе менеджера ресурсов используется структура под названием xa_switch_t. Этот переключатель содержит имя менеджера ресурсов, непустые указатели на точки входа XA менеджера ресурсов, флаг и номер версии.

Системы на основе UNIX

Переключатель DB2 UDB можно получить одним из двух способов:

- Через дополнительный уровень косвенной ссылки. В программе на языке C это можно выполнить, определив макрокоманду:

  ```
  #define db2xa_switch (*db2xa_switch)
  ```

  перед использованием *db2xa_switch*.
- С помощью вызова **db2xacic**

  DB2 UDB предоставляет этот API, который возвращает адрес структуры *db2xa_switch*. Прототип этой функции:

  ```
  struct xa_switch_t * SQL_API_FN  db2xacic( )
  ```

При использовании любого из этих методов нужно связать свою программу с библиотекой `libdb2`.

Windows NT

Указатель на структуру *xa_switch*, *db2xa_switch*, экспортируется как данные DLL. Это значит, что программа Windows® NT, использующая данную структуру, должна обращаться к ней одним из трех способов:

- Через дополнительный уровень косвенной ссылки. В программе на языке C это можно выполнить, определив макрокоманду:

  ```
  #define db2xa_switch (*db2xa_switch)
  ```

  перед использованием *db2xa_switch*.
- Если используется компилятор Microsoft® Visual C++, *db2xa_switch* можно определить так:

  ```
  extern __declspec(dllimport) struct xa_switch_t db2xa_switch
  ```
- С помощью вызова **db2xacic**

  DB2 UDB предоставляет этот API, который возвращает адрес структуры *db2xa_switch*. Прототип этой функции:

  ```
  struct xa_switch_t * SQL_API_FN  db2xacic( )
  ```

При использовании любого из этих методов нужно связать свою программу с библиотекой `db2api.lib`.

Пример кода на языке C

Следующий код демонстрирует различные способы, с помощью которых к *db2xa_switch* можно обратиться из программы на языке C с любой платформы DB2 UDB. Не забудьте связать свою программу с соответствующей библиотекой.

```
#include <stdio.h>
#include <xa.h>

struct xa_switch_t * SQL_API_FN  db2xacic( );

#ifdef DECLSPEC_DEFN
extern __declspec(dllimport) struct xa_switch_t db2xa_switch;
```

```
   #else
   #define db2xa_switch (*db2xa_switch)
   extern struct xa_switch_t db2xa_switch;
   #endif
main( )
   {
      struct xa_switch_t *foo;
      printf ( "%s \n", db2xa_switch.name );
      foo = db2xacic();
      printf ( "%s \n", foo->name );
      return ;
   }
```

Понятия, связанные с данным:

- “Модель распределенной обработки транзакций X/Open” на стр. 182

Определение неполадок интерфейса XA

Обнаружив ошибку при обработке требования XA от менеджера транзакций, программа может оказаться не в состоянии получить от менеджера транзакций код ошибки. Если программа завершается аварийно или получает от монитора транзакций или менеджера транзакций непонятный код возврата, следует проверить журнал First Failure Service Log; информация об ошибке XA заносится в него, если действует уровень диагностики 3 или выше.

Следует также просмотреть сообщение консоли, файл ошибок менеджера транзакций или другую специфическую для продукта информацию о внешних программах обработки транзакций, которые вы используете.

Менеджер баз данных записывает все ошибки XA в журнал First Failure Service Log с SQLCODE -998 (ошибки транзакций или эвристические ошибки) и соответствующими кодами причины. Вот некоторые из наиболее типичных ошибок:

- Недопустимый синтаксис в строке xa_open.
- Ошибка соединения с базой данных, заданной в строке open, по одной из следующих причин:
  - База данных не внесена в каталог.
  - База данных не была запущена.
  - Имя пользователя программы сервера или пароль не прошли авторизацию для соединения с базой данных.
- Ошибка связи.

Понятия, связанные с данным:

- “Модель распределенной обработки транзакций X/Open” на стр. 182

Ссылки, связанные с данной темой:

- “Строковые форматы xa_open” на стр. 187

Конфигурация менеджера транзакций XA

Настройка IBM WebSphere Application Server

IBM WebSphere Application Server - это сервер приложений, написанный на языке Java. Он может использовать поддержку XA продукта DB2 с помощью API Транзакция Java (JTA), предусмотренного в драйвере JDBC DB2. Информация о применении этого API при работе с WebSphere Application Server приведена в документации по IBM WebSphere. Эту документацию можно найти в Internet на Web-сайте http://www-4.ibm.com/software/webservers/appserv/infocenter.html.

Конфигурирование IBM TXSeries CICS

Информацию о конфигурировании IBM TXSeries CICS для использования DB2 в качестве менеджера ресурсов смотрите в вашем руководстве *IBM TXSeries CICS Administration Guide*. Документацию по TXSeries можно просмотреть на сайте http://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/.

Серверы баз данных хоста и iSeries могут участвовать в транзакциях, координируемых CICS.

Конфигурирование IBM TXSeries Encina

Ниже приводятся различные API и параметры конфигурации, требуемые для интеграции монитора Encina Monitor с серверами DB2 Universal Database, DB2 for z/OS and OS/390, DB2 for iSeries или DB2 for VSE&VM при обращении через DB2 Connect. Документацию по TXSeries можно просмотреть на сайтеhttp://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/.

Серверы баз данных хоста и iSeries могут участвовать в транзакциях, координируемых Encina.

Конфигурирование DB2

Чтобы сконфигурировать DB2:

1. Каждая база данных должна быть зарегистрирована в каталоге баз данных DB2. Если это отдаленная база данных, для нее должна также быть запись в каталоге узлов. Конфигурирование можно выполнить при помощи графического интерфейса Ассистента конфигурирования или процессора командной строки DB2 (CLP). Например:

```
DB2 CATALOG DATABASE inventdb AS inventdb AT NODE host1 AUTH SERVER
DB2 CATALOG TCPIP NODE host1 REMOTE hostname1 SERVER svcname1
```

2. Клиент DB2 может оптимизировать внутреннюю обработку данных для Encina, если он знает, что работает с Encina. Это можно указать, задав для параметра конфигурации менеджера баз данных *tp_mon_name* значение ENCINA. По умолчанию специальная оптимизация не производится. Если значение для параметра *tp_mon_name* задано, программа должна обеспечить, чтобы поток, который выполняет единицу работы, немедленно выполнял принятие после ее завершения. Никакие другие единицы работы до этого запускать нельзя. Если вы используете *другую* среду, следует задать для *tp_mon_name* значение NONE (или, через CLP значение NULL). Этот параметр можно изменить через Центр управления или через процессор командной строки. Команда процессора командной строки выглядит так:

   ```
   db2 update dbm cfg using tp_mon_name ENCINA
   ```

Конфигурирование Encina для каждого менеджера ресурсов

Чтобы сконфигурировать Encina для каждого менеджера ресурсов, администратор должен определить строку открытия, строку закрытия и соглашение о потоке управления для каждой базы данных DB2 как менеджера ресурсов до того, как менеджер ресурсов может быть зарегистрирован для транзакций в прикладной программе. Это можно сделать через полноэкранный интерфейс Enconcole или через интерфейс командной строки Encina. Например:

```
monadmin create rm inventdb -open "db=inventdb,uid=пользователь1,pwd=пароль1"
```

Для каждой базы данных DB2 существует одна конфигурация менеджера ресурсов, и у каждой конфигурации менеджера ресурсов должно быть свое имя rm (″логическое имя менеджера ресурсов″). Для простоты его можно задать таким же, как у базы данных.

Строка xa_open содержит информацию, которая требуется для установления соединения с базой данных. Содержание этой строки определяется конкретным менеджером ресурсов. Строка xa_open для DB2 UDB содержит алиас открываемой базы данных и, необязательно, ID пользователя и пароль, связанные с соединением. Следует отметить, что указанное здесь имя базы данных должно быть зарегистрировано в обычном каталоге баз данных, что необходимо для любого обращения к базе данных.

Строка xa_close в DB2 не используется.

Соглашение о потоке управления определяет, может ли агент прикладной программы обрабатывать несколько транзакций за раз.

При обращении к DB2 for z/OS and OS/390, DB2 for iSeries или DB2 for VSE&VM следует использовать менеджер точек синхронизации DB2.

Обращение к базе данных DB2 из прикладной программы Encina

Чтобы обратиться к базе данных DB2 из прикладной программы Encina:

1. При помощи API Encina Scheduling Policy задайте, сколько агентов прикладных программ можно запустить из одного процесса прикладной программы монитора транзакций. Например:

   ```
   rc = mon_SetSchedulingPolicy (MON_EXCLUSIVE)
   ```

2. При помощи API регистрации менеджера ресурсов Encina задайте переключатель XA и логическое имя менеджера ресурсов, используемые Encina при обращении к менеджеру ресурсов в процессе прикладной программы. Например:

   ```
   rc = mon_RegisterRmi ( &db2xa_switch,   /* переключатель xa */
                          "inventdb",/* логическое имя менеджера ресурсов */
                          &rmiId );/* внутренний ID для менеджера ресурсов */
   ```

   Переключатель XA содержит адреса подпрограмм XA в менеджере ресурсов, которые может вызвать менеджер транзакций, а также задает функциональность, обеспечиваемую менеджером ресурсов. Переключатель XA для DB2 Universal Database называется `db2xa_switch` и находится в библиотеке клиента DB2 (`db2app.dll` - для операционных систем Windows и `libdb2` - для систем на основе UNIX).

   Логическое имя менеджера ресурсов - это имя, которое использует Encina, а не действительное имя базы данных, которое использует прикладная программа SQL, работающая под Encina. Действительное имя базы данных задается в строке xa_open в API регистрации менеджера ресурсов Encina. В данном примере логическое имя менеджера ресурсов совпадает с именем базы данных.

   Третий параметр возвращает внутренний идентификатор или хэндл, который используется TM для ссылок на это соединение.

Понятия, связанные с данным:

- “DB2 Connect и мониторы транзакций” в книге *DB2 Connect. Руководство пользователя*

Ссылки, связанные с данной темой:

- “Параметр конфигурации Имя монитора процессора транзакций - tp_mon_name” в книге *Руководство администратора: Производительность*
- “Строковые форматы xa_open” на стр. 187

Конфигурирование BEA Tuxedo

Порядок действий:

Чтобы сконфигурировать Tuxedo для использования DB2 в качестве менеджера ресурсов, выполните следующие действия:

1. Установите Tuxedo согласно документации этого продукта. Надо выполнить все основное конфигурирование для Tuxedo, включая файлы журналов и переменные среды.

   Кроме того, потребуется компилятор и клиент разработки программ DB2. При необходимости установите их.
2. Под ID сервера Tuxedo задайте значение переменной среды DB2INSTANCE - экземпляр, содержащий базу данных, которую вы хотите использовать с Tuxedo. В переменную PATH включите каталоги программ DB2. Подтвердите, что ID сервера Tuxedo может соединяться с базами данных DB2.
3. Измените значение параметра конфигурации менеджера баз данных *tp_mon_name* на TUXEDO.
4. Добавьте определение для DB2 в файл определений менеджера ресурсов Tuxedo. В приведенных ниже примерах UDB_XA - это заданное локально имя менеджера ресурсов Tuxedo для DB2, а *db2xa_switch* - имя, заданное в DB2 для структуры типа xa_switch_t:
   - Для AIX. В файл ${TUXDIR}/udataobj/RM добавьте определение:

     ```
     # DB2 UDB
     UDB_XA:db2xa_switch:-L${DB2DIR} /lib -ldb2
     ```

     где {TUXDIR} - каталог установки Tuxedo, а {DB2DIR} - каталог экземпляра DB2.
   - Для Windows NT. В файл %TUXDIR%\udataobj\rm добавьте определение:

     ```
     # DB2 UDB
     UDB_XA;db2xa_switch;%DB2DIR%\lib\db2api.lib
     ```

     где %TUXDIR% - каталог установки Tuxedo, а %DB2DIR% - каталог экземпляра DB2.
5. Смонтируйте программу сервера монитора транзакций Tuxedo для DB2:
   - Для AIX:

     ```
     ${TUXDIR}/bin/buildtms -r UDB_XA -o ${TUXDIR}/bin/TMS_UDB
     ```

     где {TUXDIR} - каталог установки Tuxedo.
   - Для Windows NT:

     ```
     %TUXDIR%\bin\buildtms -r UDB_XA -o %TUXDIR%\bin\TMS_UDB
     ```
6. Постройте серверы прикладных программ. В следующих примерах опция -r определяет имя менеджера ресурсов, опция -f (использованная один или несколько раз) определяет файлы, содержащие службы прикладных программ, опция -s задает имена служб прикладных программ для этого сервера, а опция -o задает имя выходного файла сервера:
   - Для AIX:

     ```
     ${TUXDIR}/bin/buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
        -o UDBserver
     ```

где {TUXDIR} - каталог установки Tuxedo.

- Для Windows NT:

```
%TUXDIR%\bin\buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
   -o UDBserver
```

где %TUXDIR% - каталог установки Tuxedo.

7. Задайте в файле конфигурации Tuxedo ссылку на сервер DB2. В раздел *GROUPS файла UDBCONFIG, добавьте запись, подобную следующей:

```
UDB_GRP   LMID=simp GRPNO=3
  TMSNAME=TMS_UDB TMSCOUNT=2
  OPENINFO="UDB_XA:db=пример,uid=пользователь_db2,
  pwd=пароль_пользователя_db2"
```

где параметр TMSNAME задает программу сервера монитора транзакций, которую вы смонтировали ранее, а параметр OPENINFO задает имя менеджера ресурсов. За ним следует имя базы данных, имя пользователя DB2 и пароль, которые используются для аутентификации.

Ссылки на построенные вами ранее серверы находятся в разделе *SERVERS файла конфигурации Tuxedo.

8. Если программа обращается к данным, находящимся в DB2 for z/OS and OS/390, DB2 for iSeries или DB2 for VM&VSE, требуется концентратор XA для DB2 Connect.
9. Запустите Tuxedo:

```
tmboot -y
```

Когда эта команда будет выполнена, сообщение Tuxedo должно показать, что серверы запущены. Кроме того, если вы введете команду DB2 LIST APPLICATIONS ALL, вы должны увидеть два соединения (в данном случае), задаваемые параметром TMSCOUNT в группе UDB файла конфигурации Tuxedo UDBCONFIG.

Понятия, связанные с данным:

- “DB2 Connect и мониторы транзакций” в книге *DB2 Connect. Руководство пользователя*

Ссылки, связанные с данной темой:

- “Параметр конфигурации Имя монитора процессора транзакций - tp_mon_name” в книге *Руководство администратора: Производительность*
- “LIST APPLICATIONS Command” в книге *Command Reference*

Часть 3. Приложения

Приложение A. Несовместимости выпусков

В этом разделе перечислены несовместимости между DB2 Universal Database и предыдущими выпусками DB2.

Несовместимостью мы называем некоторую часть DB2 Universal Database, которая работает иначе, чем в предыдущем выпуске. При работе существующей прикладной программы несовместимость может привести к неожиданному результату, к необходимости изменить прикладную программу или к снижению производительности. В данном контексте под ″прикладной программой″ понимаются:

- Код прикладной программы
- Утилиты независимых производителей
- Интерактивные запросы SQL
- Команда или вызов API.

Ниже описаны несовместимости между Версиями 7 и 8 DB2 Universal Database. Они сгруппированы по следующим категориям:

- Информация системного каталога
- Прикладное программирование
- SQL
- Защита и настройка баз данных
- Утилиты и инструменты
- Возможности соединений и сосуществование
- Сообщения
- Параметры конфигурации

В каждом разделе описаны несовместимости, их симптомы или влияние и возможные способы решения проблем. В начале описания приводится также индикатор операционной системы, к которой относится эта несовместимость:

WIN Платформы Microsoft Windows, поддерживаемые DB2

UNIX Платформы на основе UNIX, поддерживаемые DB2

OS/2 OS/2 (только для версии 7)

Планируемые несовместимости DB2 Universal Database

В этом разделе описаны будущие несовместимости, которые пользователи DB2 Universal Database должны иметь в виду при написании новых или модификации существующих прикладных программ. Это облегчит перенастройку в будущие версии DB2 UDB.

Информация системного каталога

PK_COLNAMES и FK_COLNAMES в будущей версии DB2 Universal Database

WIN	UNIX

Изменить: SYSCAT.REFERENCES не будет включать столбцы PK_COLNAMES и FK_COLNAMES.

Симптом: Столбец не существует; выдается сообщение об ошибке.

Объяснение: В коде инструмента или прикладной программы содержится использование устаревших столбцов PK_COLNAMES и FK_COLNAMES.

Разрешение: Измените инструмент или прикладную программу, чтобы использовать вместо SYSCAT.REFERENCES производную таблицу SYSCAT.KEYCOLUSE.

Столбец COLNAMES в будущей версии DB2 Universal Database

WIN	UNIX

Изменить: SYSCAT.INDEXES не будет содержать столбец COLNAMES.

Симптом: Столбец не существует; выдается сообщение об ошибке.

Объяснение: В коде инструмента или прикладной программы содержится использование устаревшего столбца COLNAMES.

Разрешение: Измените инструмент или прикладную программу, чтобы использовать вместо SYSCAT.INDEXES производную таблицу SYSCAT.INDEXCOLUSE.

Утилиты и инструменты

Поддержка повторного создания индексов типа 1 будет удалена

WIN	UNIX

Изменение: В версии 8 появился индекс нового типа - индекс типа 2. В индексах типа 1, которые были созданы до версии 8, в процессе удаления или изменения строки таблицы ключ физически удаляется с конечной страницы. В индексах типа 2 при удалении или изменении строки ключи помечаются как удаленные, но физически удаляются только после фиксации операции удаления или изменения строки. После удаления поддержки повторного создания индексов типа 1 вам больше не придется восстанавливать индексы вручную. Индексы типа 1, тем не менее, будут работать. Все действия, приводящие к повторному созданию индексов автоматически преобразуют индексы типа 1 в индексы типа 2. В будущей версии поддержка индексов типа 1 будет удалена.

Объяснение: Индексы типа 2 обладают несколькими преимуществами, по сравнению с индексами типа 1:

- Индекс типа 2 можно создать для столбца, длина которого превышает 255 байт
- Применение блокировки следующего ключа сведено к минимуму, в результате чего улучшена одновременность.

Разрешение: Разработайте план постепенного преобразования существующих индексов в индексы типа 2. Возможность Оперативной реорганизации индексов позволяет выполнить это с минимальными перерывами в работе. При необходимости, увеличьте размер табличного пространства индекса. Рассмотрите возможность создания новых индексов в больших табличных пространствах и перемещения существующих индексов в большие табличные пространства.

Отличия версии 8 от предыдущих выпусков

Информация системного каталога

Столбец IMPLEMENTED в таблицах каталога

WIN	UNIX

Изменение: В предыдущих версиях для столбца IMPLEMENTED таблиц SYSIBM.SYSFUNCTIONS и SYSCAT.SYSFUNCTIONS были допустимы значения Y, M, H и N. В версии 8 для него допустимы значения Y и N.

Исправление: Измените прикладные программы таким образом, чтобы в них применялись только значения Y и N.

Производные таблицы OBJCAT переименованы в SYSCAT

WIN	UNIX

Изменение: Следующие производные таблицы OBJCAT были переименованы в SYSCAT: TRANSFORMS, INDEXEXTENSIONS, INDEXEXTENSIONMETHODS, INDEXEXTENSIONDEP, INDEXEXTENSIONPARMS, PREDICATESPECS, INDEXEXPLOITRULES.

Исправление: Измените имена производных таблиц на SYSCAT в прикладных программах.

Производные таблицы SYSCAT теперь доступны только для чтения

WIN	UNIX

Изменение: В версии 8 производные таблицы SYSCAT доступны только для чтения.

Симптом: При попытке выполнить операцию UPDATE или INSERT для производной таблицы из схемы SYSCAT возникает ошибка.

Объяснение: Информацию системного каталога рекомендуется обновлять с помощью производных таблиц SYSSTAT. В предыдущих выпусках некоторые производные таблицы SYSCAT допускали обновление. В этом выпуске эта ошибка исправлена.

Исправление: Замените в прикладных программах указанные производные таблицы на производные таблицы SYSSTAT, допускающие обновление.

Создание прикладных программ

При использовании опции VERSION не возвращается код ошибки SQL0818N

WIN	UNIX

Изменение: При использовании новой опции VERSION в командах PRECOMPILE, BIND, REBIND и DROP PACKAGE вместо кода ошибки SQL0818N может быть возвращен код ошибки SQL0805N.

Симптом: Может измениться алгоритм работы прикладных программ, обрабатывающих код ошибки SQL0818N.

Исправление: Измените прикладные программы таким образом, чтобы они обрабатывали и код ошибки SQL0805N, и код ошибки SQL0818N.

Код ошибки SQL0306N не возвращается препроцессором, если не определена переменная хоста

WIN	UNIX

Изменение: Если переменная хоста не объявлена в разделе BEGIN DECLARE, однако используется в разделе EXEC SQL, то препроцессор не возвращает код ошибки SQL0306N. Если переменная объявлена в другой части прикладной программы, то во время выполнения программы будет возвращен код ошибки SQL0804N. Если переменная не объявлена в прикладной программе, то во время компиляции будет возвращено сообщение об ошибке.

Симптом: Может измениться алгоритм работы прикладных программ, которые обрабатывают код ошибки SQL0306N во время предварительной компиляции.

Исправление: Переменные хоста должны быть объявлены в разделе BEGIN DECLARE. Если переменная хоста объявлена в каком-либо другом разделе, добавьте в прикладную программу процедуру обработки кода возврата SQL0804.

Типы данных, которые нельзя применять с указателем с прокруткой

WIN	UNIX

Изменение: В версии 8 не поддерживаются указатели с прокруткой, применяющие типы LONG VARCHAR, LONG VARGRAPHIC, DATALINK и LOB, особые типы данных, созданные на основе этих типов, и структурированные типы. Любые из указанных типов, для которых в версии 7 было допустимо применение указателей с прокруткой, в этой версии поддерживаться не будут.

Симптом: Если столбцы указанных типов заданы в списке выбора указателя с прокруткой, то будет отправлен код возврата SQL0270N с кодом причины 53.

Исправление: Исключите из списка выбора указателя с прокруткой столбцы указанных типов.

Таблицы преобразования кодовых страниц с поддержкой символа евро

WIN	UNIX

Изменение: В версии 8 таблицы преобразования кодовых страниц с поддержкой символа евро отличаются от таблиц преобразования, поставляемых с предыдущими версиями DB2.

Исправление: Таблицы преобразования кодовых страниц, применявшиеся в предыдущих выпусках, доступны в каталоге `sqllib/conv/v7`.

Переключение между локатором LOB и его значением

WIN	UNIX

Изменение: Изменены правила переключения между локатором большого объекта (LOB) и его значением во время выполнения связывания для оператора указателя. Если приложение установило связывание с опцией SQLRULES DB2 (значение по умолчанию), то пользователь не сможет переключаться между локатором LOB и его значением.

Исправление: Если необходимо, чтобы при выполнении связывания для оператора указателя была возможность переключаться между локатором LOB и его значением, укажите во время предварительной компиляции прикладной программы опцию SQLRULES STD.

Непринятые единицы работы на платформах UNIX

	UNIX

Изменение: Если в прикладной программе на базе UNIX не применяется явная или неявная поддержка контекста, то в предыдущих версиях при нормальном завершении работы программы без вызова оператора CONNECT RESET, COMMIT или ROLLBACK принималась невыполненная единица работы. При завершении работы прикладных программ CLI, ODBC и Java (с неявной поддержкой контекста) и прикладных программ, явно создающих контекст, выполнялся откат незавершенной единицы работы. В случае аварийного завершения работы прикладной программы для незавершенной единицы работы также неявно выполнялась операция ROLLBACK. В версии 8 при завершении работы любой прикладной программы выполняется неявный откат незавершенной единицы работы. Алгоритм работы прикладных программ на базе Windows не изменится, так как они всегда выполняют неявный откат в случае нормального или аварийного завершения работы.

Исправление: Для того чтобы транзакции были приняты, перед завершением работы прикладной программы должна быть выполнена операция COMMIT или CONNECT RESET.

Изменены правила именования точек сохранения

WIN	UNIX

Изменение: В этой версии имена точек сохранения не должны начинаться с букв ″SYS″.

Симптом: При создании точки сохранения с именем, начинающимся с ″SYS″, возникает ошибка SQL0707N.

Объяснение: Имена точек сохранения, начинающиеся с ″SYS″, зарезервированы для внутреннего использования.

Исправление: Измените имена точек сохранения, начинающиеся с ″SYS″, на допустимые имена.

Ошибки при преобразовании кодовой страницы и подстановка байтов

WIN	UNIX

Изменение: В этой версии преобразование кодовой страницы выполняется во время связывания, когда это необходимо.

Симптом: Преобразование кодовой страницы и подстановка байтов может выполняться в непредвиденных случаях. Например, в указанных ниже операторах данные в переменной хоста :hv будут преобразованы из кодовой страницы приложения в кодовую страницу базы данных, если эти кодовые страницы не совпадают:

```
SELECT :hv FROM table
VALUES :hv
```

Объяснение: Указанные данные нельзя преобразовать в кодовую страницу базы данных без ошибок или подстановки байтов. В предыдущих версиях, в отличие от данной, преобразование кодовой страницы не выполнялось.

Исправление: Измените данные, кодовую страницу приложения или кодовую страницу базы данных таким образом, чтобы во время преобразования кодовой страницы не возникали ошибки и не выполнялась подстановка байтов, либо предусмотрите в прикладной программе возможность подстановки байтов или возникновения ошибок при преобразовании кодовой страницы.

Преобразование кодовой страницы для переменных хоста

WIN	UNIX

Изменение: В этой версии преобразование кодовой страницы выполняется во время связывания, когда это необходимо.

Симптом: Может привести к получению других результатов.

Объяснение: Поскольку преобразование кодовой страницы переменных хоста выполняется всегда, когда это необходимо, при оценке предикатов всегда используется кодовая страница базы данных, а не приложения. Например, оператор

```
SELECT * FROM table WHERE :hv1 > :hv2
```

будет выполнен с помощью кодовой страницы базы данных, а не приложения. При этом по-прежнему будет применяться последовательность сортировки базы данных.

Исправление: Убедитесь, что в предыдущих версиях обработка операторов давала правильный результат. Если это так, измените предикат с учетом того, что будет применяться последовательность сортировки и кодовая страница базы данных. Либо измените кодовую страницу прикладной программы или кодовую страницу базы данных.

Изменение длины данных в переменных хоста

WIN	UNIX

Изменение: В этой версии преобразование кодовой страницы выполняется во время связывания, когда это необходимо.

Симптом: Изменилась длина данных, хранящихся в переменных хоста.

Объяснение: Поскольку в результате преобразования кодовой страницы может возрасти или сократиться длина данных, результат выполнения операций, зависящих от длины данных в переменной хоста, может измениться. Кроме того, при выполнении таких операций могут возникать ошибки.

Исправление: Измените данные, кодовую страницу прикладной программы или кодовую страницу базы данных таким образом, чтобы во время преобразования кодовой страницы длина данных не менялась, либо предусмотрите в прикладной программе возможность изменения длины данных в результате преобразования кодовой страницы.

Длина переменных хоста после преобразования кодовой страницы

WIN	UNIX

Изменение: В этой версии в результате преобразования кодовой страницы длина переменных хоста и маркеров параметров не увеличивается.

Симптом: Ошибки, вызванные усечением данных.

Объяснение: В этой версии длина символьного типа данных, заданного для нетипизированного маркера параметра, не увеличивается с учетом возможного расширения данных в результате преобразования кодовой страницы. В операциях, определяющих длину результата по длине нетипизированного маркера параметра, длина результата станет меньше. Например, если C1 - это столбец типа CHAR(10), то в операции:

```
VALUES CONCAT (?, C1)
```

тип данных результата не будет равен CHAR(40), если в результате преобразования кодовой страницы прикладной программы в кодовую страницу базы данных длина данных может возрасти в 3 раза. Вместо этого тип данных результата будет равен CHAR(20).

Исправление: Назначьте нетипизированному маркеру параметра необходимый тип с помощью операции CAST, либо укажите в соответствующем операнде тип нетипизированного маркера параметра, учитывающий возможное расширение данных в результате преобразования кодовой страницы.

Изменился вывод оператора DESCRIBE

WIN	UNIX

Изменение: В этой версии в результате преобразования кодовой страницы длина переменных хоста и маркеров параметров не увеличивается.

Симптом: Изменился вывод оператора DESCRIBE.

Объяснение: Поскольку длина результата не возрастает в случае возможного расширения данных в результате преобразования кодовой страницы, вывод оператора DESCRIBE, содержащий длину результата, будет другим.

Исправление: При необходимости измените прикладную программу таким образом, чтобы в ней обрабатывались новые значения, возвращаемые оператором DESCRIBE.

Ошибка при использовании функции SUBSTR с переменными хоста

WIN	UNIX

Изменение: В этой версии в результате преобразования кодовой страницы длина переменных хоста и маркеров параметров не увеличивается.

Симптом: Ошибка SQL0138N при выполнении функции SUBSTR.

Объяснение: В предыдущих выпусках допускалось превышение длины переменной хоста с учетом возможного расширения данных в результате

преобразования кодовой страницы. Поэтому при выполнении функции SUBSTR (:hv,19,1) для переменной хоста длиной 10 ошибка не возникала. В этой версии такая функция недопустима.

Исправление: Увеличьте длину переменной хоста с учетом длины преобразованных данных, либо не указывайте в вызове функции SUBSTR позиции, превосходящие длину переменной хоста.

В Solaris больше не применяются библиотеки без поддержки потоков

	UNIX

Изменение: Библиотека libdb2_noth.so, не поддерживающая потоки, больше недоступна.

Симптом: Инструменты и прикладные программы, использующие библиотеку libdb2_noth.so, не будут работать.

Объяснение: Поскольку устаревшие библиотеки без поддержки потоков больше не нужны, библиотека libdb2_noth.so не поставляется с этой версией DB2 UDB for Solaris.

Исправление: Измените инструмент или прикладную программу таким образом, чтобы вместо указанной библиотеки применялась библиотека libdb2.so, поддерживающая потоки. Заново скомпонуйте прикладные программы с параметром -mt.

SQL

Уникальные имена функций и процедур не должны совпадать

WIN	UNIX

Изменение: Пространство имен SPECIFICNAME теперь должно содержать только уникальные имена. В предыдущих версиях DB2 функции и процедуре можно было назначить одинаковые уникальные имена. В версии 8 это недопустимо.

Симптом: Во время перехода к базе данных версии 8 утилита db2ckmig проверит, не совпадают ли уникальные имена функций и процедур. Если будут обнаружены одинаковые имена, то переход к новой версии не будет выполнен.

Исправление: Отбросьте процедуру и создайте ее заново с другим уникальным именем.

Привилегия EXECUTE для функций и процедур

WIN	UNIX

Изменение: В предыдущих выпусках любая подпрограмма, созданная пользователем, могла применяться другими пользователями. В этом выпуске пользователь, создавший подпрограмму, должен явно предоставить привилегию EXECUTE другим пользователям, для того чтобы они могли использовать эту подпрограмму.

В предыдущих версиях при вызове процедуры авторизация не выполнялась, однако у пользователя, вызвавшего процедуру, должна была быть привилегия EXECUTE для всех пакетов, вызываемых из этой процедуры. В версии 8 для вызова встроенной прикладной программы, при предварительной компиляции которой была задана опция CALL_RESOLUTION IMMEDIATE, а также для вызова процедуры CLI из каталога у пользователя должна быть привилегия EXECUTE для процедуры. Привилегия EXECUTE для всех пакетов должна быть только у пользователя, определяющего процедуру.

Симптом:

1. Прикладная программа может работать неправильно.
2. Если существует процедура, состоящая из нескольких пакетов, и у пользователя, определившего эту процедуру, нет прав доступа ко всем пакетом, то эта процедура не будет работать правильно.

Исправление:

1. Добавьте необходимые операторы GRANT EXECUTE. Если все подпрограммы относятся к одной схеме, то с помощью одного оператора можно предоставить привилегию для целого типа подпрограмм, например:

   ```
   GRANT EXECUTE ON FUNCTION schema1.* TO PUBLIC
   ```

2. Если один пакет разрешено использовать всем пользователям, а другой пакет - только некоторым пользователям, то при доступе ко второму пакету хранимая процедура, применяющая оба пакета, убедится, что нет ошибки полномочий. Наличие такой ошибки означает, что у пользователя нет необходимых привилегий, поэтому процедура пропустит часть своего кода.

   Эту ошибку можно исправить несколькими способами:

   a. Во время предварительной компиляции программы задайте опцию CALL_RESOLUTION DEFERRED, указывающую, что программа должна быть вызвана с помощью старого API sqleproc(), если препроцессору не удалось преобразовать процедуру в операторе CALL.

   b. Способ вызова процедур можно задать с помощью ключевого слова CLI UseOldStpCall в файле db2cli.ini. Для него допустимы два значения:

значение *0* указывает, что процедуры не будут вызываться старым способом, а значение *1* указывает, что процедуры будут вызываться старым способом.

c. Предоставьте привилегию EXECUTE всем пользователям, применяющим пакет.

Добавление ограничения по внешнему ключу для таблицы

WIN	UNIX

Изменение: В предыдущих версиях при создании ограничения по внешнему ключу, которое ссылается на таблицу, находящуюся в состоянии отложенной проверки, зависимая таблица также переводилась в состояние отложенной проверки. В версии 8 при создании ограничения по внешнему ключу, ссылающегося на таблицу, находящуюся в состоянии отложенной проверки, возможны два случая:

1. Если ограничение по внешнему ключу добавляется при создании зависимой таблицы, то таблица будет создана и ограничение будет добавлено, так как таблица создается пустой, поэтому ни одна строка таблицы не нарушает ограничение.
2. Если ограничение по внешнему ключу добавляется для существующей таблицы, то будет отправлен код ошибки SQL0668N.

Исправление: Перед добавлением ограничения по внешнему ключу включите функцию проверки целостности для таблицы, находящейся в состоянии отложенной проверки, на которую ссылается это ограничение. Для этого воспользуйтесь оператором SET INTEGRITY ... IMMEDIATE.

Изменение оператора SET INTEGRITY ... IMMEDIATE CHECKED

WIN	UNIX

Изменение: В предыдущих выпусках таблица, для которой был вызван оператор SET INTEGRITY ... UNCHECKED (т.е. у которой в столбце const_checked таблицы SYSCAT.TABLES есть байты 'U'), полностью обрабатывалась при вызове следующего оператора SET INTEGRITY ... IMMEDIATE CHECKED, то есть все записи таблицы проверялись на соответствие ограничениям. Для того чтобы избежать обработки всей таблицы, требовалось явно задать опцию INCREMENTAL.

В версии 8 при вызове оператора SET INTEGRITY ... IMMEDIATE по умолчанию непроверенные данные не обрабатываются (т.е. байты 'U' сохраняются), а выполняется только инкрементная обработка. (При этом возвращается предупреждение о том, что старые данные не были проверены.)

Объяснение: Это изменение внесено для того, чтобы по умолчанию не выполнялась проверка ограничений для всех записей, требующая большого количества ресурсов.

Исправление: Для того чтобы таблица была обработана полностью, необходимо явно задать опцию NOT INCREMENTAL.

Десятичный разделитель для функции CHAR

WIN	UNIX

Изменение: Если динамическое приложение выполняется на сервере с локалью, применяющей запятую в качестве десятичного разделителя, то при вызове в этом приложении неспецифицированных функций CHAR с аргументом типа REAL или DOUBLE функция CHAR(double) даст результат, содержащий точку в качестве разделителя. Такое различие возникает и при повторном создании производных таблиц и триггеров в версии 8, а также при явном повторном связывании статических пакетов.

Объяснение: Причина такого различия заключается в том, что вместо сигнатуры функции SYSFUN.CHAR(double) применяется новая сигнатура функции SYSIBM.CHAR(double).

Исправление: Для того чтобы результат применения функций был таким же, как и в предыдущих версиях DB2, в прикладной программе необходимо явно указать имя функции SYSFUN.CHAR. В противном случае функция CHAR будет преобразована в SYSIBM.CHAR.

Изменения оператора CALL

WIN	UNIX

Изменение: В версии 8 прикладные программы, при предварительной компиляции которых была задана опция CALL_RESOLUTION IMMEDIATE, и процедуры CLI из каталога значительно отличаются от аналогичных программ и процедур в предыдущих версиях:

- Поддержка переменных хоста заменена на поддержку динамического CALL.
- Не поддерживается компиляция прикладных программ, вызывающих хранимые процедуры, не зарегистрированные в каталоге. В следующей версии DB2 хранимые процедуры, не зарегистрированные в каталоге, не будут поддерживаться.
- Поддержка хранимых процедур с переменным списком аргументов больше не применяется.
- Изменены правила загрузки библиотеки хранимых процедур.

Исправление: Старая версия оператора CALL может применяться и в версии 8. Для этого в команде PRECOMPILE PROGRAM нужно указать опцию CALL_RESOLUTION DEFERRED.

Все прикладные программы, созданные в предыдущих версиях, будут правильно работать в версии 8. Однако если вы заново предварительно скомпилируете прикладную программу без опции CALL_RESOLUTION DEFERRED, то может потребоваться изменить код этой программы.

Оператор CALL_RESOLUTION DEFERRED не будет поддерживаться в следующей версии.

Вывод UDF, возвращающих строки фиксированной длины

WIN	UNIX

Изменение: UDF (скалярная или табличная функция) может возвращать строку фиксированной длины (CHAR(n) или GRAPHIC(n)). В предыдущих версиях длина возвращаемого значения, содержащего вложенный пустой символ, составляла n байт (или 2n байт для типов данных GRAPHIC) с учетом пустого символа и всех байт, расположенных справа от этого символа. В версии 8 при наличии пустого символа в строке вместо этого символа и всех остальных байт до конца значения возвращаются пробелы.

Исправление: Для того чтобы возвращалось такое же значение, как и в предыдущих версиях, укажите в определении возвращаемого значения вместо CHAR(n) значение CHAR(n) FOR BIT DATA. Для данных типа GRAPHIC избежать замены пустого символа на пробелы нельзя.

Изменение процедуры подключения к базе данных

WIN	UNIX

Изменение: В версии 7 при попытке подключиться к несуществующей базе данных после установления соединения с другой базой данных при помощи встроенного SQL выдавался код ошибки SQL1013N. Однако соединение с первой базой данных не разрывалось. В версии 8 при попытке подключиться к несуществующей базе данных разрывается соединение с первой базой данных. В результате прикладная программа будет отключена.

Исправление: Измените встроенный SQL таким образом, чтобы после неудачной попытки подключиться к другой базе данных восстанавливалось соединение с исходной базой данных.

Отзыв привилегии CONTROL для пакетов

WIN	UNIX

Изменение: Пользователь с привилегией CONTROL может предоставить привилегии для пакета. В DB2 версии 8 предусмотрена опция WITH GRANT OPTION, позволяющая указать, какими полномочиями должен обладать пользователь, чтобы он мог предоставлять привилегии для пакетов другим пользователям. Этот способ применяется вместо привилегии CONTROL для проверки того, что пользователь может предоставлять привилегии другим пользователям. После отзыва привилегии CONTROL пользователи по-прежнему смогут предоставлять привилегии другим пользователям.

Симптом: После отзыва привилегии CONTROL пользователь по-прежнему может предоставлять привилегии для пакета.

Исправление: Для того чтобы отозвать у пользователя полномочия на предоставление привилегий для пакетов другим пользователям, отзовите все привилегии для пакета и предоставьте только те привилегии, которые необходимы.

Ошибка при изменении типа символьной строки FOR BIT DATA на CLOB

WIN	UNIX

Изменение: В этой версии при попытке изменить тип символьной строки FOR BIT DATA на тип данных CLOB (с помощью спецификации CAST или функции CLOB) будет возвращен код ошибки (SQLSTATE 42846).

Симптом: При преобразовании к типу CLOB теперь возникает ошибка, хотя раньше ее не было.

Объяснение: Опция FOR BIT DATA не поддерживается для типа данных CLOB. Результат применения спецификации CAST или функции CLOB к строке FOR BIT DATA не определен. В такой ситуации возникает ошибка.

Исправление: Укажите в качестве аргумента спецификации CAST или функции CLOB значение, отличное от строки FOR BIT DATA. Для этого преобразуйте строку FOR BIT DATA в строку FOR SBCS DATA или строку FOR MIXED DATA с помощью спецификации CAST . Например, если C1FBD - это столбец VARCHAR(20), объявленный с опцией FOR BIT DATA в базе данных без поддержки DBCS, то указанное ниже выражение будет допустимым аргументом функции CLOB:

```
CAST (C1FBD AS VARCHAR(20) FOR SBCS DATA)
```

Защита и настройка базы данных

Полномочия для применения операторов CREATE FUNCTION, CREATE METHOD и CREATE PROCEDURE

WIN	UNIX

Изменение: В версии 8 появился новый тип полномочий - CREATE_EXTERNAL_ROUTINE.

Симптом: При вызове оператора CREATE FUNCTION, CREATE METHOD или CREATE PROCEDURE с опцией EXTERNAL может возникнуть ошибка.

Исправление: Предоставьте полномочия CREATE_EXTERNAL_ROUTINE тем пользователям, которые используют операторы CREATE FUNCTION, CREATE METHOD и CREATE PROCEDURE с опцией EXTERNAL.

Утилиты и инструменты

Команды CREATE DATABASE и DROP DATABASE не поддерживаются для клиентов и серверов старых версий

WIN	UNIX

Изменение: В версии 8 запрещено принимать команды CREATE DATABASE и DROP DATABASE от клиентов старых версий и отправлять такие команды на серверы старых версий.

Симптом: При вызове одной из этих команд возвращается код ошибки SQL0901N.

Объяснение: Команды CREATE DATABASE и DROP DATABASE разрешено использовать в том случае, если и на клиенте, и на сервере установлена версия 8. Эти команды нельзя отправить с клиента версии 6 или 7 на сервер версии 8. Кроме того, эти команды нельзя отправить с клиента версии 8 на сервер версии 7.

Исправление: Создайте или отбросьте базу данных версии 8 на клиенте версии 8. Создайте или отбросьте базу данных версии 7 на клиенте версии 6 или 7.

Изменился режим таблиц после загрузки

WIN	UNIX

Изменение: Если таблица загружена с опцией INSERT, и для этой таблицы созданы непосредственные материализованные таблицы запросов (или сводные

таблицы), то в предыдущих версиях после выполнения оператора SET INTEGRITY IMMEDIATE CHECKED для этой таблицы она переводилась в нормальное состояние (Полный доступ). В версии 8 после выполнения оператора SET INTEGRITY IMMEDIATE CHECKED такая таблица будет находиться в режиме без перемещения данных.

Объяснение: Правила доступа к таблице, находящейся в нормальном режиме (Полный доступ), мало отличаются от правил доступа к таблице, находящейся в режиме без перемещения данных. Отличие может быть ощутимо только для некоторых операторов и утилит, выполняющих перемещение данных внутри таблицы.

Исправление: Загруженную базовую таблицы, у которой есть зависимые непосредственные сводные таблицы, можно перевести в состояние Полный доступ, минуя состояние без перемещения данных. Для этого нужно вызвать оператор SET INTEGRITY ... IMMEDIATE CHECKED FULL ACCESS для этой таблицы. Однако применять эту опцию не рекомендуется, так как ее использование влечет за собой полное обновление зависимых непосредственных материализованных таблиц запроса (или сводных таблиц).

Утилита загрузки в режиме вставки или замены

WIN	UNIX

Изменение: В предыдущих версиях при использовании утилиты загрузки в режиме вставки или замены по умолчанию применялась опция CASCADE IMMEDIATE, если проверка целостности была выключена. После перевода таблицы в состояние с отложенной проверкой все зависящие от нее таблицы внешних ключей и материализованные таблицы запросов (или сводные таблицы) также немедленно переводились в состояние с отложенной проверкой.

В версии 8 при использовании утилиты загрузки в режиме вставки или замены по умолчанию применяется опция CASCADE DEFERRED, если проверка целостности выключена.

Исправление: Для того чтобы зависимые таблицы внешних ключей и материализованные таблицы запросов переводились в состояние с отложенной проверкой вместе с родительскими таблицами, укажите в команде LOAD опцию CHECK PENDING CASCADE IMMEDIATE.

Средства связи и поддержка предыдущих версий

Поддержка серверов предыдущих версий

WIN	UNIX

Изменение: В процессе перехода от версии 7 к версии 8 может возникнуть ситуация, при которой все клиентские компьютеры уже обновлены до версии 8, но еще есть серверы версии 7. В этом случае действуют некоторые ограничения. Эти ограничения не связаны с продуктом DB2 Connect, а также серверами баз данных для zSeries, OS/390 и iSeries.

Исправление: Для того чтобы клиенты версии 8 могли работать с серверами версии 7, необходимо настроить и включить функцию сервера приложений DRDA на сервере версии 7. Более подробная информация по этому вопросу приведена в руководстве *Дополнение по установке и настройке* для версии 7.

Для того чтобы избежать этих ограничений, рекомендуется обновить все серверы до версии 8 перед обновлением клиентов. Если это сделать нельзя, то необходимо учесть, что при работе клиентов версии 8 с серверами версии 7 не поддерживаются следующие функции:

- Некоторые типы данных:
  - Большие объекты (LOB).
  - Пользовательские особые типы (UDT).
  - Типы данных DATALINK.

    Тип данных DATALINK применяется для управления внешними данными, расположенными в нереляционной памяти. Значение этого типа содержит ссылку на файл, физически расположенный в файловой системе, внешней по отношению к DB2 Universal Database.
- Некоторые функции защиты:
  - Тип аутентификации SERVER_ENCRYPT.

    SERVER_ENCRYPT задает способ шифрования пароля. Зашифрованный пароль вместе с ID пользователя применяется для аутентификации пользователя.
  - Изменение паролей.

    Клиент версии 8 не может изменить пароль на сервере версии 7.
- Некоторые типы соединений и протоколы связи:
  - Запросы к экземпляру, для выполнения которых вместо установления соединения требуется выполнить операцию ATTACH.

    Клиенты версии 8 не могут выполнять операцию ATTACH при работе с сервером версии 7.
  - Поддерживается только протокол связи TCP/IP.

    Прочие протоколы связи, такие как SNA, NetBIOS и IPX/SPX, не поддерживаются.
- Некоторые функции и задачи прикладных программ:
  - Оператор DESCRIBE INPUT не поддерживается. Исключение составляют только прикладные программы ODBC/JDBC.

Для поддержки клиентов версии 8, применяющих прикладные программы ODBC/JDBC и работающих с серверами версии 7, на этих серверах версии 7 необходимо применить исправление оператора DESCRIBE INPUT. Это исправление связано с APAR IY30655. Оно будет выпущено до того, как версия 8 станет общедоступной. Информацию о том, как получить исправление, связанное с APAR IY30655, можно найти в разделе "Обращение в IBM" любого документа по продукту DB2 Universal Database.

Оператор DESCRIBE INPUT предназначен для повышения производительности и упрощения работы пользователя. Он позволяет инициатору прикладной программы получить описание маркеров входных параметров в подготовленном операторе. Для оператора CALL в их число входят маркеры параметров IN и INOUT хранимой процедуры.

– Двухфазное принятие.

  Сервер версии 7 нельзя применять как базу данных менеджера транзакций при выполнении скоординированных транзакций, в которых участвуют клиенты версии 8. Кроме того, сервер 7 не должен принимать участие в скоординированной транзакции, если роль базы данных менеджера транзакций играет сервер версии 8.

– Менеджеры транзакций XA.

  Прикладная программа, применяющая клиент версии 8, не должна использовать сервер версии 7 как ресурс XA. Это относится к программа WebSphere, Microsoft COM+/MTS, BEA WebLogic и другим программам, которые входят в среду управления транзакциями.

– Монитор.

– Утилиты.

  Клиент версии 8 не может запускать на сервере версии 7 никакие утилиты.

– Операторы SQL размером больше 32 Кб.

Для инструментов версии 8, работающих с серверами версии 7, действуют аналогичные ограничения.

Перечисленные ниже инструменты версии 8 могут применяться только при работе с серверами версии 8:

- Центр управления
- Центр заданий
- Журнал
- Центр управления сателлитами
- Центр каталогов данных (включая версию этого инструмента с Web-интерфейсом)
- Центр работоспособности (включая версию этого инструмента с Web-интерфейсом)
- Центр лицензий

- Модуль Spatial Extender
- Параметры инструментов

Ниже перечислены инструменты версии 8, поддерживающие серверы версии 7 (с некоторыми ограничениями) и серверы версии 8:

- Ассистент конфигурирования (этот инструмент состоит из различных компонентов; при работе с сервером версии 7 могут применяться только функции импорта и экспорта файлов конфигурации)
- Центр хранилищ данных
- Центр репликации
- Центр команд (включая версию этого инструмента с Web-интерфейсом)
- SQL Assist
- Центр разработки
- Визуальное представление

В общем случае все инструменты версии 8, которые запускаются только из дерева объектов Центра управления, а также все панели с подробными сведениями, связанные с этими инструментами, недоступны при работе с серверами версии 7 и ниже. При работе с такими серверами рекомендуется использовать инструменты версии 7.

Поддержка курсора с прокруткой

WIN	UNIX

Изменение: Курсор с прокруткой не поддерживается при работе клиентов DB2 UDB for Unix или Windows версии 8 с серверами DB2 UDB for Unix или Windows версии 7. Курсор с прокруткой поддерживается только при работе клиентов DB2 UDB for Unix или Windows версии 8 с сервером DB2 UDB for Unix или Windows версии 8 или сервером DB2 UDB for z/OS или OS/390 версии 7. Клиенты DB2 UDB for Unix или Windows версии 7 будут по-прежнему поддерживать курсор с прокруткой при работе с серверами DB2 UDB for Unix или Windows версии 8.

Исправление: Обновите серверы до версии 8.

Доступ к серверу версии 7 через сервер DB2 Connect версии 8

WIN	UNIX

Изменение: В версии 8 клиенты DB2 UDB for Unix или Windows не могут подключаться к серверу DB2 UDB версии 7 через сервер DB2 Connect Enterprise Edition версии 8 или DB2 UDB Enterprise Server Edition версии 8.

Исправление: Обновите серверы до версии 8.

Сообщения

Сообщения DB2 Connect, возвращаемые вместо сообщений DB2

WIN	UNIX

Изменение: В версии 8 при возникновении ряда событий вместо сообщений DB2, возвращавшихся в предыдущих выпусках, возвращается сообщение DB2 Connect.

Примеры:

- Вместо SQLCODE -1224 возвращается SQLCODE -30081
- Вместо SQLCODE -1403 возвращается SQLCODE -30082
- Вместо SQLCODE -4930 возвращается SQLCODE -30104

Симптом: Может измениться алгоритм работы прикладных программ, в которых предусмотрена процедура обработки сообщений DB2.

Параметры конфигурации

Устаревшие параметры конфигурации менеджера баз данных

WIN	UNIX

Изменение: Следующие параметры конфигурации менеджера баз данных устарели:

- *backbufsz*: В предыдущих версиях при резервном копировании мог применяться размер буфера по умолчанию, определяемый параметром *backbufsz*. В версии 8 при работе с утилитой резервного копирования необходимо явно указывать размер буферов резервного копирования.
- *dft_client_adpt*: Службы каталогов DCE больше не поддерживаются
- *dft_client_comm*: Службы каталогов DCE больше не поддерживаются
- *dir_obj_name*: Службы каталогов DCE больше не поддерживаются
- *dir_path_name*: Службы каталогов DCE больше не поддерживаются
- *dir_type*: Службы каталогов DCE больше не поддерживаются
- *dos_rqrioblk*
- *drda_heap_sz*
- *fcm_num_anchors*, *fcm_num_connect* и *fcm_num_rqb*: В этой версии DB2 автоматически и динамически настраивает метки сообщений, записи соединений и блоки запросов, поэтому эти параметры задавать не нужно
- *fileserver*: IPX/SPX больше не поддерживается

- *initdari_jvm*: В этой версии хранимые процедуры Java по умолчанию поддерживают несколько потоков и выполняются отдельно от остальных языковых функций, поэтому этот параметр больше не поддерживается
- *ipx_socket*: IPX/SPX больше не поддерживается
- *jdk11_path*: Заменен на параметр конфигурации менеджера баз данных *jdk_path*
- *keepdari*: Заменен на параметр конфигурации менеджера баз данных *keepfenced*
- *max_logicagents*: Заменен на параметр конфигурации менеджера баз данных *max_connections*
- *maxdari*: Заменен на параметр конфигурации менеджера баз данных *fenced_pool*
- *num_initdaris*: Заменен на параметр конфигурации менеджера баз данных *num_initfenced*
- *objectname*: IPX/SPX больше не поддерживается
- *restbufsz*: В предыдущих версиях при восстановлении данных мог применяться размер буфера по умолчанию, определяемый параметром *restbufsz*. В версии 8 при работе с утилитой восстановления требуется явно указывать размер буферов восстановления.
- *route_obj_name*: Службы каталогов DCE больше не поддерживаются
- *ss_logon*: Это параметр операционной системы OS/2, которая больше не поддерживается
- *udf_mem_sz*: UDF больше не записывают данные в общую память, поэтому этот параметр не поддерживается

Исправление: Удалите все ссылки на эти параметры из ваших программ.

Устаревшие параметры конфигурации базы данных

WIN	UNIX

Изменение: Следующие параметры конфигурации баз данных устарели:

- *buffpage*: В предыдущих версиях при создании и изменении пула буферов можно было использовать размер буфера по умолчанию, который задавался параметром *buffpage*. В версии 8 размер пула буферов требуется явно указывать в ключевом слове SIZE операторов ALTER BUFFERPOOL и CREATE BUFFERPOOL.
- *copyprotect*
- *indexsort*

Исправление: Удалите все ссылки на эти параметры из ваших программ.

Отличия версии 7 от предыдущих выпусков

Прикладное программирование

Универсальный клиент Query Patroller

WIN	UNIX	OS/2

Изменение: Новая версия CAE (Client Application Enabler) будет работать только с сервером Query Patroller Версии 7, так как использует новые хранимые процедуры. CAE - это программный интерфейс к DB2, с помощью которого все прикладные программы получают в конечном счете доступ к базе данных.

Симптом: Если CAE работает с сервером устаревшей версии, выдается сообщение SQL29001.

Функции преобразования объектов и структурные типы

WIN	UNIX	OS/2

Изменение: Есть небольшая потенциально возможная несовместимость между клиентом версий до 7 и сервером Версии 7, связанная с изменениями в SQLDA. Восьмой байт второй переменной SQLVAR теперь может принимать значение X'12' (в дополнение к значениям X'00' и X'01'). Это значение может повлиять на поведение прикладных программ, где такая возможность не предусмотрена.

Исправление: Поскольку в будущих версиях могут быть добавлены другие значения этого поля, мы советуем разработчикам при проверке задавать значения явно.

Версии файлов классов и jar-файлов, используемых виртуальной Java-машиной

WIN	UNIX	OS/2

Изменение: Раньше, когда запускалась хранимая Java-процедура или пользовательская функция, виртуальная Java-машина блокировала все файлы, указанные в CLASSPATH (включая файлы в `sqllib/function`). Виртуальная Java-машина использовала эти файлы вплоть до остановки менеджера баз данных. Теперь в зависимости от среды, в которой запускается хранимая процедура или пользовательская функция (то есть в зависимости от значения параметра конфигурации менеджера баз данных *keepdari* и от того, является ли хранимая процедура изолированной), обновляемые классы позволят заменять файлы классов и jar-файлы, не останавливая менеджер баз данных. Это отличается от поведения предыдущих версий.

Изменение функциональных возможностей команд Install, Replace и Remove для файлов jar

WIN	UNIX	OS/2

Изменение: Раньше установка файла jar вызывала очистку всех процессов DARI. Таким образом, при следующем вызове гарантированно выбирался новый класс хранимых процедур. Теперь команды jar не очищают процессы DARI. Чтобы обеспечить выбор классов из только что установленных или замененных файлов jar, выполните явно команду SQLEJ.REFRESH_CLASSES.

Другая несовместимость, связанная с отказом от очистки процессов DARI - это то, что для изолированных хранимых процедур при значении параметра конфигурации менеджера баз данных *keepdari* "YES" клиенты могут получить различные версии файлов jar. Рассмотрим следующий сценарий:

1. Пользователь A заменяет файл jar и не обновляет классы.
2. Далее он вызывает из jar хранимую процедуру. Если этот вызов использует тот же процесс DARI, пользователь A получит устаревшую версию файла jar.
3. Пользователь B вызывает ту же хранимую процедуру. Этот вызов использует новый процесс DARI, а это означает, что вновь созданный загрузчик классов выберет новую версию файла jar.

Другими словами, если после операций jar не обновлять классы, могут быть вызваны хранимые процедуры из разных версий файлов jar в зависимости от того, какие процессы DARI используются. Это отличается от предыдущего поведения, где (путем очистки процессов DARI) гарантировалось использование новых классов.

Несовместимость 32-битных прикладных программ

	UNIX	

Изменение: 32-битные выполняемые программы (прикладные программы DB2) не будут работать с новым 64-битным механизмом баз данных.

Симптом: Не удается связать прикладную программу. При попытке связать 32-битные объекты с 64-битной библиотекой прикладных программ DB2 выдается сообщение об ошибке компоновщика операционной системы.

Исправление: Прикладную программу надо перекомпилировать как 64-битную и пересвязать с новыми 64-битными библиотеками DB2.

Изменение поля длины в Scratchpad

WIN	UNIX	OS/2

Изменение: Любая пользовательская функция, изменяющая поле длины в передаваемой ей области scratchpad, теперь получит SQLCODE -450.

Симптом: Пользовательская функция, изменяющая поле длины области scratchpad, завершается неудачно. Вызывающий оператор получит SQLCODE -450 с подставленной схемой и именем конкретной функции.

Исправление: Перепишите тело пользовательской функции, чтобы она не меняла поле длины области scratchpad.

SQL

Прикладные программы, использующие обычные таблицы в схеме SESSION

WIN	UNIX	OS/2

Изменение: Временные таблицы могут размещаться только в схеме SESSION; теперь эта схема используется DB2, чтобы указать, что таблица может иметь отношение к временной таблице. Однако SESSION не является ключевым словом, зарезервированным для временных таблиц; эту схему можно использовать для обычных таблиц базы. Это значит, что в прикладной программе может оказаться, что одновременно существуют настоящая таблица SESSION.T1 и объявленная временная таблица SESSION.T1. Если при связывании пакета встречается статический оператор со ссылкой на таблицу со спецификатором (явным или неявным) ″SESSION″, ни раздел, ни зависимости этого оператора не будут записаны в каталоги. Вместо этого данный раздел надо будет инкрементно связать во время выполнения. При этом копия раздела будет помещена в динамический кэш SQL, причем эта копия будет доступной только одному экземпляру прикладной программы. Если во время выполнения объявленная временная таблица с соответствующим именем существует, будет использована она, даже если существует постоянная таблица базы с тем же именем.

Симптом: В Версии 6 (и более ранних) любой пакет со статическими операторами, ссылающимися на таблицы из схемы SESSION, всегда относился к постоянной таблице базы. При связывании пакета раздел, как и соответствующие записи зависимостей этого оператора, сохранялся в каталогах. В Версии 7 эти операторы не связываются во время связывания, а во время выполнения ссылки могут быть разрешены как ссылки на объявленную временную таблицу. Таким образом, могут возникнуть следующие ситуации:

- Перенастройка из Версии 5. Если такой пакет существовал в Версии 5, он будет снова связан в Версии 6, и статические операторы теперь будут связаны инкрементно. Это может повлиять на производительность, потому что инкрементно связанные разделы ведут себя подобно кэшированному динамическому SQL, за исключением того, что кэшированный динамический

раздел не может совместно использоваться другими прикладными программами (и даже разными экземплярами одной и той же выполняемой прикладной программы).

- Перенастройка из Версии 6 в Версию 7. Если такой пакет существовал в Версии 6, он необязательно будет пересвязан в Версии 7. Вместо этого операторы будут работать с ним как с обычным статическим SQL, используя раздел, сохраненный в каталоге во время исходного связывания. Однако если данный пакет повторно связывается (явно или неявно), операторы в пакете со ссылками на таблицы схемы SESSION не будут записываться в каталог и потребуют инкрементного связывания. Это может ухудшить производительность.

В целом можно сказать, что любые связанные в Версии 7 пакеты со статическими операторами, в которых есть ссылки на таблицы схемы SESSION, не будут теперь выполняться как статический SQL, потому что для них требуется инкрементное связывание. Если в процессе прикладной программы будет выполнен оператор DECLARE GLOBAL TEMPORARY TABLE для таблицы с таким же именем, как у существующей таблицы, производной таблицы или алиаса схемы SESSION, ссылки на эти объекты всегда будут считаться ссылками на объявленную временную таблицу.

Исправление: Если это возможно, измените имена схем постоянных таблиц с ″SESSION″ на другие. В противном случае надо иметь в виду последствия для производительности и возможные конфликты с объявленными временными таблицами.

Следующий запрос можно использовать для определения таблиц, производных таблиц и алиасов, на которые может повлиять использование прикладной программой временных таблиц:

```
select tabschema, tabname from SYSCAT.TABLES where tabschema = 'SESSION'
```

Следующий запрос можно использовать для определения связанных пакетов Версии 7, у которых есть статические разделы, хранящиеся в каталогах, и чье поведение может измениться, если пакет связывается повторно (это относится только к перенастройке с Версии 6 в Версию 7):

```
select pkgschema, pkgname, bschema, bname from syscat.packagedep
   where bschema = 'SESSION' and btype in ('T', 'V', 'I')
```

Утилиты и инструменты

db2set в AIX и Solaris

	UNIX	

Изменение: Команда ″db2set -ul (user level - пользовательский уровень)″ и связанные с ней функции не переносятся на AIX или Solaris.

Возможности соединений и сосуществование

Несовместимость 32-битного клиента

WIN	UNIX	OS/2

Изменение: 32-битные клиенты не могут подключаться к экземплярам или соединяться с базами данных на 64-битных серверах.

Симптом: Если на клиенте и на сервере работает код Версии 7, возвращается сообщение SQL1434N; в противном случае попытка подключения или соединения завершается с SQLCODE -30081.

Исправление: Используйте 64-битные клиенты.

Приложение B. Поддержка национальных языков (NLS)

В этом разделе содержится информация о поддержке в DB2 национальных языков (national language support - NLS), включая информацию о поддерживаемых регионах, языках и кодовых страницах (кодовых наборах), а также о конфигурировании и использовании функций NLS DB2 в базах данных и прикладных программах.

Версии на национальных языках

DB2 версии 8 поставляется на упрощенном китайском, традиционном китайском, датском, английском, финском, французском, немецком, итальянском, японском, корейском, норвежском, польском, португальском (Бразилия), русском, испанском и шведском языках.

Клиент DB2 поставляется, кроме того, на следующих языках: арабском, болгарском, хорватском, чешском, нидерландском, греческом, венгерском, португальском, румынском, словацком, словенском и турецком языках и на иврите.

Ссылки, связанные с данной темой:

- “Поддерживаемые коды регионов и кодовые страницы” на стр. 239

Поддерживаемые коды регионов и кодовые страницы

В Табл. 23 на стр. 240 показаны языки и кодовые наборы, поддерживаемые серверами баз данных, а также отображение этих значений на значения кода региона и кодовой страницы, используемые менеджером баз данных.

Далее приводится описание всех столбцов в этой таблице:

- **Кодовая страница** содержит кодовую страницу стандарта IBM, как она отображается из кодового набора операционной системы.
- **Группа** показывает, является ли кодовая страница однобайтной (″S″), двухбайтной (″D″) или нейтральной (″N″). К этой букве для создания буквенно-цифровой комбинации добавляется число ″-n″. Совпадение комбинаций означает, что DB2 разрешает соединение и поддерживает преобразование. Например, все группы ″S-1″ можно использовать вместе. Нейтральная группа позволяет устанавливать соединения и выполнять преобразования с любой другой кодовой страницей в списке.

- **Кодовый набор** показывает кодовый набор, связанный с поддерживаемым языком. Этот кодовый набор отображается на кодовую страницу DB2.
- **ID региона** содержит идентификатор региона.
- **Код региона** содержит код региона, используемый внутри менеджера баз данных для поддержки особенностей этой страны.
- **Локаль** содержит значения локалей (национальных версий), поддерживаемые менеджером баз данных.
- **ОС** содержит операционную систему, поддерживающую эти языки и кодовые наборы.
- **Регион** содержит имя региона, страны или нескольких стран.

Таблица 23. Поддерживаемые языки и кодовые наборы

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
1200		16-разрядный Unicode			-	Любая	Любой
1208		Кодировка UTF-8 Unicode				Любая	Любой
819	S-1	ISO8859-1	AL	355	sq_AL	AIX	Албания
850	S-1	IBM-850	AL	355	-	AIX	Албания
923	S-1	ISO8859-15	AL	355	sq_AL.8859-15	AIX	Албания
1208	N-1	UTF-8	AL	355	SQ_AL	AIX	Албания
37	S-1	IBM-37	AL	355	-	HOST	Албания
1140	S-1	IBM-1140	AL	355	-	HOST	Албания
819	S-1	iso88591	AL	355	-	HP	Албания
923	S-1	iso885915	AL	355	-	HP	Албания
1051	S-1	roman8	AL	355	-	HP	Албания
437	S-1	IBM-437	AL	355	-	OS2	Албания
850	S-1	IBM-850	AL	355	-	OS2	Албания
819	S-1	ISO8859-1	AL	355	-	Вс	Албания
923	S-1	ISO8859-15	AL	355	-	Вс	Албания
1252	S-1	1252	AL	355	-	WIN	Албания
1046	S-6	IBM-1046	AA	785	Ar_AA	AIX	Арабские страны/регионы
1089	S-6	ISO8859-6	AA	785	ar_AA	AIX	Арабские страны/регионы
1208	N-1	UTF-8	AA	785	AR_AA	AIX	Арабские страны/регионы
420	S-6	IBM-420	AA	785	-	HOST	Арабские страны/регионы
425	S-6	IBM-425	AA	785	-	HOST	Арабские страны/регионы

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
1089	S-6	iso88596	AA	785	ar_SA.iso88596	HP	Арабские страны/регионы
864	S-6	IBM-864	AA	785	-	OS2	Арабские страны/регионы
1256	S-6	1256	AA	785	-	WIN	Арабские страны/регионы
819	S-1	ISO8859-1	AU	61	en_AU	AIX	Австралия
850	S-1	IBM-850	AU	61	-	AIX	Австралия
923	S-1	ISO8859-15	AU	61	en_AU.8859-15	AIX	Австралия
1208	N-1	UTF-8	AU	61	EN_AU	AIX	Австралия
37	S-1	IBM-37	AU	61	-	HOST	Австралия
1140	S-1	IBM-1140	AU	61	-	HOST	Австралия
819	S-1	iso88591	AU	61	-	HP	Австралия
923	S-1	iso885915	AU	61	-	HP	Австралия
1051	S-1	roman8	AU	61	-	HP	Австралия
437	S-1	IBM-437	AU	61	-	OS2	Австралия
850	S-1	IBM-850	AU	61	-	OS2	Австралия
819	S-1	ISO8859-1	AU	61	en_AU	SCO	Австралия
819	S-1	ISO8859-1	AU	61	en_AU	Bc	Австралия
923	S-1	ISO8859-15	AU	61	-	Bc	Австралия
1252	S-1	1252	AU	61	-	WIN	Австралия
819	S-1	ISO8859-1	AT	43	-	AIX	Австрия
850	S-1	IBM-850	AT	43	-	AIX	Австрия
923	S-1	ISO8859-15	AT	43	-	AIX	Австрия
1208	N-1	UTF-8	AT	43	-	AIX	Австрия
37	S-1	IBM-37	AT	43	-	HOST	Австрия
1140	S-1	IBM-1140	AT	43	-	HOST	Австрия
819	S-1	iso88591	AT	43	-	HP	Австрия
923	S-1	iso885915	AT	43	-	HP	Австрия
1051	S-1	roman8	AT	43	-	HP	Австрия
819	S-1	windows-1251	AT	43	de_AT	Linux	Австрия
923	S-1	ISO-8859-15	AT	43	de_AT@euro	Linux	Австрия
437	S-1	IBM-437	AT	43	-	OS2	Австрия
850	S-1	IBM-850	AT	43	-	OS2	Австрия
819	S-1	ISO8859-1	AT	43	de_AT	SCO	Австрия
819	S-1	ISO8859-1	AT	43	de_AT	Bc	Австрия
923	S-1	ISO8859-15	AT	43	de_AT.ISO8859-15	Bc	Австрия
1252	S-1	1252	AT	43	-	WIN	Австрия
915	S-5	ISO8859-5	BY	375	be_BY	AIX	Беларусь

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
1208	N-1	UTF-8	BY	375	BE_BY	AIX	Беларусь
1025	S-5	IBM-1025	BY	375	-	HOST	Беларусь
1154	S-5	IBM-1154	BY	375	-	HOST	Беларусь
915	S-5	ISO8859-5	BY	375	-	OS2	Беларусь
1131	S-5	IBM-1131	BY	375	-	OS2	Беларусь
1251	S-5	1251	BY	375	-	WIN	Беларусь
819	S-1	ISO8859-1	BE	32	fr_BE	AIX	Бельгия
819	S-1	ISO8859-1	BE	32	nl_BE	AIX	Бельгия
850	S-1	IBM-850	BE	32	Fr_BE	AIX	Бельгия
850	S-1	IBM-850	BE	32	Nl_BE	AIX	Бельгия
923	S-1	ISO8859-15	BE	32	fr_BE.8859-15	AIX	Бельгия
923	S-1	ISO8859-15	BE	32	nl_BE.8859-15	AIX	Бельгия
1208	N-1	UTF-8	BE	32	FR_BE	AIX	Бельгия
1208	N-1	UTF-8	BE	32	NL_BE	AIX	Бельгия
274	S-1	IBM-274	BE	32	-	HOST	Бельгия
500	S-1	IBM-500	BE	32	-	HOST	Бельгия
1148	S-1	IBM-1148	BE	32	-	HOST	Бельгия
819	S-1	iso88591	BE	32	-	HP	Бельгия
923	S-1	iso885915	BE	32	-	HP	Бельгия
819	S-1	windows-1251	BE	32	fr_BE	Linux	Бельгия
819	S-1	windows-1251	BE	32	nl_BE	Linux	Бельгия
923	S-1	ISO-8859-15	BE	32	fr_BE@euro	Linux	Бельгия
923	S-1	ISO-8859-15	BE	32	nl_BE@euro	Linux	Бельгия
437	S-1	IBM-437	BE	32	-	OS2	Бельгия
850	S-1	IBM-850	BE	32	-	OS2	Бельгия
819	S-1	ISO8859-1	BE	32	fr_BE	SCO	Бельгия
819	S-1	ISO8859-1	BE	32	nl_BE	SCO	Бельгия
819	S-1	ISO8859-1	BE	32	fr_BE	Bc	Бельгия
819	S-1	ISO8859-1	BE	32	nl_BE	Bc	Бельгия
923	S-1	ISO8859-15	BE	32	fr_BE.ISO8859-15	Bc	Бельгия
923	S-1	ISO8859-15	BE	32	nl_BE.ISO8859-15	Bc	Бельгия
1252	S-1	1252	BE	32	-	WIN	Бельгия
915	S-5	ISO8859-5	BG	359	bg_BG	AIX	Болгария
1208	N-1	UTF-8	BG	359	BG_BG	AIX	Болгария
1025	S-5	IBM-1025	BG	359	-	HOST	Болгария
1154	S-5	IBM-1154	BG	359	-	HOST	Болгария
915	S-5	iso88595	BG	359	bg_BG.iso88595	HP	Болгария
855	S-5	IBM-855	BG	359	-	OS2	Болгария
915	S-5	ISO8859-5	BG	359	-	OS2	Болгария
1251	S-5	1251	BG	359	-	WIN	Болгария

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
819	S-1	ISO8859-1	BR	55	pt_BR	AIX	Бразилия
850	S-1	IBM-850	BR	55	-	AIX	Бразилия
923	S-1	ISO8859-15	BR	55	pt_BR.8859-15	AIX	Бразилия
1208	N-1	UTF-8	BR	55	PT_BR	AIX	Бразилия
37	S-1	IBM-37	BR	55	-	HOST	Бразилия
1140	S-1	IBM-1140	BR	55	-	HOST	Бразилия
819	S-1	ISO8859-1	BR	55	-	HP	Бразилия
923	S-1	ISO8859-15	BR	55	-	HP	Бразилия
819	S-1	windows-1251	BR	55	pt_BR	Linux	Бразилия
923	S-1	ISO-8859-15	BR	55	-	Linux	Бразилия
850	S-1	IBM-850	BR	55	-	OS2	Бразилия
819	S-1	ISO8859-1	BR	55	pt_BR	SCO	Бразилия
819	S-1	ISO8859-1	BR	55	pt_BR	Вс	Бразилия
923	S-1	ISO8859-15	BR	55	-	Вс	Бразилия
1252	S-1	1252	BR	55	-	WIN	Бразилия
819	S-1	ISO8859-1	CA	1	fr_CA	AIX	Канада
850	S-1	IBM-850	CA	1	Fr_CA	AIX	Канада
923	S-1	ISO8859-15	CA	1	fr_CA.8859-15	AIX	Канада
1208	N-1	UTF-8	CA	1	FR_CA	AIX	Канада
37	S-1	IBM-37	CA	1	-	HOST	Канада
1140	S-1	IBM-1140	CA	1	-	HOST	Канада
819	S-1	iso88591	CA	1	fr_CA.iso88591	HP	Канада
923	S-1	iso885915	CA	1	-	HP	Канада
1051	S-1	roman8	CA	1	fr_CA.roman8	HP	Канада
819	S-1	windows-1251	CA	1	en_CA	Linux	Канада
923	S-1	ISO-8859-15	CA	1	-	Linux	Канада
850	S-1	IBM-850	CA	1	-	OS2	Канада
863	S-1	IBM-863	CA	2	-	OS2	Канада (Французский)
819	S-1	ISO8859-1	CA	1	en_CA	SCO	Канада
819	S-1	ISO8859-1	CA	1	fr_CA	SCO	Канада
819	S-1	ISO8859-1	CA	1	en_CA	Вс	Канада
923	S-1	ISO8859-15	CA	1	-	Вс	Канада
1252	S-1	1252	CA	1	-	WIN	Канада
1383	D-4	IBM-eucCN	CN	86	zh_CN	AIX	Китай (КНР)
1386	D-4	GBK	CN	86	Zh_CN.GBK	AIX	Китай (КНР)
1208	N-1	UTF-8	CN	86	ZH_CN	AIX	Китай (КНР)
935	D-4	IBM-935	CN	86	-	HOST	Китай (КНР)
1388	D-4	IBM-1388	CN	86	-	HOST	Китай (КНР)

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
1383	D-4	hp15CN	CN	86	zh_CN.hp15CN	HP	Китай (КНР)
1383	D-4	GBK	CN	86	zh_CN.GBK	Linux	Китай (КНР)
1381	D-4	IBM-1381	CN	86	-	OS2	Китай (КНР)
1386	D-4	GBK	CN	86	-	OS2	Китай (КНР)
1383	D-4	eucCN	CN	86	zh_CN	SCO	Китай (КНР)
1383	D-4	eucCN	CN	86	zh_CN.eucCN	SCO	Китай (КНР)
1383	D-4	gb2312	CN	86	zh	Bc	Китай (КНР)
1208	N-1	UTF-8	CN	86	zh.UTF-8	Bc	Китай (КНР)
1381	D-4	IBM-1381	CN	86	-	WIN	Китай (КНР)
1386	D-4	GBK	CN	86	-	WIN	Китай (КНР)
1392/5488	D-4		CN	86	-		Китай (КНР)
См. примечание 1 на стр. 258.							
912	S-2	ISO8859-2	HR	385	hr_HR	AIX	Хорватия
1208	N-1	UTF-8	HR	385	HR_HR	AIX	Хорватия
870	S-2	IBM-870	HR	385	-	HOST	Хорватия
1153	S-2	IBM-1153	HR	385	-	HOST	Хорватия
912	S-2	iso88592	HR	385	hr_HR.iso88592	HP	Хорватия
912	S-2	ISO-8859-2	HR	385	hr_HR	Linux	Хорватия
852	S-2	IBM-852	HR	385	-	OS2	Хорватия
912	S-2	ISO8859-2	HR	385	hr_HR.ISO8859-2	SCO	Хорватия
1250	S-2	1250	HR	385	-	WIN	Хорватия
912	S-2	ISO8859-2	CZ	421	cs_CZ	AIX	Чехия
1208	N-1	UTF-8	CZ	421	CS_CZ	AIX	Чехия
870	S-2	IBM-870	CZ	421	-	HOST	Чехия
1153	S-2	IBM-1153	CZ	421	-	HOST	Чехия
912	S-2	iso88592	CZ	421	cs_CZ.iso88592	HP	Чехия
912	S-2	ISO-8859-2	CZ	421	cs_CZ	Linux	Чехия
852	S-2	IBM-852	CZ	421	-	OS2	Чехия
912	S-2	ISO8859-2	CZ	421	cs_CZ.ISO8859-2	SCO	Чехия
1250	S-2	1250	CZ	421	-	WIN	Чехия
819	S-1	ISO8859-1	DK	45	da_DK	AIX	Дания
850	S-1	IBM-850	DK	45	Da_DK	AIX	Дания
923	S-1	ISO8859-15	DK	45	da_DK.8859-15	AIX	Дания
1208	N-1	UTF-8	DK	45	DA_DK	AIX	Дания
277	S-1	IBM-277	DK	45	-	HOST	Дания
1142	S-1	IBM-1142	DK	45	-	HOST	Дания
819	S-1	iso88591	DK	45	da_DK.iso88591	HP	Дания
923	S-1	iso885915	DK	45	_	HP	Дания
1051	S-1	roman8	DK	45	da_DK.roman8	HP	Дания

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
819	S-1	windows-1251	DK	45	da_DK	Linux	Дания
923	S-1	ISO-8859-15	DK	45	-	Linux	Дания
850	S-1	IBM-850	DK	45	-	OS2	Дания
819	S-1	ISO8859-1	DK	45	da	SCO	Дания
819	S-1	ISO8859-1	DK	45	da_DA	SCO	Дания
819	S-1	ISO8859-1	DK	45	da_DK	SCO	Дания
819	S-1	ISO8859-1	DK	45	da	Вс	Дания
923	S-1	ISO8859-15	DK	45	da.ISO8859-15	Вс	Дания
1252	S-1	1252	DK	45	-	WIN	Дания
922	S-10	IBM-922	EE	372	Et_EE	AIX	Эстония
1208	N-1	UTF-8	EE	372	ET_EE	AIX	Эстония
1122	S-10	IBM-1122	EE	372	-	HOST	Эстония
1157	S-10	IBM-1157	EE	372	-	HOST	Эстония
922	S-10	IBM-922	EE	372	-	OS2	Эстония
1257	S-10	1257	EE	372	-	WIN	Эстония
819	S-1	ISO8859-1	FI	358	fi_FI	AIX	Финляндия
850	S-1	IBM-850	FI	358	Fi_FI	AIX	Финляндия
923	S-1	ISO8859-15	FI	358	fi_FI.8859-15	AIX	Финляндия
1208	N-1	UTF-8	FI	358	FI_FI	AIX	Финляндия
278	S-1	IBM-278	FI	358	-	HOST	Финляндия
1143	S-1	IBM-1143	FI	358	-	HOST	Финляндия
819	S-1	iso88591	FI	358	fi_FI.iso88591	HP	Финляндия
923	S-1	iso885915	FI	358	-	HP	Финляндия
1051	S-1	roman8	FI	358	fi-FI.roman8	HP	Финляндия
819	S-1	windows-1251	FI	358	fi_FI	Linux	Финляндия
923	S-1	ISO-8859-15	FI	358	fi_FI@euro	Linux	Финляндия
437	S-1	IBM-437	FI	358	-	OS2	Финляндия
850	S-1	IBM-850	FI	358	-	OS2	Финляндия
819	S-1	ISO8859-1	FI	358	fi	SCO	Финляндия
819	S-1	ISO8859-1	FI	358	fi_FI	SCO	Финляндия
819	S-1	ISO8859-1	FI	358	sv_FI	SCO	Финляндия
819	S-1	ISO8859-1	FI	358	-	Вс	Финляндия
923	S-1	ISO8859-15	FI	358	fi.ISO8859-15	Вс	Финляндия
1252	S-1	1252	FI	358	-	WIN	Финляндия
915	S-5	ISO8859-5	MK	389	mk_MK	AIX	Македония
1208	N-1	UTF-8	MK	389	MK_MK	AIX	Македония
1025	S-5	IBM-1025	MK	389	-	HOST	Македония
1154	S-5	IBM-1154	MK	389	-	HOST	Македония
915	S-5	iso88595	MK	389	-	HP	Македония

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
855	S-5	IBM-855	MK	389	-	OS2	Македония
915	S-5	ISO8859-5	MK	389	-	OS2	Македония
1251	S-5	1251	MK	389	-	WIN	Македония
819	S-1	ISO8859-1	FR	33	fr_FR	AIX	Франция
850	S-1	IBM-850	FR	33	Fr_FR	AIX	Франция
923	S-1	ISO8859-15	FR	33	fr_FR.8859-15	AIX	Франция
1208	N-1	UTF-8	FR	33	FR_FR	AIX	Франция
297	S-1	IBM-297	FR	33	-	HOST	Франция
1147	S-1	IBM-1147	FR	33	-	HOST	Франция
819	S-1	iso88591	FR	33	fr_FR.iso88591	HP	Франция
923	S-1	iso885915	FR	33	-	HP	Франция
1051	S-1	roman8	FR	33	fr_FR.roman8	HP	Франция
819	S-1	windows-1251	FR	33	fr_FR	Linux	Франция
923	S-1	ISO-8859-15	FR	33	fr_FR@euro	Linux	Франция
437	S-1	IBM-437	FR	33	-	OS2	Франция
850	S-1	IBM-850	FR	33	-	OS2	Франция
819	S-1	ISO8859-1	FR	33	fr	SCO	Франция
819	S-1	ISO8859-1	FR	33	fr_FR	SCO	Франция
819	S-1	ISO8859-1	FR	33	fr	Bc	Франция
923	S-1	ISO8859-15	FR	33	fr.ISO8859-15	Bc	Франция
1208	N-1	UTF-8	FR	33	fr.UTF-8	Bc	Франция
1252	S-1	1252	FR	33	-	WIN	Франция
819	S-1	ISO8859-1	DE	49	de_DE	AIX	Германия
850	S-1	IBM-850	DE	49	De_DE	AIX	Германия
923	S-1	ISO8859-15	DE	49	de_DE.8859-15	AIX	Германия
1208	N-1	UTF-8	DE	49	DE_DE	AIX	Германия
273	S-1	IBM-273	DE	49	-	HOST	Германия
1141	S-1	IBM-1141	DE	49	-	HOST	Германия
819	S-1	iso88591	DE	49	de_DE.iso88591	HP	Германия
923	S-1	iso885915	DE	49	_	HP	Германия
1051	S-1	roman8	DE	49	de_DE.roman8	HP	Германия
819	S-1	windows-1251	DE	49	de_DE	Linux	Германия
923	S-1	ISO-8859-15	DE	49	de_DE@euro	Linux	Германия
437	S-1	IBM-437	DE	49	-	OS2	Германия
850	S-1	IBM-850	DE	49	-	OS2	Германия
819	S-1	ISO8859-1	DE	49	de	SCO	Германия
819	S-1	ISO8859-1	DE	49	de_DE	SCO	Германия
819	S-1	ISO8859-1	DE	49	de	Bc	Германия
923	S-1	ISO8859-15	DE	49	de.ISO8859-15	Bc	Германия
1208	N-1	UTF-8	DE	49	de.UTF-8	Bc	Германия

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
1252	S-1	1252	DE	49	-	WIN	Германия
813	S-7	ISO8859-7	GR	30	el_GR	AIX	Греция
1208	N-1	UTF-8	GR	30	EL_GR	AIX	Греция
423	S-7	IBM-423	GR	30	-	HOST	Греция
875	S-7	IBM-875	GR	30	-	HOST	Греция
813	S-7	iso88597	GR	30	el_GR.iso88597	HP	Греция
813	S-7	ISO-8859-7	GR	30	el_GR	Linux	Греция
813	S-7	ISO8859-7	GR	30	-	OS2	Греция
869	S-7	IBM-869	GR	30	-	OS2	Греция
813	S-7	ISO8859-7	GR	30	el_GR.ISO8859-7	SCO	Греция
737	S-7	737	GR	30	-	WIN	Греция
1253	S-7	1253	GR	30	-	WIN	Греция
912	S-2	ISO8859-2	HU	36	hu_HU	AIX	Венгрия
1208	N-1	UTF-8	HU	36	HU_HU	AIX	Венгрия
870	S-2	IBM-870	HU	36	-	HOST	Венгрия
1153	S-2	IBM-1153	HU	36	-	HOST	Венгрия
912	S-2	iso88592	HU	36	hu_HU.iso88592	HP	Венгрия
912	S-2	ISO-8859-2	HU	36	hu_HU	Linux	Венгрия
852	S-2	IBM-852	HU	36	-	OS2	Венгрия
912	S-2	ISO8859-2	HU	36	hu_HU.ISO8859-2	SCO	Венгрия
1250	S-2	1250	HU	36	-	WIN	Венгрия
819	S-1	ISO8859-1	IS	354	is_IS	AIX	Исландия
850	S-1	IBM-850	IS	354	Is_IS	AIX	Исландия
923	S-1	ISO8859-15	IS	354	is_IS.8859-15	AIX	Исландия
1208	N-1	UTF-8	IS	354	IS_IS	AIX	Исландия
871	S-1	IBM-871	IS	354	-	HOST	Исландия
1149	S-1	IBM-1149	IS	354	-	HOST	Исландия
819	S-1	iso88591	IS	354	is_IS.iso88591	HP	Исландия
923	S-1	iso885915	IS	354	-	HP	Исландия
1051	S-1	roman8	IS	354	is_IS.roman8	HP	Исландия
819	S-1	windows-1251	IS	354	is_IS	Linux	Исландия
923	S-1	ISO-8859-15	IS	354	-	Linux	Исландия
850	S-1	IBM-850	IS	354	-	OS2	Исландия
819	S-1	ISO8859-1	IS	354	is	SCO	Исландия
819	S-1	ISO8859-1	IS	354	is_IS	SCO	Исландия
819	S-1	ISO8859-1	IS	354	-	Bc	Исландия
923	S-1	ISO8859-15	IS	354	-	Bc	Исландия
1252	S-1	1252	IS	354	-	WIN	Исландия

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
806	S-13	IBM-806	IN	91	hi_IN	-	Индия
1137	S-13	IBM-1137	IN	91	-	HOST	Индия
1252	S-1	1252	ID	62	-	WIN	Индонезия
819	S-1	ISO8859-1	IE	353	-	AIX	Ирландия
850	S-1	IBM-850	IE	353	-	AIX	Ирландия
923	S-1	ISO8859-15	IE	353	-	AIX	Ирландия
1208	N-1	UTF-8	IE	353	-	AIX	Ирландия
285	S-1	IBM-285	IE	353	-	HOST	Ирландия
1146	S-1	IBM-1146	IE	353	-	HOST	Ирландия
819	S-1	iso88591	IE	353	-	HP	Ирландия
923	S-1	iso885915	IE	353	-	HP	Ирландия
1051	S-1	roman8	IE	353	-	HP	Ирландия
819	S-1	windows-1251	IE	353	en_IE	Linux	Ирландия
923	S-1	ISO-8859-15	IE	353	en_IE@euro	Linux	Ирландия
437	S-1	IBM-437	IE	353	-	OS2	Ирландия
850	S-1	IBM-850	IE	353	-	OS2	Ирландия
819	S-1	ISO8859-1	IE	353	en_IE.ISO8859-1	SCO	Ирландия
819	S-1	ISO8859-1	IE	353	en_IE	Bc	Ирландия
923	S-1	ISO8859-15	IE	353	en_IE.ISO8859-15	Bc	Ирландия
1252	S-1	1252	IE	353	-	WIN	Ирландия
856	S-8	IBM-856	IL	972	Iw_IL	AIX	Израиль
916	S-8	ISO8859-8	IL	972	iw_IL	AIX	Израиль
1208	N-1	UTF-8	IL	972	HE-IL	AIX	Израиль
916	S-8	ISO-8859-8	IL	972	iw_IL	Linux	Израиль
424	S-8	IBM-424	IL	972	-	HOST	Израиль
862	S-8	IBM-862	IL	972	-	OS2	Израиль
1255	S-8	1255	IL	972	-	WIN	Израиль
819	S-1	ISO8859-1	IT	39	it_IT	AIX	Италия
850	S-1	IBM-850	IT	39	It_IT	AIX	Италия
923	S-1	ISO8859-15	IT	39	it_IT.8859-15	AIX	Италия
1208	N-1	UTF-8	IT	39	It_IT	AIX	Италия
280	S-1	IBM-280	IT	39	-	HOST	Италия
1144	S-1	IBM-1144	IT	39	-	HOST	Италия
819	S-1	iso88591	IT	39	it_IT.iso88591	HP	Италия
923	S-1	iso885915	IT	39	_	HP	Италия
1051	S-1	roman8	IT	39	it_IT.roman8	HP	Италия
819	S-1	windows-1251	IT	39	it_IT	Linux	Италия
923	S-1	ISO-8859-15	IT	39	it_IT@euro	Linux	Италия

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
437	S-1	IBM-437	IT	39	-	OS2	Италия
850	S-1	IBM-850	IT	39	-	OS2	Италия
819	S-1	ISO8859-1	IT	39	it	SCO	Италия
819	S-1	ISO8859-1	IT	39	it_IT	SCO	Италия
819	S-1	ISO8859-1	IT	39	it	Вс	Италия
923	S-1	ISO8859-15	IT	39	it.ISO8859-15	Вс	Италия
1208	N-1	UTF-8	IT	39	it.UTF-8	Вс	Италия
1252	S-1	1252	IT	39	-	WIN	Италия
932	D-1	IBM-932	JP	81	Ja_JP	AIX	Япония
943	D-1	IBM-943	JP	81	Ja_JP	AIX	Япония
См. примечание 2 на стр. 258.							
954	D-1	IBM-eucJP	JP	81	ja_JP	AIX	Япония
1208	N-1	UTF-8	JP	81	JA_JP	AIX	Япония
930	D-1	IBM-930	JP	81	-	HOST	Япония
939	D-1	IBM-939	JP	81	-	HOST	Япония
5026	D-1	IBM-5026	JP	81	-	HOST	Япония
5035	D-1	IBM-5035	JP	81	-	HOST	Япония
1390	D-1		JP	81	-	HOST	Япония
1399	D-1		JP	81	-	HOST	Япония
954	D-1	eucJP	JP	81	ja_JP.eucJP	HP	Япония
5039	D-1	SJIS	JP	81	ja_JP.SJIS	HP	Япония
954	D-1	EUC-JP	JP	81	ja_JP	Linux	Япония
932	D-1	IBM-932	JP	81	-	OS2	Япония
942	D-1	IBM-942	JP	81	-	OS2	Япония
943	D-1	IBM-943	JP	81	-	OS2	Япония
954	D-1	eucJP	JP	81	ja	SCO	Япония
954	D-1	eucJP	JP	81	ja_JP	SCO	Япония
954	D-1	eucJP	JP	81	ja_JP.EUC	SCO	Япония
954	D-1	eucJP	JP	81	ja_JP.eucJP	SCO	Япония
943	D-1	IBM-943	JP	81	ja_JP.PCK	Вс	Япония
954	D-1	eucJP	JP	81	ja	Вс	Япония
954	D-1	eucJP	JP	81	японская	Вс	Япония
1208	N-1	UTF-8	JP	81	ja_JP.UTF-8	Вс	Япония
943	D-1	IBM-943	JP	81	-	WIN	Япония
1394	D-1		JP	81	-		Япония
См. примечание 3 на стр. 258.							
1251	S-5	1251	KZ	7	-	WIN	Казахстан
970	D-3	IBM-eucKR	KR	82	ko_KR	AIX	Корея, Южная
1208	N-1	UTF-8	KR	82	KO_KR	AIX	Корея, Южная

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
933	D-3	IBM-933	KR	82	-	HOST	Корея, Южная
1364	D-3	IBM-1364	KR	82	-	HOST	Корея, Южная
970	D-3	eucKR	KR	82	ko_KR.eucKR	HP	Корея, Южная
970	D-3	EUC-KR	KR	82	ko_KR	Linux	Корея, Южная
949	D-3	IBM-949	KR	82	-	OS2	Корея, Южная
970	D-3	eucKR	KR	82	ko_KR.eucKR	SGI	Корея, Южная
970	D-3	5601	KR	82	ko	Вс	Корея, Южная
1208	N-1	UTF-8	KR	82	ko.UTF-8	Вс	Корея, Южная
1363	D-3	1363	KR	82	-	WIN	Корея, Южная
819	S-1	ISO8859-1	Lat	3	-	AIX	Латинская Америка
850	S-1	IBM-850	Lat	3	-	AIX	Латинская Америка
923	S-1	ISO8859-15	Lat	3	-	AIX	Латинская Америка
1208	N-1	UTF-8	Lat	3	-	AIX	Латинская Америка
284	S-1	IBM-284	Lat	3	-	HOST	Латинская Америка
1145	S-1	IBM-1145	Lat	3	-	HOST	Латинская Америка
819	S-1	iso88591	Lat	3	-	HP	Латинская Америка
923	S-1	iso885915	Lat	3	-	HP	Латинская Америка
1051	S-1	roman8	Lat	3	-	HP	Латинская Америка
819	S-1	windows-1251	Lat	3	-	Linux	Латинская Америка
923	S-1	ISO-8859-15	Lat	3	-	Linux	Латинская Америка
437	S-1	IBM-437	Lat	3	-	OS2	Латинская Америка
850	S-1	IBM-850	Lat	3	-	OS2	Латинская Америка
819	S-1	ISO8859-1	Lat	3	-	Вс	Латинская Америка
923	S-1	ISO8859-15	Lat	3	-	Вс	Латинская Америка
1252	S-1	1252	Lat	3	-	WIN	Латинская Америка

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
921	S-10	IBM-921	LV	371	Lv_LV	AIX	Латвия
1208	N-1	UTF-8	LV	371	LV_LV	AIX	Латвия
1112	S-10	IBM-1112	LV	371	-	HOST	Латвия
1156	S-10	IBM-1156	LV	371	-	HOST	Латвия
921	S-10	IBM-921	LV	371	-	OS2	Латвия
1257	S-10	1257	LV	371	-	WIN	Латвия
921	S-10	IBM-921	LT	370	Lt_LT	AIX	Литва
1208	N-1	UTF-8	LT	370	LT_LT	AIX	Литва
1112	S-10	IBM-1112	LT	370	-	HOST	Литва
1156	S-10	IBM-1156	LT	370	-	HOST	Литва
921	S-10	IBM-921	LT	370	-	OS2	Литва
1257	S-10	1257	LT	370	-	WIN	Литва
1252	S-1	1252	ID	60	-	WIN	Малайзия
819	S-1	ISO8859-1	NL	31	nl_NL	AIX	Нидерланды
850	S-1	IBM-850	NL	31	Nl_NL	AIX	Нидерланды
923	S-1	ISO8859-15	NL	31	nl_NL.8859-15	AIX	Нидерланды
1208	N-1	UTF-8	NL	31	NL_NL	AIX	Нидерланды
37	S-1	IBM-37	NL	31	-	HOST	Нидерланды
1140	S-1	IBM-1140	NL	31	-	HOST	Нидерланды
819	S-1	iso88591	NL	31	nl_NL.iso88591	HP	Нидерланды
923	S-1	iso885915	NL	31	_	HP	Нидерланды
1051	S-1	roman8	NL	31	nl_NL.roman8	HP	Нидерланды
819	S-1	windows-1251	NL	31	nl_NL	Linux	Нидерланды
923	S-1	ISO-8859-15	NL	31	nl_NL@euro	Linux	Нидерланды
437	S-1	IBM-437	NL	31	-	OS2	Нидерланды
850	S-1	IBM-850	NL	31	-	OS2	Нидерланды
819	S-1	ISO8859-1	NL	31	nl	SCO	Нидерланды
819	S-1	ISO8859-1	NL	31	nl_NL	SCO	Нидерланды
819	S-1	ISO8859-1	NL	31	nl	Bc	Нидерланды
923	S-1	ISO8859-15	NL	31	nl.ISO8859-15	Bc	Нидерланды
1252	S-1	1252	NL	31	-	WIN	Нидерланды
819	S-1	ISO8859-1	NZ	64	-	AIX	Новая Зеландия
850	S-1	IBM-850	NZ	64	-	AIX	Новая Зеландия
923	S-1	ISO8859-15	NZ	64	-	AIX	Новая Зеландия
1208	N-1	UTF-8	NZ	64	-	AIX	Новая Зеландия
37	S-1	IBM-37	NZ	64	-	HOST	Новая Зеландия
1140	S-1	IBM-1140	NZ	64	-	HOST	Новая Зеландия
819	S-1	ISO8859-1	NZ	64	-	HP	Новая Зеландия

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
923	S-1	ISO8859-15	NZ	64	-	HP	Новая Зеландия
850	S-1	IBM-850	NZ	64	-	OS2	Новая Зеландия
819	S-1	ISO8859-1	NZ	64	en_NZ	SCO	Новая Зеландия
819	S-1	ISO8859-1	NZ	64	en_NZ	Bc	Новая Зеландия
923	S-1	ISO8859-15	NZ	64	-	Bc	Новая Зеландия
1252	S-1	1252	NZ	64	-	WIN	Новая Зеландия
819	S-1	ISO8859-1	NO	47	no_NO	AIX	Норвегия
850	S-1	IBM-850	NO	47	No_NO	AIX	Норвегия
923	S-1	ISO8859-15	NO	47	no_NO.8859-15	AIX	Норвегия
1208	N-1	UTF-8	NO	47	NO_NO	AIX	Норвегия
277	S-1	IBM-277	NO	47	-	HOST	Норвегия
1142	S-1	IBM-1142	NO	47	-	HOST	Норвегия
819	S-1	iso88591	NO	47	no_NO.iso88591	HP	Норвегия
923	S-1	iso885915	NO	47	-	HP	Норвегия
1051	S-1	roman8	NO	47	no_NO.roman8	HP	Норвегия
819	S-1	windows-1251	NO	47	no_NO	Linux	Норвегия
923	S-1	ISO-8859-15	NO	47	-	Linux	Норвегия
850	S-1	IBM-850	NO	47	-	OS2	Норвегия
819	S-1	ISO8859-1	NO	47	no	SCO	Норвегия
819	S-1	ISO8859-1	NO	47	no_NO	SCO	Норвегия
819	S-1	ISO8859-1	NO	47	no	Bc	Норвегия
923	S-1	ISO8859-15	NO	47	-	Bc	Норвегия
1252	S-1	1252	NO	47	-	WIN	Норвегия
912	S-2	ISO8859-2	PL	48	pl_PL	AIX	Польша
1208	N-1	UTF-8	PL	48	PL_PL	AIX	Польша
870	S-2	IBM-870	PL	48	-	HOST	Польша
1153	S-2	IBM-1153	PL	48	-	HOST	Польша
912	S-2	iso88592	PL	48	pl_PL.iso88592	HP	Польша
912	S-2	ISO-8859-2	PL	48	pl_PL	Linux	Польша
852	S-2	IBM-852	PL	48	-	OS2	Польша
912	S-2	ISO8859-2	PL	48	pl_PL.ISO8859-2	SCO	Польша
1250	S-2	1250	PL	48	-	WIN	Польша
819	S-1	ISO8859-1	PT	351	pt_PT	AIX	Португалия
850	S-1	IBM-850	PT	351	Pt_PT	AIX	Португалия
923	S-1	ISO8859-15	PT	351	pt_PT.8859-15	AIX	Португалия
1208	N-1	UTF-8	PT	351	PT_PT	AIX	Португалия
37	S-1	IBM-37	PT	351	-	HOST	Португалия
1140	S-1	IBM-1140	PT	351	-	HOST	Португалия
819	S-1	iso88591	PT	351	pt_PT.iso88591	HP	Португалия

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
923	S-1	iso885915	PT	351	-	HP	Португалия
1051	S-1	roman8	PT	351	pt_PT.roman8	HP	Португалия
819	S-1	windows-1251	PT	351	pt_PT	Linux	Португалия
923	S-1	ISO-8859-15	PT	351	pt_PT@euro	Linux	Португалия
850	S-1	IBM-850	PT	351	-	OS2	Португалия
860	S-1	IBM-860	PT	351	-	OS2	Португалия
819	S-1	ISO8859-1	PT	351	pt	SCO	Португалия
819	S-1	ISO8859-1	PT	351	pt_PT	SCO	Португалия
819	S-1	ISO8859-1	PT	351	pt	Bc	Португалия
923	S-1	ISO8859-15	PT	351	pt.ISO8859-15	Bc	Португалия
1252	S-1	1252	PT	351	-	WIN	Португалия
912	S-2	ISO8859-2	RO	40	ro_RO	AIX	Румыния
1208	N-1	UTF-8	RO	40	RO_RO	AIX	Румыния
870	S-2	IBM-870	RO	40	-	HOST	Румыния
1153	S-2	IBM-1153	RO	40	-	HOST	Румыния
912	S-2	iso88592	RO	40	ro_RO.iso88592	HP	Румыния
912	S-2	ISO-8859-2	RO	40	ro_RO	Linux	Румыния
852	S-2	IBM-852	RO	40	-	OS2	Румыния
912	S-2	ISO8859-2	RO	40	ro_RO.ISO8859-2	SCO	Румыния
1250	S-2	1250	RO	40	-	WIN	Румыния
915	S-5	ISO8859-5	RU	7	ru_RU	AIX	Россия
1208	N-1	UTF-8	RU	7	RU_RU	AIX	Россия
1025	S-5	IBM-1025	RU	7	-	HOST	Россия
1154	S-5	IBM-1154	RU	7	-	HOST	Россия
915	S-5	iso88595	RU	7	ru_RU.iso88595	HP	Россия
915	S-5	ISO-8859-5	RU	7	ru_RU	Linux	Россия
866	S-5	IBM-866	RU	7	-	OS2	Россия
915	S-5	ISO8859-5	RU	7	-	OS2	Россия
915	S-5	ISO8859-5	RU	7	ru_RU.ISO8859-5	SCO	Россия
1251	S-5	1251	RU	7	-	WIN	Россия
915	S-5	ISO8859-5	SP	381	sr_SP	AIX	Сербия/Черногория
1208	N-1	UTF-8	SP	381	SR_SP	AIX	Сербия/Черногория
1025	S-5	IBM-1025	SP	381	-	HOST	Сербия/Черногория
1154	S-5	IBM-1154	SP	381	-	HOST	Сербия/Черногория
915	S-5	iso88595	SP	381	-	HP	Сербия/Черногория
855	S-5	IBM-855	SP	381	-	OS2	Сербия/Черногория
915	S-5	ISO8859-5	SP	381	-	OS2	Сербия/Черногория
1251	S-5	1251	SP	381	-	WIN	Сербия/Черногория

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
912	S-2	ISO8859-2	SK	422	sk_SK	AIX	Словакия
1208	N-1	UTF-8	SK	422	SK_SK	AIX	Словакия
870	S-2	IBM-870	SK	422	-	HOST	Словакия
1153	S-2	IBM-1153	SK	422	-	HOST	Словакия
912	S-2	iso88592	SK	422	sk_SK.iso88592	HP	Словакия
852	S-2	IBM-852	SK	422	-	OS2	Словакия
912	S-2	ISO8859-2	SK	422	sk_SK.ISO8859-2	SCO	Словакия
1250	S-2	1250	SK	422	-	WIN	Словакия
912	S-2	ISO8859-2	SI	386	sl_SI	AIX	Словения
1208	N-1	UTF-8	SI	386	SL_SI	AIX	Словения
870	S-2	IBM-870	SI	386	-	HOST	Словения
1153	S-2	IBM-1153	SI	386	-	HOST	Словения
912	S-2	iso88592	SI	386	sl_SI.iso88592	HP	Словения
912	S-2	ISO-8859-2	SI	386	sl_SI	Linux	Словения
852	S-2	IBM-852	SI	386	-	OS2	Словения
912	S-2	ISO8859-2	SI	386	sl_SI.ISO8859-2	SCO	Словения
1250	S-2	1250	SI	386	-	WIN	Словения
819	S-1	ISO8859-1	ZA	27	en_ZA	AIX	Южная Африка
850	S-1	IBM-850	ZA	27	En_ZA	AIX	Южная Африка
923	S-1	ISO8859-15	ZA	27	en_ZA.8859-15	AIX	Южная Африка
1208	N-1	UTF-8	ZA	27	EN_ZA	AIX	Южная Африка
285	S-1	IBM-285	ZA	27	-	HOST	Южная Африка
1146	S-1	IBM-1146	ZA	27	-	HOST	Южная Африка
819	S-1	iso88591	ZA	27	-	HP	Южная Африка
923	S-1	iso885915	ZA	27	-	HP	Южная Африка
1051	S-1	roman8	ZA	27	-	HP	Южная Африка
437	S-1	IBM-437	ZA	27	-	OS2	Южная Африка
850	S-1	IBM-850	ZA	27	-	OS2	Южная Африка
819	S-1	ISO8859-1	ZA	27	en_ZA.ISO8859-1	SCO	Южная Африка
819	S-1	ISO8859-1	ZA	27	-	Bc	Южная Африка
923	S-1	ISO8859-15	ZA	27	-	Bc	Южная Африка
1252	S-1	1252	ZA	27	-	WIN	Южная Африка
819	S-1	ISO8859-1	ES	34	es_ES	AIX	Испания
819	S-1	ISO8859-1	ES	34	ca_ES	AIX	Испания (Каталонский)
850	S-1	IBM-850	ES	34	Es_ES	AIX	Испания
850	S-1	IBM-850	ES	34	Ca_ES	AIX	Испания (Каталонский)
923	S-1	ISO8859-15	ES	34	es_ES.8859-15	AIX	Испания

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
923	S-1	ISO8859-15	ES	34	ca_ES.8859-15	AIX	Испания (Каталонский)
1208	N-1	UTF-8	ES	34	ES_ES	AIX	Испания
1208	N-1	UTF-8	ES	34	CA_ES	AIX	Испания (Каталонский)
284	S-1	IBM-284	ES	34	-	HOST	Испания
1145	S-1	IBM-1145	ES	34	-	HOST	Испания
819	S-1	iso88591	ES	34	es_ES.iso88591	HP	Испания
923	S-1	iso885915	ES	34	-	HP	Испания
1051	S-1	roman8	ES	34	es_ES.roman8	HP	Испания
819	S-1	windows-1251	ES	34	es_ES	Linux	Испания
923	S-1	ISO-8859-15	ES	34	es_ES@euro	Linux	Испания
437	S-1	IBM-437	ES	34	-	OS2	Испания
850	S-1	IBM-850	ES	34	-	OS2	Испания
819	S-1	ISO8859-1	ES	34	es	SCO	Испания
819	S-1	ISO8859-1	ES	34	es_ES	SCO	Испания
819	S-1	ISO8859-1	ES	34	es	Вс	Испания
923	S-1	ISO8859-15	ES	34	es.ISO8859-15	Вс	Испания
1208	N-1	UTF-8	ES	34	es.UTF-8	Вс	Испания
1252	S-1	1252	ES	34	-	WIN	Испания
819	S-1	ISO8859-1	SE	46	sv_SE	AIX	Швеция
850	S-1	IBM-850	SE	46	Sv_SE	AIX	Швеция
923	S-1	ISO8859-15	SE	46	sv_SE.8859-15	AIX	Швеция
1208	N-1	UTF-8	SE	46	SV_SE	AIX	Швеция
278	S-1	IBM-278	SE	46	-	HOST	Швеция
1143	S-1	IBM-1143	SE	46	-	HOST	Швеция
819	S-1	iso88591	SE	46	sv_SE.iso88591	HP	Швеция
923	S-1	iso885915	SE	46	-	HP	Швеция
1051	S-1	roman8	SE	46	sv_SE.roman8	HP	Швеция
819	S-1	windows-1251	SE	46	sv_SE	Linux	Швеция
923	S-1	ISO-8859-15	SE	46	-	Linux	Швеция
437	S-1	IBM-437	SE	46	-	OS2	Швеция
850	S-1	IBM-850	SE	46	-	OS2	Швеция
819	S-1	ISO8859-1	SE	46	sv	SCO	Швеция
819	S-1	ISO8859-1	SE	46	sv_SE	SCO	Швеция
819	S-1	ISO8859-1	SE	46	sv	Вс	Швеция
923	S-1	ISO8859-15	SE	46	sv.ISO8859-15	Вс	Швеция
1208	N-1	UTF-8	SE	46	sv.UTF-8	Вс	Швеция
1252	S-1	1252	SE	46	-	WIN	Швеция
819	S-1	ISO8859-1	CH	41	de_CH	AIX	Швейцария

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
850	S-1	IBM-850	CH	41	De_CH	AIX	Швейцария
923	S-1	ISO8859-15	CH	41	de_CH.8859-15	AIX	Швейцария
1208	N-1	UTF-8	CH	41	DE_CH	AIX	Швейцария
500	S-1	IBM-500	CH	41	-	HOST	Швейцария
1148	S-1	IBM-1148	CH	41	-	HOST	Швейцария
819	S-1	iso88591	CH	41	-	HP	Швейцария
923	S-1	iso885915	CH	41	-	HP	Швейцария
1051	S-1	roman8	CH	41	-	HP	Швейцария
819	S-1	windows-1251	CH	41	de_CH	Linux	Швейцария
923	S-1	ISO-8859-15	CH	41	-	Linux	Швейцария
437	S-1	IBM-437	CH	41	-	OS2	Швейцария
850	S-1	IBM-850	CH	41	-	OS2	Швейцария
819	S-1	ISO8859-1	CH	41	de_CH	SCO	Швейцария
819	S-1	ISO8859-1	CH	41	fr_CH	SCO	Швейцария
819	S-1	ISO8859-1	CH	41	it_CH	SCO	Швейцария
819	S-1	ISO8859-1	CH	41	de_CH	Bc	Швейцария
923	S-1	ISO8859-15	CH	41	-	Bc	Швейцария
1252	S-1	1252	CH	41	-	WIN	Швейцария
950	D-2	big5	TW	88	Zh_TW	AIX	Тайвань
964	D-2	IBM-eucTW	TW	88	zh_TW	AIX	Тайвань
1208	N-1	UTF-8	TW	88	ZH_TW	AIX	Тайвань
937	D-2	IBM-937	TW	88	-	HOST	Тайвань
1371	D-2	IBM-1371	TW	88	-	HOST	Тайвань
950	D-2	big5	TW	88	zh_TW.big5	HP	Тайвань
964	D-2	eucTW	TW	88	zh_TW.eucTW	HP	Тайвань
950	D-2	BIG5	TW	88	zh_TW	Linux	Тайвань
938	D-2	IBM-938	TW	88	-	OS2	Тайвань
948	D-2	IBM-948	TW	88	-	OS2	Тайвань
950	D-2	big5	TW	88	-	OS2	Тайвань
950	D-2	big5	TW	88	zh_TW.BIG5	Bc	Тайвань
См. примечание 8 на стр. 259.							
964	D-2	cns11643	TW	88	zh_TW	Bc	Тайвань
1208	N-1	UTF-8	TW	88	zh_TW.UTF-8	Bc	Тайвань
950	D-2	big5	TW	88	-	WIN	Тайвань
874	S-20	TIS620-1	TH	66	th_TH	AIX	Таиланд
1208	N-1	UTF-8	TH	66	TH_TH	AIX	Таиланд
838	S-20	IBM-838	TH	66	-	HOST	Таиланд
1160	S-20	IBM-1160	TH	66	-	HOST	Таиланд
874	S-20	tis620	TH	66	th_TH.tis620	HP	Таиланд
874	S-20	TIS620-1	TH	66	-	OS2	Таиланд

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
874	S-20	TIS620-1	TH	66	-	WIN	Таиланд
920	S-9	ISO8859-9	TR	90	tr_TR	AIX	Турция
1208	N-1	UTF-8	TR	90	TR_TR	AIX	Турция
1026	S-9	IBM-1026	TR	90	-	HOST	Турция
1155	S-9	IBM-1155	TR	90	-	HOST	Турция
920	S-9	iso88599	TR	90	tr_TR.iso88599	HP	Турция
920	S-9	ISO-8859-9	TR	90	tr_TR	Linux	Турция
857	S-9	IBM-857	TR	90	-	OS2	Турция
920	S-9	ISO8859-9	TR	90	tr_TR.ISO8859-9	SCO	Турция
1254	S-9	1254	TR	90	-	WIN	Турция
819	S-1	ISO8859-1	GB	44	en_GB	AIX	Великобритания
850	S-1	IBM-850	GB	44	En_GB	AIX	Великобритания
923	S-1	ISO8859-15	GB	44	en_GB.8859-15	AIX	Великобритания
1208	N-1	UTF-8	GB	44	EN_GB	AIX	Великобритания
285	S-1	IBM-285	GB	44	-	HOST	Великобритания
1146	S-1	IBM-1146	GB	44	-	HOST	Великобритания
819	S-1	iso88591	GB	44	en_GB.iso88591	HP	Великобритания
923	S-1	iso885915	GB	44	-	HP	Великобритания
1051	S-1	roman8	GB	44	en_GB.roman8	HP	Великобритания
819	S-1	windows-1251	GB	44	en_GB	Linux	Великобритания
923	S-1	ISO-8859-15	GB	44	-	Linux	Великобритания
437	S-1	IBM-437	GB	44	-	OS2	Великобритания
850	S-1	IBM-850	GB	44	-	OS2	Великобритания
819	S-1	ISO8859-1	GB	44	en_GB	SCO	Великобритания
819	S-1	ISO8859-1	GB	44	ru	SCO	Великобритания
819	S-1	ISO8859-1	GB	44	en_GB	Bc	Великобритания
923	S-1	ISO8859-15	GB	44	en_GB.ISO8859-15	Bc	Великобритания
1252	S-1	1252	GB	44	-	WIN	Великобритания
1124	S-12	IBM-1124	UA	380	Uk_UA	AIX	Украина
1208	N-1	UTF-8	UA	380	UK_UA	AIX	Украина
1123	S-12	IBM-1123	UA	380	-	HOST	Украина
1158	S-12	IBM-1158	UA	380	-	HOST	Украина
1125	S-12	IBM-1125	UA	380	-	OS2	Украина
1251	S-12	1251	UA	380	-	WIN	Украина
819	S-1	ISO8859-1	RU	1	en_US	AIX	США
850	S-1	IBM-850	RU	1	En_US	AIX	США
923	S-1	ISO8859-15	RU	1	en_US.8859-15	AIX	США
1208	N-1	UTF-8	RU	1	EN_US	AIX	США

Таблица 23. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая страница	Группа	Кодовый набор	ID региона	Код региона	Локаль	ОС	Регион
37	S-1	IBM-37	RU	1	-	HOST	США
1140	S-1	IBM-1140	RU	1	-	HOST	США
819	S-1	iso88591	RU	1	en_US.iso88591	HP	США
923	S-1	iso885915	RU	1	-	HP	США
1051	S-1	roman8	RU	1	en_US.roman8	HP	США
819	S-1	windows-1251	RU	1	en_US	Linux	США
923	S-1	ISO-8859-15	RU	1	-	Linux	США
437	S-1	IBM-437	RU	1	-	OS2	США
850	S-1	IBM-850	RU	1	-	OS2	США
819	S-1	ISO8859-1	RU	1	en_US	SCO	США
819	S-1	ISO8859-1	RU	1	en_US	SGI	США
819	S-1	ISO8859-1	RU	1	en_US	Вс	США
923	S-1	ISO8859-15	RU	1	en_US.ISO8859-15	Вс	США
1208	N-1	UTF-8	RU	1	en_US.UTF-8	Вс	США
1252	S-1	1252	RU	1	-	WIN	США
1129	S-11	IBM-1129	VN	84	Vi_VN	AIX	Вьетнам
1208	N-1	UTF-8	VN	84	VI_VN	AIX	Вьетнам
1130	S-11	IBM-1130	VN	84	-	HOST	Вьетнам
1164	S-11	IBM-1164	VN	84	-	HOST	Вьетнам
1129	S-11	IBM-1129	VN	84	-	OS2	Вьетнам
1258	S-11	1258	VN	84	-	WIN	Вьетнам

Примечания:

1. CCSID 1392 и 5488 (GB 18030) могут применяться в утилитах загрузки и импорта только для перемещения данных из CCSID 1392 и 5488 в базу данных DB2 Unicode или для экспорта из базы данных DB2 Unicode в CCSID 1392 и 5488.
2. В AIX 4.3 и более поздних версиях применяется кодовая страница 943. В AIX 4.2 и более ранних версий, применяется кодовая страница 932.
3. Кодовая страница 1394 (Shift JIS X0213) может применяться только в утилитах загрузки и импорта для перемещения данных из кодовой страницы 1394 в базу данных DB2 Unicode или для экспорта из из базы данных DB2 Unicode в кодовую страницу 1394.
4. Арабские страны/регионы (AA):
   - Арабский (Саудовская Аравия)
   - Арабский (Ирак)
   - Арабский (Египет)
   - Арабский (Ливия)
   - Арабский (Алжир)

- Арабский (Марокко)
- Арабский (Тунис)
- Арабский (Оман)
- Арабский (Йемен)
- Арабский (Сирия)
- Арабский (Иордан)
- Арабский (Ливан)
- Арабский (Кувейт)
- Арабский (Объединенные Арабские Эмираты)
- Арабский (Бахрейн)
- Арабский (Катар)

5. Английский язык (US):
   - Английский (Ямайка)
   - Английский (Карибский бассейн)
6. Латинская Америка (Lat):
   - Испанский (Мексиканский)
   - Испанский (Гватемала)
   - Испанский (Коста-Рика)
   - Испанский (Панама)
   - Испанский (Доминиканская Республика)
   - Испанский (Венесуэла)
   - Испанский (Колумбия)
   - Испанский (Перу)
   - Испанский (Аргентина)
   - Испанский (Эквадор)
   - Испанский (Чили)
   - Испанский (Уругвай)
   - Испанский (Парагвай)
   - Испанский (Боливия)
7. Unicode поддерживает индийские рукописные шрифты следующих языков: Хинди, Гуджарати, Каннада, Конкани, Маратхи, Панджаби, Санскрит, Тамильский и Телугу.
8. Кодовая страница 950 Microsoft и кодовая страница 950 Solaris, известная также под названием Big5, не поддерживают следующие символы из IBM 950:

Диапазон кодов	Описание	Solaris Big5	Microsoft Big5	IBM Big5
X'C6A1'-X'C8FE'	Символы	Зарезервир. область	Пользоват. символы	Символы
X'F9D6'-X'F9FE'	Расширение ETen	Зарезервир. область	Системные шрифты	UDC
X'F286'-X'F9A0'	Выбранные IBM символы	Зарезервир. область	Не используется	Выбранные IBM символы

Управление поддержкой символа евро

DB2 Universal Database поддерживает символ валюты евро. Символ евро добавлен во множество кодовых страниц. Поддержка символа евро добавлена во многие внутренние таблицы преобразования кодовых страниц DB2 UDB и файлы внешних таблиц преобразования кодовых страниц, расположенные в каталоге `sqllib/conv/`.

В кодовые страницы ANSI Microsoft символ валюты евро был внесен в позицию X'80'. В кодовой странице 850 символ евро заменил символ DOTLESS I (на позиции X'D5'). По умолчанию, внутренние процедуры преобразования кодовых страниц DB2 UDB применяют эти измененные определения кодовых страниц для обеспечения поддержки символа евро.

Для применения определений таблиц преобразования кодовых страниц, не содержащих символа евро, выполните описанную ниже процедуру после установки продукта.

Предварительные требования для установки:

Перед заменой установленных файлов таблиц преобразования кодовых страниц на файлы, не содержащие символа евро, рекомендуется создать резервную копию текущих файлов.

Эти файлы находятся в каталоге `sqllib/conv/`. В операционной системе UNIX каталог `sqllib/conv/` связан с установочным каталогом DB2.

Порядок действий:

Для отключения поддержки символа евро:

1. Загрузите необходимые файлы таблиц преобразования в двоичном формате из каталога ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/. Это сервер FTP анонимного доступа, поэтому при подключении из командной строки войдите в систему как пользователь ″anonymous″ и укажите в качестве пароля

ваш адрес электронной почты. Войдя в систему, перейдите в каталог таблиц преобразования: `cd ps/products/db2/info/vr8/conv`

2. Скопируйте файлы в каталог `sqllib/conv/` в вашей системе.
3. Перезапустите DB2.

Кодовые страницы 819 и 1047:

Для кодовых 819 (ISO 8859-1 Latin 1 ASCII) и 1047 (Latin 1 Open System EBCDIC), новые кодовые страницы с символом евро, 923 (ISO 8859-15 Latin 9 ASCII) и 924 (Latin 9 Open System EBCDIC) соответственно, содержат, помимо символа евро, несколько новых символов. DB2, по умолчанию, продолжает применять прежние определения (без евро) этих двух кодовых страниц и таблицы преобразования, а именно 819 и 1047. Активизировать новые кодовые страницы 923/924 и соответствующие таблицы преобразования можно двумя способами:

- Создать новую базу данных, использующую новые кодовые страницы. Например,

  ```
  DB2 CREATE DATABASE dbname USING CODESET ISO8859-15 TERRITORY US
  ```

- Переименовать или скопировать файл таблицы преобразования 923 или 924 в каталоге `sqllib/conv/` в 819 или 1047, соответственно.

Понятия, связанные с данным:

- “Character conversion” в книге *SQL Reference, Том 1*

Ссылки, связанные с данной темой:

- “Таблицы преобразования для кодовых страниц 923 и 924” на стр. 271
- “Файл таблиц преобразования для кодовых страниц с поддержкой символа евро” на стр. 261

Файл таблиц преобразования для кодовых страниц с поддержкой символа евро

Ниже приведен список таблиц преобразования, поддерживающих символ денежной единицы евро. Для того чтобы отключить поддержку символа евро, загрузите таблицу преобразования, указанную в столбце ″Файл таблицы преобразования″.

Арабский:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
420, 16804	1046, 9238	04201046.cnv, IBM00420.ucs

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
420, 16804	1256, 5352	04201256.cnv, IBM00420.ucs
420, 16804	1200, 1208, 13488, 17584	IBM00420.ucs
864, 17248	1046, 9238	08641046.cnv, 10460864.cnv, IBM00864.ucs
864, 17248	1256, 5352	08641256.cnv, 12560864.cnv, IBM00864.ucs
864, 17248	1200, 1208, 13488, 17584	IBM00864.ucs
1046, 9238	864, 17248	10460864.cnv, 08641046.cnv, IBM01046.ucs
1046, 9238	1089	10461089.cnv, 10891046.cnv, IBM01046.ucs
1046, 9238	1256, 5352	10461256.cnv, 12561046.cnv, IBM01046.ucs
1046, 9238	1200, 1208, 13488, 17584	IBM01046.ucs
1089	1046, 9238	10891046.cnv, 10461089.cnv
1256, 5352	864, 17248	12560864.cnv, 08641256.cnv, IBM01256.ucs
1256, 5352	1046, 9238	12561046.cnv, 10461256.cnv, IBM01256.ucs
1256, 5352	1200, 1208, 13488, 17584	IBM01256.ucs

Прибалтика:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
921, 901	1257	09211257.cnv, 12570921.cnv, IBM00921.ucs
921, 901	1200, 1208, 13488, 17584	IBM00921.ucs
1112, 1156	1257, 5353	11121257.cnv
1257, 5353	921, 901	12570921.cnv, 09211257.cnv, IBM01257.ucs
1257, 5353	922, 902	12570922.cnv, 09221257.cnv, IBM01257.ucs
1257, 5353	1200, 1208, 13488, 17584	IBM01257.ucs

Беларусь:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
1131, 849	1251, 5347	11311251.cnv, 12511131.cnv
1131, 849	1283	11311283.cnv

Кириллица:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
855, 872	866, 808	08550866.cnv, 08660855.cnv
855, 872	1251, 5347	08551251.cnv, 12510855.cnv
866, 808	855, 872	08660855.cnv, 08550866.cnv
866, 808	1251, 5347	08661251.cnv, 12510866.cnv
1025, 1154	855, 872	10250855.cnv, IBM01025.ucs
1025, 1154	866, 808	10250866.cnv, IBM01025.ucs
1025, 1154	1131, 849	10251131.cnv, IBM01025.ucs
1025, 1154	1251, 5347	10251251.cnv, IBM01025.ucs
1025, 1154	1200, 1208, 13488, 17584	IBM01025.ucs
1251, 5347	855, 872	12510855.cnv, 08551251.cnv, IBM01251.ucs
1251, 5347	866, 808	12510866.cnv, 08661251.cnv, IBM01251.ucs
1251, 5347	1124	12511124.cnv, 11241251.cnv, IBM01251.ucs
1251, 5347	1125, 848	12511125.cnv, 11251251.cnv, IBM01251.ucs
1251, 5347	1131, 849	12511131.cnv, 11311251.cnv, IBM01251.ucs
1251, 5347	1200, 1208, 13488, 17584	IBM01251.ucs

Эстония:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
922, 902	1257	09221257.cnv, 12570922.cnv, IBM00922.ucs
922, 902	1200, 1208, 13488, 17584	IBM00922.ucs

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
1122, 1157	1257, 5353	11221257.cnv

Греческий:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
423	869, 9061	04230869.cnv
813, 4909	869, 9061	08130869.cnv, 08690813.cnv, IBM00813.ucs
813, 4909	1253, 5349	08131253.cnv, 12530813.cnv, IBM00813.ucs
813, 4909	1200, 1208, 13488, 17584	IBM00813.ucs
869, 9061	813, 4909	08690813.cnv, 08130869.cnv
869, 9061	1253, 5349	08691253.cnv, 12530869.cnv
875, 4971	813, 4909	08750813.cnv, IBM00875.ucs
875, 4971	1253, 5349	08751253.cnv, IBM00875.ucs
875, 4971	1200, 1208, 13488, 17584	IBM00875.ucs
1253, 5349	813, 4909	12530813.cnv, 08131253.cnv, IBM01253.ucs
1253, 5349	869, 9061	12530869.cnv, 08691253.cnv, IBM01253.ucs
1253, 5349	1200, 1208, 13488, 17584	IBM01253.ucs

Иврит:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
424, 12712	856, 9048	04240856.cnv, IBM00424.ucs
424, 12712	862, 867	04240862.cnv, IBM00424.ucs
424, 12712	916	04240916.cnv, IBM00424.ucs
424, 12712	1255, 5351	04241255.cnv, IBM00424.ucs
424, 12712	1200, 1208, 13488, 17584	IBM00424.ucs
856, 9048	862, 867	08560862.cnv, 08620856.cnv, IBM0856.ucs
856, 9048	916	08560916.cnv, 09160856.cnv, IBM0856.ucs

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
856, 9048	1255, 5351	08561255.cnv, 12550856.cnv, IBM0856.ucs
856, 9048	1200, 1208, 13488, 17584	IBM0856.ucs
862, 867	856, 9048	08620856.cnv, 08560862.cnv, IBM00862.ucs
862, 867	916	08620916.cnv, 09160862.cnv, IBM00862.ucs
862, 867	1255, 5351	08621255.cnv, 12550862.cnv, IBM00862.ucs
862, 867	1200, 1208, 13488, 17584	IBM00862.ucs
916	856, 9048	09160856.cnv, 08560916.cnv
916	862, 867	09160862.cnv, 08620916.cnv
1255, 5351	856, 9048	12550856.cnv, 08561255.cnv, IBM01255.ucs
1255, 5351	862, 867	12550862.cnv, 08621255.cnv, IBM01255.ucs
1255, 5351	1200, 1208, 13488, 17584	IBM01255.ucs

Японский:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
290, 8482	850, 858	02900850.cnv
954	1200, 1208, 13488, 17584	0954ucs2.cnv, ucs20954.cnv
1027, 5123	850, 858	10270850.cnv

Latin-1:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
37, 1140	437	00370437.cnv, IBM00037.ucs
37, 1140	850, 858	00370850.cnv, IBM00037.ucs
37, 1140	860	00370860.cnv, IBM00037.ucs
37, 1140	1051	00371051.cnv, IBM00037.ucs
37, 1140	1252, 5348	00371252.cnv, IBM00037.ucs
37, 1140	1275	00371275.cnv, IBM00037.ucs

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
37, 1140	1200, 1208, 13488, 17584	IBM00037.ucs
273, 1141	437	02730437.cnv, IBM00273.ucs
273, 1141	850, 858	02730850.cnv, IBM00273.ucs
273, 1141	1051	02731051.cnv, IBM00273.ucs
273, 1141	1252, 5348	02731252.cnv, IBM00273.ucs
273, 1141	1275	02731275.cnv, IBM00273.ucs
273, 1141	1200, 1208, 13488, 17584	IBM00273.ucs
277, 1142	437	02770437.cnv, IBM00277.ucs
277, 1142	850, 858	02770850.cnv, IBM00277.ucs
277, 1142	1051	02771051.cnv, IBM00277.ucs
277, 1142	1252, 5348	02771252.cnv, IBM00277.ucs
277, 1142	1275	02771275.cnv, IBM00277.ucs
277, 1142	1200, 1208, 13488, 17584	IBM00277.ucs
278, 1143	437	02780437.cnv, IBM00278.ucs
278, 1143	850, 858	02780850.cnv, IBM00278.ucs
278, 1143	1051	02781051.cnv, IBM00278.ucs
278, 1143	1252, 5348	02781252.cnv, IBM00278.ucs
278, 1143	1275	02781275.cnv, IBM00278.ucs
278, 1143	1200, 1208, 13488, 17584	IBM00278.ucs
280, 1144	437	02800437.cnv, IBM00280.ucs
280, 1144	850, 858	02800850.cnv, IBM00280.ucs
280, 1144	1051	02801051.cnv, IBM00280.ucs
280, 1144	1252, 5348	02801252.cnv, IBM00280.ucs
280, 1144	1275	02801275.cnv, IBM00280.ucs
280, 1144	1200, 1208, 13488, 17584	IBM00280.ucs
284, 1145	437	02840437.cnv, IBM00284.ucs
284, 1145	850, 858	02840850.cnv, IBM00284.ucs
284, 1145	1051	02841051.cnv, IBM00284.ucs
284, 1145	1252, 5348	02841252.cnv, IBM00284.ucs
284, 1145	1275	02841275.cnv, IBM00284.ucs
284, 1145	1200, 1208, 13488, 17584	IBM00284.ucs
285, 1146	437	02850437.cnv, IBM00285.ucs

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
285, 1146	850, 858	02850850.cnv, IBM00285.ucs
285, 1146	1051	02851051.cnv, IBM00285.ucs
285, 1146	1252, 5348	02851252.cnv, IBM00285.ucs
285, 1146	1275	02851275.cnv, IBM00285.ucs
285, 1146	1200, 1208, 13488, 17584	IBM00285.ucs
297, 1147	437	02970437.cnv, IBM00297.ucs
297, 1147	850, 858	02970850.cnv, IBM00297.ucs
297, 1147	1051	02971051.cnv, IBM00297.ucs
297, 1147	1252, 5348	02971252.cnv, IBM00297.ucs
297, 1147	1275	02971275.cnv, IBM00297.ucs
297, 1147	1200, 1208, 13488, 17584	IBM00297.ucs
437	850, 858	04370850.cnv, 08500437.cnv
500, 1148	437	05000437.cnv, IBM00500.ucs
500, 1148	850, 858	05000850.cnv, IBM00500.ucs
500, 1148	857, 9049	05000857.cnv, IBM00500.ucs
500, 1148	920	05000920.cnv, IBM00500.ucs
500, 1148	1051	05001051.cnv, IBM00500.ucs
500, 1148	1114, 5210	05001114.cnv, IBM00500.ucs
500, 1148	1252, 5348	05001252.cnv, IBM00500.ucs
500, 1148	1254, 5350	05001254.cnv, IBM00500.ucs
500, 1148	1275	05001275.cnv, IBM00500.ucs
500, 1148	1200, 1208, 13488, 17584	IBM00500.ucs
850, 858	437	08500437.cnv, 04370850.cnv
850, 858	860	08500860.cnv, 08600850.cnv
850, 858	1114, 5210	08501114.cnv, 11140850.cnv
850, 858	1275	08501275.cnv, 12750850.cnv
860	850, 858	08600850.cnv, 08500860.cnv
871, 1149	437	08710437.cnv, IBM00871.ucs
871, 1149	850, 858	08710850.cnv, IBM00871.ucs
871, 1149	1051	08711051.cnv, IBM00871.ucs
871, 1149	1252, 5348	08711252.cnv, IBM00871.ucs
871, 1149	1275	08711275.cnv, IBM00871.ucs

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
871, 1149	1200, 1208, 13488, 17584	IBM00871.ucs
1275	850, 858	12750850.cnv, 08501275.cnv

Latin-2:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
852, 9044	1250, 5346	08521250.cnv, 12500852.cnv
870, 1153	852, 9044	08700852.cnv, IBM00870.ucs
870, 1153	1250, 5346	08701250.cnv, IBM00870.ucs
870, 1153	1200, 1208, 13488, 17584	IBM00870.ucs
1250, 5346	852, 9044	12500852.cnv, 08521250.cnv, IBM01250.ucs
1250, 5346	1200, 1208, 13488, 17584	IBM01250.ucs

Упрощенный китайский:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
837, 935, 1388	1200, 1208, 13488, 17584	1388ucs2.cnv
1386	1200, 1208, 13488, 17584	1386ucs2.cnv, ucs21386.cnv

Традиционный китайский:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
937, 835, 1371	950, 1370	09370950.cnv, 0937ucs2.cnv
937, 835, 1371	1200, 1208, 13488, 17584	0937ucs2.cnv
1114, 5210	850, 858	11140850.cnv, 08501114.cnv

Таиланд:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
838, 1160	874, 1161	08380874.cnv, IBM00838.ucs
838, 1160	1200, 1208, 13488, 17584	IBM00838.ucs

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
874, 1161	1200, 1208, 13488, 17584	IBM00874.ucs

Турецкий:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
857, 9049	1254, 5350	08571254.cnv, 12540857.cnv
1026, 1155	857, 9049	10260857.cnv, IBM01026.ucs
1026, 1155	1254, 5350	10261254.cnv, IBM01026.ucs
1026, 1155	1200, 1208, 13488, 17584	IBM01026.ucs
1254, 5350	857, 9049	12540857.cnv, 08571254.cnv, IBM01254.ucs
1254, 5350	1200, 1208, 13488, 17584	IBM01254.ucs

Украина:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
1123, 1158	1124	11231124.cnv
1123, 1158	1125, 848	11231125.cnv
1123, 1158	1251, 5347	11231251.cnv
1124	1251, 5347	11241251.cnv, 12511124.cnv
1125, 848	1251, 5347	11251251.cnv, 12511125.cnv

Unicode:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
1200, 1208, 13488, 17584	813, 4909	IBM00813.ucs
1200, 1208, 13488, 17584	862, 867	IBM00862.ucs
1200, 1208, 13488, 17584	864, 17248	IBM00864.ucs
1200, 1208, 13488, 17584	874, 1161	IBM00874.ucs

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
1200, 1208, 13488, 17584	921, 901	IBM00921.ucs
1200, 1208, 13488, 17584	922, 902	IBM00922.ucs
1200, 1208, 13488, 17584	954	ucs20954.cnv, 0954ucs2.cnv
1200, 1208, 13488, 17584	1046, 9238	IBM01046.ucs
1200, 1208, 13488, 17584	1250, 5346	IBM01250.ucs
1200, 1208, 13488, 17584	1251, 5347	IBM01251.ucs
1200, 1208, 13488, 17584	1253, 5349	IBM01253.ucs
1200, 1208, 13488, 17584	1254, 5350	IBM01254.ucs
1200, 1208, 13488, 17584	1255, 5351	IBM01255.ucs
1200, 1208, 13488, 17584	1256, 5352	IBM01256.ucs
1200, 1208, 13488, 17584	1386	ucs21386.cnv, 1386ucs2.cnv

Вьетнамский:

CCSID/CPGID сервера баз данных	CCSID/CPGID клиента базы данных	Файлы таблиц преобразования
1130, 1164	1258, 5354	11301258.cnv
1258, 5354	1129, 1163	12581129.cnv

Понятия, связанные с данным:

- “Character conversion” в книге *SQL Reference, Том 1*

Задачи, связанные с данной темой:

- “Управление поддержкой символа евро” на стр. 260

Таблицы преобразования для кодовых страниц 923 и 924

Ниже приведен список всех файлов таблиц преобразования для кодовых страниц 923 и 924. Имена этих файлов имеют вид XXXXYYYY.cnv или ibmZZZZZ.ucs, где XXXXX - номер исходной кодовой страницы, а YYYY - номер целевой кодовой страницы. Файл с именем ibmZZZZZ.ucs содержит таблицы для преобразования кодовой страницы ZZZZZ в Unicode и наоборот.

Для того чтобы активировать таблицу преобразования кодовой страницы, переименуйте или скопируйте соответствующий файл таблицы преобразования, присвоив ему новое имя, указанное во втором столбце.

Например, для того чтобы при работе клиента 8859-1/15 (Latin 1/9) с базой данных Windows 1252 поддерживался символ евро, переименуйте или скопируйте следующие файлы таблиц преобразования из каталога sqllib/conv/:

- 09231252.cnv в 08191252.cnv
- 12520923.cnv в 12520819.cnv
- ibm00923.ucs в ibm00819.ucs

Файлы таблиц преобразования 923 и 924 из каталога sqllib/conv/	**Новое имя**
00370923.cnv	00370819.cnv
02730923.cnv	02730819.cnv
02770923.cnv	02770819.cnv
02780923.cnv	02780819.cnv
02800923.cnv	02800819.cnv
02840923.cnv	02840819.cnv
02850923.cnv	02850819.cnv
02970923.cnv	02970819.cnv
04370923.cnv	04370819.cnv
05000923.cnv	05000819.cnv
08500923.cnv	08500819.cnv
08600923.cnv	08600819.cnv
08630923.cnv	08630819.cnv
08710923.cnv	08710819.cnv
09230437.cnv	08190437.cnv
09230500.cnv	08190500.cnv
09230850.cnv	08190850.cnv
09230860.cnv	08190860.cnv

Файлы таблиц преобразования 923 и 924 из каталога sqllib/conv/	Новое имя
09230863.cnv	08190863.cnv
09231043.cnv	08191043.cnv
09231051.cnv	08191051.cnv
09231114.cnv	08191114.cnv
09231252.cnv	08191252.cnv
09231275.cnv	08191275.cnv
09241252.cnv	10471252.cnv
10430923.cnv	10430819.cnv
10510923.cnv	10510819.cnv
11140923.cnv	11140819.cnv
12520923.cnv	12520819.cnv
12750923.cnv	12750819.cnv
ibm00923.ucs	ibm00819.ucs

Понятия, связанные с данным:

- “Character conversion” в книге *SQL Reference, Том 1*

Задачи, связанные с данной темой:

- “Управление поддержкой символа евро” на стр. 260

Выбор языка для базы данных

Создавая базу данных, необходимо выбрать язык для хранения данных. При создании базы данных можно указать регион и кодовый набор. Выбранные регион и кодовый набор могут отличаться от текущих параметров операционной системы. Если не указать явно регион и кодовый набор при создании базы данных, то база данных будет создана с использованием текущей локали. Выбирая кодовый набор убедитесь, что он позволяет зашифровать все символы языка, который будет применяться.

Вместо этого, можно хранить данные в базе данных Unicode, что позволяет не указывать конкретный язык; Кодировка Unicode включает символы большинства существующих языков мира.

Национальные настройки для сервера администратора DB2

Убедитесь, что национальная версия экземпляра сервера администратора DB2 совместима с национальной версией экземпляра DB2. В противном случае экземпляр DB2 не сможет связываться с сервером администратора DB2.

Если в профиле пользователя сервера администратора DB2 не задана переменная среды LANG, сервер администратора DB2 будет запущен с системной национальной версией по умолчанию. Если системная национальная версия по умолчанию не определена, сервер администратора будет запущен с кодовой страницей 819. Если экземпляр DB2 использует одну из национальных версий DBCS, а сервер администратора DB2 запущен с кодовой страницей 819, этот экземпляр не сможет связываться с сервером администратора DB2. Национальная версия экземпляра сервера администратора DB2 и национальная версия экземпляра DB2 должны быть совместимы.

Например, в системе Linux с упрощенным китайским в профиле пользователя сервера администратора DB2 должно быть задано `LANG=zh_CN`.

Включение поддержки двунаправленного письма

Размещение текста с двумя направлениями письма реализуется в DB2 Universal Database при помощи новых определений идентификаторов наборов кодовых символов (CCSID). Для этих новых CCSID с двумя направлениями письма вместо преобразований кодовых страниц или дополнительно к ним производится преобразование размещения. Чтобы использовать эту поддержку, для переменной регистра DB2BIDI должно быть задано значение YES. По умолчанию эта переменная не задается. Она используется сервером для всех преобразований и может быть установлена только при запуске сервера. Задание значения YES для DB2BIDI может оказать влияние на производительность из-за дополнительной проверки и преобразований размещения.

Ограничения:

Применяются следующие ограничения:

- Если выбрать CCSID, который не подходит для кодовой страницы или типа строки вашей платформы клиента, можно получить непредсказуемые результаты. Если выбран несовместимый CCSID (например, CCSID иврита для связи с базой данных на арабском языке) или если DB2BIDI не был установлен для этого сервера, при попытке соединения вы получите сообщение об ошибке.
- Переопределение CCSID не поддерживается для тех случаев, когда платформа хоста с кодом EBCDIC является клиентом, а DB2 UDB - сервером.

Порядок действий:

Для того чтобы указать конкретный CCSID в среде, отличной от DRDA:

- Присвойте переменной реестра DB2BIDI значение YES.
- Выберите CCSID, соответствующий свойствам клиента и присвойте это значение переменной DB2CODEPAGE.

- Если у вас уже установлено соединение с базой данных, необходимо ввести команду TERMINATE, а затем снова установить соединение, чтобы новое значение DB2CODEPAGE вступило в силу.

В средах DRDA, если платформа хоста с кодом EBCDIC также поддерживает двунаправленные CCSID, необходимо только задать значение DB2CODEPAGE. Однако если платформа хоста не поддерживает эти CCSID, необходимо также указать новый CCSID для сервера базы данных хоста, с которым вы связываетесь. Это необходимо потому, что в среде DRDA преобразования кодовых страниц и размещения выполняются получателем данных. Однако если сервер хоста не поддерживает эти CCSID с двумя направлениями письма, он не выполняет преобразование размещения для данных, которые получает от DB2 UDB. Если вы используете переопределение CCSID, клиент DB2 UDB выполняет также преобразование размещения для выходных данных.

Понятия, связанные с данным:

- “Поддержка двунаправленного письма для DB2 Connect” на стр. 278
- “Обработка данных с двумя направлениями письма” в книге *DB2 Connect. Руководство пользователя*

Ссылки, связанные с данной темой:

- “CCSID с двумя направлениями письма” на стр. 274
- “Общие переменные реестра” в книге *Руководство администратора: Производительность*

CCSID с двумя направлениями письма

Для правильной обработки данных с двумя направлениями письма на разных платформах требуются следующие атрибуты направления письма:

- Тип текста
- Изменение формы цифр
- Направление
- Изменение формы текста
- Симметричная замена

Поскольку на разных платформах по умолчанию устанавливаются разные значения, при переносе данных DB2 с одной платформы на другую могут происходить ошибки. Например, операционная система Windows использует данные LOGICAL UNSHAPED, а z/OS и OS/390 обычно используют данные SHAPED VISUAL. Поэтому без поддержки для двунаправленных атрибутов данные, посланные из DB2 Universal Database for OS/390 and z/OS в DB2 UDB 32-битной системы Windows, могут отображаться неправильно.

DB2 поддерживает атрибуты данных с двумя направлениями письма через специальные идентификаторы наборов кодовых символов с двумя направлениями письма (Coded Character Set Identifiers - CCSID). Для DB2 UDB определены и используются следующие CCSID с двумя направлениями письма, как показано на схеме Табл. 24. Типы строк CDRA определены, как показано на схеме Табл. 25 на стр. 277.

Таблица 24. CCSID с двумя направлениями письма

CCSID	Кодовая страница	Тип строки
420	420	4
424	424	4
856	856	5
862	862	4
864	864	5
867	862	4
916	916	5
1046	1046	5
1089	1089	5
1255	1255	5
1256	1256	5
5351	1255	5
5352	1256	5
8612	420	5
8616	424	10
9048	856	5
9238	1046	5
12712	424	4
16804	420	4
17248	864	5
62208	856	4
62209	862	10
62210	916	4
62211	424	5
62213	862	5
62215	1255	4
62218	864	4

Таблица 24. CCSID с двумя направлениями письма (продолжение)

CCSID	Кодовая страница	Тип строки
62220	856	6
62221	862	6
62222	916	6
62223	1255	6
62224	420	6
62225	864	6
62226	1046	6
62227	1089	6
62228	1256	6
62229	424	8
62230	856	8
62231	862	8
62232	916	8
62233	420	8
62234	420	9
62235	424	6
62236	856	10
62237	1255	8
62238	916	10
62239	1255	10
62240	424	11
62241	856	11
62242	862	11
62243	916	11
62244	1255	11
62245	424	10
62246	1046	8
62247	1046	9
62248	1046	4
62249	1046	12
62250	420	12

Таблица 25. Типы строк CDRA

Тип строки	Тип текста	Изменение формы цифр	Направление	Изменение формы текста	Симметричная замена
4	Видимый	Passthrough	Слева направо	Изменяется	Выключена
5	Невидимый	Арабский	Слева направо	Не изменяется	Включена
6	Невидимый	Арабский	Справа налево	Не изменяется	Включена
7*	Видимый	Passthrough	Зависит от контекста*	Лигатура без изменения	Выключена
8	Видимый	Passthrough	Справа налево	Изменяется	Выключена
9	Видимый	Passthrough	Справа налево	Изменяется	Включена
10	Невидимый	Арабский	Контекстно, слева направо	Не изменяется	Включена
11	Невидимый	Арабский	Контекстно, справа налево	Не изменяется	Включена
12	Невидимый	Арабский	Справа налево	Изменяется	Выключена

Примечание: * Ориентация поля - слева направо (LTR), если первый алфавитный символ - латинский, и справа налево (RTL), если первый алфавитный символ - не латинский. Форма символов не меняется, но лигатуры типа лам-алеф сохраняются, а не разбиваются на составляющие.

Понятия, связанные с данным:

- “Поддержка двунаправленного письма для DB2 Connect” на стр. 278

Задачи, связанные с данной темой:

- “Включение поддержки двунаправленного письма” на стр. 273

Поддержка двунаправленного письма для DB2 Connect

При обмене данными между DB2® Connect и базой данных на сервере преобразование входящих данных обычно выполняет получатель. Такое же соглашение будет обычно действовать и для преобразования размещения в дополнение к обычному преобразованию кодовых страниц. У DB2™ Connect есть дополнительная возможность выполнять преобразования размещения и для данных, подготовленных к отправке в базу данных сервера, в дополнение к данным, получаемым от базы данных сервера.

Чтобы DB2 Connect выполняла преобразования размещения для исходящих данных, отправляемых на базу данных сервера, CCSID базы сервера надо переопределить. Это достигается с помощью параметра BIDI в поле PARMS записи каталога базы данных DCS для базы данных сервера.

Примечание: Если вы хотите, чтобы DB2 Connect выполняла преобразования размещения для данных, подготовленных к отправке на базу данных хоста DB2 или iSeries™, даже в том случае, если вы не переопределили ее CCSID, вам все равно необходимо добавить параметр BIDI в поле PARMS каталога базы данных DCS. В этом случае надо задать CCSID, который используется по умолчанию базой данных хоста DB2 или iSeries.

Параметр BIDI должен быть указан девятым параметром в поле PARMS вместе с CCSID, на который вы хотите заменить устанавливаемый по умолчанию CCSID базы данных сервера:

```
",,,,,,,,BIDI=xyz"
```

где *xyz* - новый CCSID.

Примечание: Чтобы параметр BIDI вступил в действие, для переменной регистра DB2BIDI должно быть задано значение YES.

Использование этой функции лучше всего описать на примере.

Предположим, что у вас есть клиент DB2 с ивритом, на котором работает CCSID 62213 (тип строки 5), и вы хотите обратиться к базе данных хоста DB2 или iSeries, на которой запущен CCSID 00424 (тип строки 4). Однако вы знаете, что для данных, содержащихся в базе данных хоста DB2 или iSeries, используется CCSID 08616 (тип строки 6).

Здесь мы сталкиваемся с двумя проблемами: Во-первых, база данных хоста DB2 или iSeries не знает, чем различаются типы двунаправленных строк с CCSID 00424 и 08616. Во-вторых, база данных хоста DB2 или iSeries не распознает

CCSID клиента DB2 (62213). Она поддерживает только CCSID 00862, основанный на той же кодовой странице, что и CCSID 62213.

Вам надо добиться, чтобы данные, посланные на базу данных хоста DB2 или iSeries, начинались с формата строки типа 6, а также сообщить DB2 Connect, что она должна выполнять преобразование размещения для данных, которые она получает от базы данных хоста DB2 или iSeries. Надо использовать следующую команду catalog для базы данных хоста DB2 или iSeries:

```
db2 catalog dcs database nydb1 as telaviv parms ",,,,,,,,BIDI=08616"
```

Эта команда заставляет DB2 Connect использовать CCSID базы данных хоста DB2 или iSeries 08616 вместо 00424. Такая замена включает в себя следующие действия:

1. DB2 Connect связывается с базой данных хоста DB2 или iSeries с использованием CCSID 00862.
2. DB2 Connect выполняет преобразование размещения для данных, которые она собирается *послать* базе данных хоста DB2 или iSeries. Производится преобразование из CCSID 62213 (тип строки 5) в CCSID 62221 (тип строки 6).
3. DB2 Connect выполняет преобразование размещения для данных, которые она *получает* от базы данных хоста DB2 или iSeries. Эта трансформация производится из CCSID 08616 (тип строки 6) в CCSID 62213 (тип строки 5).

Примечание: В некоторых случаях использование CCSID с двумя направлениями письма может привести к тому, что сам запрос SQL будет модифицирован и сервер DB2 его не распознает. Чтобы этого не случилось, следует исключить использование CCSID с IMPLICIT CONTEXTUAL и IMPLICIT RIGHT-TO-LEFT, когда могут быть использованы разные типы строк. CCSID с CONTEXTUAL может давать непредсказуемые результаты, если в запросе SQL содержатся строки в двойных кавычках. Избегайте использования строк в двойных кавычках в операторах SQL; где это возможно, используйте переменные хоста.

Если использование конкретного CCSID с двумя направлениями письма вызывает ошибки, которые не удается устранить посредством приведенных рекомендаций, установите NO для DB2BIDI.

Понятия, связанные с данным:

- “Обработка данных с двумя направлениями письма” в книге *DB2 Connect. Руководство пользователя*

Ссылки, связанные с данной темой:

- “CCSID с двумя направлениями письма” на стр. 274

Последовательность сортировки

Менеджер баз данных сравнивает символьные данные с помощью *последовательности сортировки*. Это упорядочение для набора символов, которое определяет, будет ли при сортировке конкретный символ расположен до другого, после другого или там же, где другой.

Примечание: Данные строк символов, определенные с атрибутом FOR BIT DATA, и данные двоичных больших объектов сортируются при помощи двоичной последовательности сортировки.

Например, последовательность сортировки может быть использована для того, чтобы указать, что прописные и строчные версии конкретного символа должны сортироваться одинаково.

Менеджер баз данных позволяет создавать базы данных с произвольной последовательностью сортировки. Приведенная ниже информация поможет вам создать последовательность сортировки для базы данных.

Любой однобайтовый символ представляется внутри базы данных как уникальное число от 0 до 255 (в шестнадцатеричном формате - значение от X'00' до X'FF'). Это число называется *кодом* символа. Таблица, определяющая соответствие между кодами и символами, называется *кодовой страницей*. Последовательность сортировки определяет взаимосвязь между кодом символа и его положением в отсортированной последовательности. Номер позиции символа называется *весом* символа в последовательности сортировки. В простейшей последовательности сортировки вес совпадает с кодом символа. Такая последовательность сортировки называется *тождественной (identity) последовательностью*.

Например, предположим, что коды символов B и b равны X'42' и X'62', соответственно. Если в соответствии с таблицей последовательности сортировки вес обоих символов равен X'42' (B), то с точки зрения сортировки эти символы равнозначны. Если вес символа B равен X'9E', а вес символа b равен X'9D', то в отсортированной последовательности символ b будет расположен раньше, чем B. Веса символов задаются в таблице последовательности сортировки. Не следует путать эту таблицу с кодовой страницей, которая задает коды символов.

Рассмотрим следующий пример. В кодировке ASCII буквам от A до Z присвоены коды от X'41' до X'5A'. Для того чтобы описать последовательность сортировки, в которой эти буквы располагаются последовательно (без пропусков), можно указать X'41', X'42', ... X'59', X'5A'.

Для многобайтового символа в качестве веса используется его шестнадцатеричное представление. Например, предположим, что

многобайтовым символам A и B соответствуют коды X'8260' и X'8261', тогда для сортировки этих символов в соответствии с их кодами применяются веса X'82', X'60' и X'61'.

Не требуется, чтобы веса в последовательности сортировки были уникальны. Например, прописным и соответствующим строчным символам можно назначить одинаковый вес.

Последовательность сортировки значительно проще указать в том случае, если она задает веса для всех 256 кодов. В этом случае вес символа можно определить по его коду.

DB2® всегда применяет ту таблицу сортировки, которая была задана при создании базы данных. Для того чтобы многобайтовые символы сортировались в последовательности их перечисления в таблице кодов, укажите при создании таблицы последовательность сортировки IDENTITY.

Примечание: Для двухбайтовых символов и символов Unicode, содержащихся в полях GRAPHIC, последовательность сортировки всегда равна IDENTITY.

После определения последовательности сортировки все последующие сравнения символов для этой базы данных будут выполняться с ее помощью. За исключением символьных данных, определенных как FOR BIT DATA или данные типа двоичный большой объект, последовательность сортировки будет использоваться для всех сравнений SQL и условий ORDER BY, а также для построения индексов и статистики.

Ошибки могут возникать в следующих случаях:

- Прикладная программа производит слияние сортированных данных из базы данных с данными прикладной программы, сортированными с использованием другой последовательности сортировки.
- Прикладная программа производит слияние сортированных данных из одной базы данных с сортированными данными другой базы данных, но эти базы данных имеют разные последовательности сортировки.
- Прикладная программа делает предположения относительно сортированных данных, которые не верны для определенной последовательности сортировки. Например, предположение о расположении цифр после букв может быть верным для конкретной последовательности сортировки, а может и не быть.

Наконец, следует помнить, что результаты любой сортировки, основанной на прямом сравнении значений кода символов, будут совпадать только с результатами запроса, упорядоченными с использованием идентичной последовательности сортировки.

Понятия, связанные с данным:

- “Character conversion” в книге *SQL Reference, Том 1*
- “Character Comparisons Based on Collating Sequences” в книге *Application Development Guide: Programming Client Applications*

Сортировка тайских символов

В тайском языке есть специальные гласные (″начальные гласные″), тональные знаки и прочие специальные символы, которые не сортируются последовательно.

Ограничения:

Необходимо создать базу данных с тайским кодовым набором и локалью.

Порядок действий:

При создании базы данных укажите в команде CREATE DATABASE условие COLLATE USING NLSCHAR.

Понятия, связанные с данным:

- “Последовательность сортировки” на стр. 280

Ссылки, связанные с данной темой:

- “CREATE DATABASE Command” в книге *Command Reference*

Форматы даты и времени по коду региона

Формат символьной строки, представляющей дату и время - это формат по умолчанию для значений даты и времени, связанный с кодом региона для программы. Этот формат по умолчанию может быть переопределен, если при прекомпиляции программы или ее связывании с базой данных задана опция формата DATETIME.

Ниже приводится описание входных и выходных форматов для даты и времени:

- Входной формат времени
  - Для времени не существует входного формата по умолчанию
  - Любой формат допустим на вводе для любого кода региона.
- Выходной формат времени
  - Выходной формат времени по умолчанию совпадает с местным форматом времени.
- Входной формат даты

- Для даты не существует входного формата по умолчанию
- Там, где локальный формат даты несовместим с форматами дат ISO, JIS, EUR или USA, при вводе распознается местный формат даты. (Например, посмотрите запись в Табл. 26 для Великобритании.)

• Выходной формат даты
- Выходные форматы дат по умолчанию показаны в Табл. 26.

Примечание: В Табл. 26 приводится также список форматов строк для различных кодов регионов.

Таблица 26. Форматы даты и времени для кодов регионов

Код региона	Местный формат даты	Местный формат времени	Выходной формат даты по умолчанию	Входные форматы даты
355 Албания	гггг-мм-дд	JIS	LOC	LOC, USA, EUR, ISO
785 Арабский	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
001 Австралия (1)	мм-дд-гггг	JIS	LOC	LOC, USA, EUR, ISO
061 Австралия	дд-мм-гггг	JIS	LOC	LOC, USA, EUR, ISO
032 Бельгия	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
055 Бразилия	дд.мм.гггг	JIS	LOC	LOC, EUR, ISO
359 Болгария	дд.мм.гггг	JIS	EUR	LOC, USA, EUR, ISO
001 Канада	мм-дд-гггг	JIS	USA	LOC, USA, EUR, ISO
002 Канада (французский)	дд-мм-гггг	ISO	ISO	LOC, USA, EUR, ISO
385 Хорватия	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
042 Чехия	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
045 Дания	дд-мм-гггг	ISO	ISO	LOC, USA, EUR, ISO
358 Финляндия	дд/мм/гггг	ISO	EUR	LOC, EUR, ISO
389 Македония	дд.мм.гггг	JIS	EUR	LOC, USA, EUR, ISO
033 Франция	дд/мм/гггг	JIS	EUR	LOC, EUR, ISO
049 Германия	дд/мм/гггг	ISO	ISO	LOC, EUR, ISO

Таблица 26. Форматы даты и времени для кодов регионов (продолжение)

Код региона	Местный формат даты	Местный формат времени	Выходной формат даты по умолчанию	Входные форматы даты
030 Греция	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
036 Венгрия	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
354 Исландия	дд-мм-гггг	JIS	LOC	LOC, USA, EUR, ISO
091 Индия	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
972 Израиль	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
039 Италия	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
081 Япония	мм/дд/гггг	JIS	ISO	LOC, USA, EUR, ISO
082 Корея	мм/дд/гггг	JIS	ISO	LOC, USA, EUR, ISO
001 Латинская Америка (1)	мм-дд-гггг	JIS	LOC	LOC, USA, EUR, ISO
003 Латинская Америка	дд-мм-гггг	JIS	LOC	LOC, EUR, ISO
031 Нидерланды	дд-мм-гггг	JIS	LOC	LOC, USA, EUR, ISO
047 Норвегия	дд/мм/гггг	ISO	EUR	LOC, EUR, ISO
048 Польша	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
351 Португалия	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
086 КНР	мм/дд/гггг	JIS	ISO	LOC, USA, EUR, ISO
040 Румыния	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
007 Россия	дд/мм/гггг	ISO	LOC	LOC, EUR, ISO
381 Югославия	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
042 Словакия	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
386 Словения	гггг-мм-дд	JIS	ISO	LOC, USA, EUR, ISO
034 Испания	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
046 Швеция	дд/мм/гггг	ISO	ISO	LOC, EUR, ISO

Таблица 26. Форматы даты и времени для кодов регионов (продолжение)

Код региона	Местный формат даты	Местный формат времени	Выходной формат даты по умолчанию	Входные форматы даты
041 Швейцария	дд/мм/гггг	ISO	EUR	LOC, EUR, ISO
088 Тайвань	мм-дд-гггг	JIS	ISO	LOC, USA, EUR, ISO
066 Таиланд (2)	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
090 Турция	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
044 Великобритания	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO
001 США	мм-дд-гггг	JIS	USA	LOC, USA, EUR, ISO
084 Вьетнам	дд/мм/гггг	JIS	LOC	LOC, EUR, ISO

Примечания:

1. Для стран/регионов, использующих национальную версию C по умолчанию, используется код региона 001.
2. гггг в буддийском летоисчислении эквивалентно григорианскому + 543 года (только Таиланд).

Ссылки, связанные с данной темой:

- “BIND Command” в книге *Command Reference*
- “PRECOMPILE Command” в книге *Command Reference*

Кодировка символов Unicode

Стандарт кодировки символов Unicode - схема кодировки символов с фиксированной длиной, включающая символы практически изо всех живых языков мира.

Дополнительную информацию о Unicode можно найти в последнем издании книги The Unicode Standard и на Web-сайте The Unicode Consortium (www.unicode.org).

Unicode использует два варианта кодирования: 8-битный и 16-битный. По умолчанию используется 16-битное кодирование, то есть каждый символ занимает 16 бит (два байта); обычно его записывают как U+hhhh, где hhhh - шестнадцатеричный код символа. Несмотря на то, что получающихся более 65000 элементов кода достаточно для кодирования большинства символов основных мировых языков, стандарт Unicode также предоставляет механизм

расширения, позволяющий кодирование еще около одного миллиона символов. Механизм расширения использует для кодирования одного символа расширения пару из верхнего и нижнего замещающих символов. Значение кода первого (или верхнего) замещающего символа находится между U+D800 и U+DBFF, а второго (или нижнего) - между U+DC00 и U+DFFF.

UCS-2

Стандарт Международной организации по стандартизации (ISO) и Международной электротехнической комиссии (IEC) 10646 (ISO/IEC 10646) описывает Универсальный многооктетный набор кодовых символов (Universal Multiple-Octet Coded Character Set, UCS) с 16-битной (двухбайтной) версией (UCS-2) и 32-битной (четырехбайтной) версией (UCS-4). UCS-2 идентичен 16-битному варианту Unicode без замещающих символов. UCS-2 позволяет кодировать все (16-битные) символы, определенные в Unicode версии 3.0. Два символа UCS-2 — верхний и нижний замещающий — кодируют каждый дополнительный символ, введенный, начиная в Unicode версии 3.1. Эти дополнительные символы определены вне исходной 16-битной Basic Multilingual Plane (BMP или Plane 0).

UTF-8

Шестнадцатибитные символы Unicode ставят серьезную проблему для программ и файловых систем на основе ASCII, ориентированных на работу с байтами. Например, программа, не рассчитанная на работу с Unicode, может ошибочно интерпретировать 8 ведущих нулевых битов символа 'A' (U+0041) как однобайтный символ ASCII NULL.

UTF-8 (формат преобразования UCS 8, UCS Transformation Format 8) - алгоритмическое преобразование, переводящее символы Unicode фиксированной длины в байтовые строки переменной длины из обычных (не управляющих) символов ASCII. В UTF-8 символы ASCII и управляющие символы представлены своими обычными однобайтными кодами, а остальные символы UCS-2 получают длину два или больше байт. UTF-8 позволяет как дополнительные, так и недополнительные символы.

UTF-16

ISO/IEC 10646 определяет также технику расширения для кодирования некоторых символов UCS-4 двумя символами UCS-2. Это расширение, называющееся UTF-16, идентично 16-битному Unicode с замещающими символами. Суммарно набор символов UTF-16 содержит все символы UCS-2 плюс еще миллион символов, доступных через замещающие пары.

При разделении 16-битных символов Unicode на байты некоторые процессоры помещают старший байт первым, а некоторые - младший байт первым. По умолчанию для Unicode старший бит ставится первым.

Число байтов для каждого символа UTF-16 в формате UTF-8 можно определить из Табл. 27 на стр. 287.

Таблица 27. Положение битов UTF-8

Код (двоичный)	**UTF-16** (двоичный)	**1-й байт** (двоичный)	**2-й байт** (двоичный)	**3-й байт** (двоичный)	**4-й байт** (двоичный)
00000000 0xxxxxxx	00000000 0xxxxxxx	0xxxxxxx			
00000yyy yyxxxxxx	00000yyy yyxxxxxx	110yyyyy	10xxxxxx		
zzzzyyyy yyxxxxxx	zzzzyyyy yyxxxxxx	1110zzzz	10yyyyyy	10xxxxxx	
uuuuu zzzzyyyy yyxxxxxx	110110ww wwzzzzyy 110111yy yyxxxxxx	11110uuu (где uuuuu = wwww+1)	10uuzzzz	10yyyyyy	10xxxxxx

Здесь последовательности из u, w, x, y и z - битовое представление символа. Например, U+0080 преобразуется в 11000010 10000000 (двоичное), а замещающая пара символов U+D800 U+DC00 становится 11110000 10010000 10000000 10000000 (двоичное).

Понятия, связанные с данным:

- “Реализация Unicode в DB2” на стр. 287
- “Обработка типов данных Unicode” на стр. 290
- “Литералы Unicode” на стр. 292

Задачи, связанные с данной темой:

- “Создание базы данных Unicode” на стр. 291

Реализация Unicode в DB2

DB2® UDB поддерживает UTF-8 и UCS-2, то есть Unicode без замещающих символов.

Когда создается база данных Unicode, данные CHAR, VARCHAR, LONG VARCHAR и символьных больших объектов сохраняются в UTF-8, а данные GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC и двухбайтных символьных больших объектов - в UCS-2.

В версиях DB2 UDB до Версии 7.2 FixPak 4 DB2 UDB рассматривала два символа замещающей пары как два отдельных символа Unicode. Поэтому преобразование такой пары из UTF-16/UCS-2 в UTF-8 приводило к двум трехбайтным последовательностям. Начиная с DB2 UDB Версии 7.2 FixPak 4, DB2 UDB распознает замещающие пары при преобразовании между UTF-16/UCS-2 и UTF-8, то есть замещающая пара UTF-16 становится одной четырехбайтной последовательностью UTF-8. В остальных случаях, DB2 рассматривает пару замещающих символов как два отдельных символа UCS-2. В базах данных DB2 Unicode можно хранить дополнительные символы, при наличии способа их идентификации от других символов.

DB2 UDB рассматривает каждый символ Unicode, в том числе не занимающий отдельного места (такой как COMBINING ACUTE ACCENT - U+0301), как отдельный символ. Таким образом, DB2 UDB не распознает, что символ LATIN SMALL LETTER A WITH ACUTE (U+00E1) эквивалентен символу LATIN SMALL LETTER A (U+0061), за которым идет символ COMBINING ACUTE ACCENT (U+0301).

По умолчанию для базы данных UCS-2 Unicode используется последовательность сортировки IDENTITY, то есть символы упорядочиваются по их кодам. Поэтому по умолчанию все символы Unicode упорядочиваются и сравниваются в соответствии с последовательностью их кодов. Для основных символов Unicode их порядок слияния при кодировке UTF-8 и UCS-2 совпадает. Однако если вы используете дополнительный символ, для которого требуется пара замещающих символов, при упорядочивании в UTF-8 он попадет в конец, а в UCS-2 будет находиться где-то в середине, причем замещающие символы будут рассматриваться как отдельные. Причина в том, что при кодировании символа расширения в UTF-8 получится четырехбайтная последовательность 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx, которая больше, чем кодировка в UTF-8 символа U+FFFF, а именно, namely X'EFBFBF'. Но в UCS-2 тот же дополнительный символ кодируется как пара замещающих символов UCS-2 и будет представлен в двоичном формате как 1101 1000 xxxx xxxx 1101 1100 xxxx xxxx, что меньше, чем кодировка UCS-2 для U+FFFF.

Базу данных Unicode можно также создать с опцией сортировки IDENTITY_16BIT. Опция IDENTITY_16BIT differs отличается от опции IDENTITY по умолчанию тем, что для сортировки данных типов CHAR, VARCHAR, LONG VARCHAR и CLOB в базе данных Unicode вместо двоичной последовательности UTF-8 применяется двоичная последовательность CESU-8. CESU-8 означает *Compatibility Encoding Scheme for UTF-16: 8-Bit* (совместимая схема кодирования для UTF-16, 8-битная); ее спецификация включена в Draft Unicode Technical Report #26, доступный на сайте Unicode Technical Consortium (www.unicode.org). CESU-8 двоично-идентична UTF-8, за исключением дополнительных символов Unicode, то есть символов, определенных вне 16-битной Basic Multilingual Plane (BMP или Plane 0). В кодировке UTF-8 дополнительный символ представляется одной 4-байтной последовательностью,

в CESU-8 для того же символа требуется две 3-байтные последовательности. Применение опции сортировки IDENTITY_16BIT позволяет обеспечить одинаковую последовательность сортировки для символьных и графических данных. Даже при наличии опции IDENTITY_16BIT, данные типов CHAR, VARCHAR, LONG VARCHAR и CLOB продолжают храниться в формате UTF-8.

Все зависящие от территории параметры, такие как формат даты и времени, разделитель десятичных разрядов и прочие, определяются по текущему значению территории клиента.

База данных Unicode позволяет выполнять соединения из любых кодовых страниц, поддерживаемых DB2 UDB. Преобразования символов кодовой страницы между кодовой страницей клиента и UTF-8 автоматически выполняются менеджером баз данных. Данные с типами графических строк всегда находятся в UCS-2 и не подвергаются преобразованиям кодовых страниц. Исключением является среда процессора командной строки (CLP). Если в CLP выбрать данные графической строки (UCS-2), возвращенные данные графической строки преобразуются (посредством CLP) из UCS-2 в кодовую страницу среды вашего клиента.

Все клиенты ограничены наборами символов, методами ввода и шрифтами, поддерживаемыми в их средах, однако сама база данных UCS-2 принимает и сохраняет все символы UCS-2. Следовательно, каждый клиент обычно работает с подмножеством символов UCS-2, но менеджер баз данных позволяет использовать все символы UCS-2.

При преобразовании символов из локальной кодовой страницы в UTF-8 может произойти увеличение числа байтов. Для символов ASCII такого увеличения не происходит, но другие символы UCS-2 увеличиваются в два или три раза.

Номера кодовых страниц/CCSID

В программных продуктах фирмы IBM кодовая страница UCS-2 зарегистрирована как кодовая страница 1200 с расширяемым набором символов, то есть когда к кодовой странице добавляются новые символы, номер этой кодовой страницы не изменяется. Кодовая страница 1200 всегда относится к текущей версии Unicode.

Конкретная версия стандарта UCS, как она описана Unicode 2.0 и ISO/IEC 10646-1, также зарегистрирована в IBM® как CCSID 13488. Этот CCSID используется внутри DB2 UDB для хранения данных графических строк в базах данных euc-Japan и euc-Taiwan. И CCSID 13488, и кодовая страница 1200 относятся к UCS-2 и обрабатываются одинаково, за исключением значения двухбайтного (DBCS) пробела:

CP/CCSID	Однобайтный (SBCS) пробел	Двухбайтный (DBCS) пробел
1200	Отсутствует	U+0020

```
13488              Отсутствует                    U+3000

ПРИМЕЧАНИЕ: в базе данных UCS-2 у U+3000 нет специального значения.
```

Что касается таблиц преобразования, поскольку кодовая страница 1200 является надмножеством CCSID 13488, для обоих используются одни и те же таблицы (надмножества).

В продуктах фирмы IBM UTF-8 зарегистрирован как CCSID 1208 с расширяемым набором символов (иногда также называемым кодовой страницей 1208). При добавлении к стандарту новых символов этот номер (1208) не изменяется.

Номер кодовой страницы MBCS равен 1208, что является номером кодовой страницы базы данных и кодовой страницей данных символьных строк в базе данных. Номер двухбайтной кодовой страницы для UCS-2 - 1200, что является кодовой страницей данных графических строк внутри базы данных.

Понятия, связанные с данным:

- “Кодировка символов Unicode” на стр. 285
- “Обработка типов данных Unicode” на стр. 290
- “Литералы Unicode” на стр. 292

Задачи, связанные с данной темой:

- “Создание базы данных Unicode” на стр. 291

Обработка типов данных Unicode

Все типы данных, поддерживаемые DB2® UDB, поддерживаются и в базе данных UCS-2. В частности, данные графической строки поддерживаются для базы данных UCS-2 и хранятся в UCS-2/Unicode. Все клиенты, включая клиенты SBCS, могут работать с типами данных графических строк в UCS-2/Unicode, когда они связаны с базой данных UCS-2.

База данных UCS-2 подобна любой базе данных MBCS, в которой строчные данные измеряются в числе байтов. При работе с данными символьных строк в UTF-8 не следует полагать, что каждый символ равен одному байту. При многобайтном кодировании UTF-8 каждый символ ASCII занимает один байт, а остальные символы - два или три байта. Это следует учитывать при определении полей CHAR. В зависимости от соотношения символов ASCII и не-ASCII поле CHAR размером *n* байтов может содержать от *n*/4 до *n* символов.

Использование кодирования UTF-8 символьной строки по сравнению с типом данных UCS-2 графической строки также оказывает влияние на общие требования к хранению. Когда большая часть символов является символами

ASCII, а между ними расположено несколько прочих символов, лучшей альтернативой может быть сохранение данных в UTF-8, поскольку требования к хранению близки к одному байту на символ. С другой стороны, в ситуации, когда большинство символов не являются символами ASCII и расширяются до трехбайтных или четырехбайтных последовательностей UTF-8, лучшей альтернативой может быть формат графической строки UCS-2, поскольку каждому символу UCS-2 требуется точно два байта, а не три, как для соответствующего символа в формате UTF-8.

В средах MBCS функции SQL, работающие с символьными строками, такие как LENGTH, SUBSTR, POSSTR, MAX, MIN и подобные, оперируют с числом ″байтов″, а не с числом ″символов″. В базе данных UCS-2 поведение такое же, но необходимо соблюдать дополнительные предосторожности при указании сдвигов и длин для базы данных UCS-2, поскольку эти значения всегда определяются в контексте кодовой страницы базы данных. То есть в случае базы данных UCS-2 эти сдвиги должны быть определены в UTF-8. Поскольку для некоторых однобайтных символов в UTF-8 требуется больше одного байта, индексы SUBSTR, действительные для однобайтной базы данных, могут не быть действительными для базы данных UCS-2. Если указаны неправильные индексы, будет возвращено SQLCODE -191 (SQLSTATE 22504).

Типы данных SQL CHAR поддерживаются (в языке C) в пользовательских программах, как тип данных `char`. Типы данных SQL GRAPHIC в пользовательских программах поддерживаются, как тип `sqldbchar`. Обратите внимание на то, что для базы данных UCS-2 данные `sqldbchar` всегда находятся в формате с прямым порядком байтов (первым идет старший байт). Когда прикладная программа связана с базой данных UCS-2, данные символьной строки преобразуются DB2 UDB между кодовой страницей этой программы и UTF-8, но данные графической строки всегда находятся в UCS-2.

Понятия, связанные с данным:

- “Кодировка символов Unicode” на стр. 285
- “Реализация Unicode в DB2” на стр. 287

Создание базы данных Unicode

Порядок действий:

По умолчанию базы данных создаются в кодовой странице создавшей их прикладной программы. Поэтому, если вы создаете базу данных из клиента Unicode (UTF-8) (например, из локали UNIVERSAL в AIX или при значении переменной реестра DB2CODEPAGE, равном 1208), база данных будет создана как база данных Unicode. В качестве альтернативы можно явным образом указать ″UTF-8″ для CODESET и использовать любые действительные буквы кода TERRITORY, поддерживаемого DB2 UDB.

Для создания базы данных Unicode с кодом территории для США, введите:

```
DB2 CREATE DATABASE dbname USING CODESET UTF-8 TERRITORY US
```

Чтобы создать базу данных Unicode с использованием API **sqlecrea**, необходимо соответствующим образом установить значения в *sqledbterritoryinfo*. Например, установите для SQLDBCODESET `UTF-8`, а для SQLDBLOCALE - любой действительный код территории (например, `US`).

Понятия, связанные с данным:

- “Реализация Unicode в DB2” на стр. 287

Ссылки, связанные с данной темой:

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

Литералы Unicode

Литералы Unicode могут быть заданы двумя способами:

- Как графическая строковая константа с использованием формата G’...’ или N’....’ . Любой заданный таким образом литерал будет преобразован менеджером баз данных из кодовой страницы прикладной программы в 16-битный Unicode.
- Как шестнадцатеричная строка Unicode с использование формата UX’....’ или GX’....’. Длина константы, указанной в кавычках после UX или GX, должна быть кратной четырем шестнадцатеричным цифрам, старший байт идет в конце. Каждая группа из четырех цифр представляет один кодовый элемент 16-битного Unicode. Обратите внимание на то, что заменяющие символы всегда появляются парами, поэтому вам понадобится две 4-значных группы для представления верхнего и нижнего заменяющего символа.

При использовании процессора командной строки (CLP) первый метод проще, если символ UCS-2 существует в локальной кодовой странице прикладной программы (например, при вводе любого символа кодовой страницы 850 с терминала, использующего кодовую страницу 850). Второй метод должен использоваться для символов, выходящих за пределы совокупности кодовой страницы прикладной программы (например, при вводе символов японского языка с терминала, использующего кодовую страницу 850).

Понятия, связанные с данным:

- “Кодировка символов Unicode” на стр. 285
- “Реализация Unicode в DB2” на стр. 287

Ссылки, связанные с данной темой:

- “Constants” в книге *SQL Reference, Том 1*

Сравнение строк в базе данных Unicode

Поиск по шаблону - одна из областей, в которой поведение существующих баз данных MBCS несколько отличается от поведения базы данных UCS-2.

Для баз данных MBCS в DB2® UDB текущее поведение следующее: Если искомая строка содержит данные MBCS, шаблон может включать в себя как символы SBCS, так и прочие символы (не SBCS). Специальные символы в шаблоне интерпретируются так:

- Однобайтный символ подчеркивания соответствует одному символу SBCS.
- Двухбайтный символ подчеркивания соответствует одному символу MBCS.
- Символ процента (как SBCS, так и DBCS) соответствует строке из нулевого или большего числа символов (SBCS или прочих).

В базе данных Unicode нет реальных различий между ″однобайтными″ и ″двухбайтными″ символами; каждый 16-битный символ занимает два байта. Хотя формат UTF-8 представляет собой кодирование символов Unicode со смешанным числом байтов, в нем нет реальных различий между символами SBCS и MBCS. Каждый символ рассматривается как символ Unicode, независимо от числа его байтов, используемых в формате UTF-8. При указании выражения символьной строки или графической строки символ подчеркивания соответствует одному символу Unicode, а символ процента соответствует строке из нулевого или большего числа символов Unicode. Для подстановки одного расширенного символа GRAPHIC необходимо два символа подчеркивания, так как одному графическому символу GRAPHIC в столбце GRAPHIC соответствует два символа подчеркивания UCS-2.

Со стороны клиента выражения символьных строк из кодовой страницы клиента будут преобразованы менеджером баз данных в UTF-8. В кодовых страницах клиента SBCS нет двухбайтных символов процента и подчеркивания, но каждая поддерживаемая кодовая страница содержит однобайтный символ процента (соответствующий U+0025) и однобайтный символ подчеркивания (соответствующий U+005F). Интерпретация специальных символов для базы данных UCS-2:

- Однобайтный символ подчеркивания (U+0025) соответствует одному символу UCS-2 в выражении графической строки или одному символу UTF-8 в выражении символьной строки.
- Однобайтный символ процента (U+005F) соответствует строке из нулевого или большего числа символов UCS-2 в выражении графической строки или строке из нулевого или большего числа символов UTF-8 в выражении символьной строки.

Кодовые страницы DBCS также поддерживают двухбайтный символ процента (соответствующий U+FF05) и двухбайтный символ подчеркивания (соответствующий U+FF3F). У этих символов нет специальных значений для базы данных UCS-2.

Для необязательного ″эскейп-выражения″, которое указывает, что символ используется для изменения специального значения символов подчеркивания и процента, поддерживаются только символы ASCII или символы, которые расширяются до двухбайтной последовательности UTF-8. Если указать управляющий символ, расширяющийся в трехбайтное значение UTF-8, будет возвращено сообщение об ошибке (SQL0130N, SQLSTATE 22019).

Понятия, связанные с данным:

- “Кодировка символов Unicode” на стр. 285
- “Реализация Unicode в DB2” на стр. 287

Ссылки, связанные с данной темой:

- “Character strings” в книге *SQL Reference, Том 1*
- “Graphic strings” в книге *SQL Reference, Том 1*

Приложение C. DB2 Universal Database - техническая информация

Обзор технической информации DB2 Universal Database

Техническую информацию DB2 Universal Database можно получить в следующих форматах:

- Книги (в формате PDF и как печатные копии)
- Дерево тем (в формате HTML)
- Справка по инструментам DB2 (в формате HTML)
- Программы примеров (в формате HTML)
- Справка командной строки
- Обучающие программы

В этом разделе приводится обзор поставляемой технической информации с возможными способами ее получения.

Пакеты FixPak для документации DB2

IBM может периодически выпускать пакеты FixPak к документации. Пакеты FixPak к документации позволяют обновлять информацию, установленную с *компакт-диска документации HTML для DB2*, когда становится доступной новая информация.

Примечание: После установки пакетов FixPaks к документации ваша документация в формате HTML будет содержать более свежую информацию, чем печатные руководства по DB2 и книги в формате PDF.

Категории технической информации DB2

Техническая информация DB2 подразделена на следующие категории:

- Базовая информация о DB2
- Информация об управлении
- Информация о разработке программ
- Информация о возможностях для бизнеса
- Информация о DB2 Connect
- Информация Начинаем работу
- Информация по обучающим программам
- Информация о дополнительных компонентах
- Замечания по выпуску

В следующих таблицах содержится информация, необходимая для заказа печатных копий, печати или просмотра файлов PDF, а также поиска каталогов HTML для каждой книги библиотеки DB2. Полное описание каждой из книг библиотеки DB2 можно посмотреть в центре публикаций IBM на странице www.ibm.com/shop/publications/order

Для каждой категории информации на компакт-диске документации в формате HTML предусмотрен свой каталог установки:

путь_компакт_диска_html/doc/htmlcd/*%L*/*категория*

где:

- *путь_компакт_диска_html* - каталог, где установлен компакт-диск HTML.
- *%L* - идентификатор языка. Например, ru_RU.
- *категория* - идентификатор категории. Например, core - идентификатор базовой информации DB2.

В следующих таблицах в столбце имен файла PDF символ на шестой позиции в имени файла обозначает национальную версию книги. Например, имя файла db2d1e80 говорит о том, что это английская версия книги *Administration Guide: Planning* (Руководство администратора: Планирование), а имя файла db2d1r80 соответствует русской версии этой же книги. Для обозначений языков используются на шестой позиции имени файла используются следующие буквы:

Язык	**Обозначение**
Арабский	w
Бразильский португальский	b
Болгарский	u
Хорватский	9
Чешский	x
Датский	d
Голландский	q
Английский	e
Финский	y
Французский	f
Немецкий	g
Греческий	a
Венгерский	h
Итальянский	i
Японский	j
Корейский	k
Норвежский	n
Польский	p
Португальский	v
Румынский	8
Русский	r

Упрощенный китайский	c
Словацкий	7
Словенский	l
Испанский	z
Шведский	s
Традиционный китайский	t
Turkish	m

Если **номера формы нет**, это значит, что книга доступна только в электронном виде, и для нее не существует печатной версии.

Базовая информация о DB2

Информация в этой категории охватывает темы DB2, существенные для всех пользователей DB2. Информация в этой категории будет полезна и программисту, и администратору баз данных, и тому, кто работает с DB2 Connect, Менеджером хранилищ DB2 или с другими продуктами DB2.

Каталог установки для данной категории - `doc/htmlcd/%L/core`.

Таблица 28. Базовая информация о DB2

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Command Reference	SC09-4828	db2n0x80
IBM DB2 Universal Database Glossary (Глоссарий IBM DB2 Universal Database)	Номера формы нет	db2t0x80
IBM DB2 Universal Database Master Index	SC09-4839	db2w0x80
IBM DB2 Universal Database Message Reference, Volume 1 (Справочник по сообщениям IBM DB2 Universal Database, том 1)	GC09-4840 (GH43-0197)	db2m1x80
IBM DB2 Universal Database Message Reference, Volume 2 (Справочник по сообщениям IBM DB2 Universal Database, том 2)	GC09-4841 (GH43-0196)	db2m2x80
IBM DB2 Universal Database What's New (IBM DB2 Universal Database. Что нового)	SC09-4848 (GH43-0198-00)	db2q0x80

Информация об управлении

Информация в этой категории охватывает темы, необходимые для эффективной разработки, реализации и обслуживания баз данных, хранилищ данных и систем объединения DB2.

Каталог установки для данной категории - `doc/htmlcd/%L/admin`.

Таблица 29. Информация об управлении

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Administration Guide: Planning (Руководство администратора IBM DB2 Universal Database: Планирование)	SC09-4822 (GH43-0200)	db2d1x80
IBM DB2 Universal Database Administration Guide: Implementation (Руководство администратора IBM DB2 Universal Database: Реализация)	SC09-4820 (GH43-0202)	db2d2x80
IBM DB2 Universal Database Administration Guide: Performance (Руководство администратора IBM DB2 Universal Database: Производительность)	SC09-4821 (GH43-0201)	db2d3x80
IBM DB2 Universal Database Administrative API Reference	SC09-4824	db2b0x80
IBM DB2 Universal Database Data Movement Utilities Guide and Reference	SC09-4830	db2dmx80
IBM DB2 Universal Database Data Recovery and High Availability Guide and Reference (Справочное руководство по восстановлению данных и высокой доступности IBM DB2 Universal Database)	SC09-4831 (SH43-0210)	db2hax80
IBM DB2 Universal Database Data Warehouse Center Administration Guide	SC27-1123	db2ddx80
IBM DB2 Universal Database Federated Systems Guide	GC27-1224	db2fpx80

Таблица 29. Информация об управлении (продолжение)

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Guide to GUI Tools for Administration and Development (Руководство IBM DB2 Universal Database по инструментам GUI для управления и разработки)	SC09-4851 (GH43-0203)	db2atx80
IBM DB2 Universal Database Replication Guide and Reference	SC27-1121	db2e0x80
IBM DB2 Installing and Administering a Satellite Environment	GC09-4823	db2dsx80
IBM DB2 Universal Database SQL Reference, Volume 1	SC09-4844	db2s1x80
IBM DB2 Universal Database SQL Reference, Volume 2	SC09-4845	db2s2x80
IBM DB2 Universal Database System Monitor Guide and Reference	SC09-4847	db2f0x80

Информация о разработке программ

Информация в этой категории представляет особый интерес для разработчиков и программистов, работающих с DB2. Здесь вы найдете информацию о поддерживаемых языках и компиляторах, а также документацию, требуемую для обращения к DB2 при помощи разнообразных поддерживаемых интерфейсов программирования, таких как встроенный SQL, ODBC, JDBC, SQLj и CLI. При просмотре этой информации в электронном виде доступен также набор программ примеров DB2 в формате HTML.

Каталог установки для данной категории - `doc/htmlcd/%L/ad`.

Таблица 30. Информация о разработке программ

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Application Development Guide: Building and Running Applications	SC09-4825	db2axx80

Таблица 30. Информация о разработке программ (продолжение)

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Application Development Guide: Programming Client Applications	SC09-4826	db2a1x80
IBM DB2 Universal Database Application Development Guide: Programming Server Applications	SC09-4827	db2a2x80
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1	SC09-4849	db2l1x80
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2	SC09-4850	db2l2x80
IBM DB2 Universal Database Data Warehouse Center Application Integration Guide	SC27-1124	db2adx80
IBM DB2 XML Extender Administration and Programming	SC27-1234	db2sxx80

Информация о возможностях для бизнеса

Информация в этой категории описывает, как использовать компоненты, расширяющие возможности центров данных и аналитической обработки в DB2 Universal Database.

Каталог установки для данной категории - `doc/htmlcd/%L/wareh`.

Таблица 31. Информация о возможностях для бизнеса

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Warehouse Manager Information Catalog Center Administration Guide	SC27-1125	db2dix80
IBM DB2 Warehouse Manager Installation Guide	GC27-1122	db2idx80

Информация о DB2 Connect

Информация в этой категории описывает, как работать с данными хоста или iSeries при помощи DB2 Connect Enterprise Edition или DB2 Connect Personal Edition.

Каталог установки для данной категории - `doc/htmlcd/%L/conn`.

Таблица 32. Информация о DB2 Connect

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
Смысловые коды APPC, CPI-C и SNA	Номера формы нет	db2apx80
IBM Connectivity Supplement (Дополнение по возможностям соединений IBM)	Номера формы нет	db2h1x80
IBM DB2 Connect Quick Beginnings for DB2 Connect Enterprise Edition	GC09-4833	db2c6x80
IBM DB2 Connect Quick Beginnings for DB2 Connect Personal Edition (Быстрый старт DB2 Connect для DB2 Connect Personal Edition)	GC09-4834 (GH43-0223)	db2c1x80
IBM DB2 Connect User's Guide (Руководство пользователя IBM DB2 Connect)	SC09-4835 (GH43-0199)	db2c0x80

Информация Начинаем работу

Информация в этой категории полезна при установке и конфигурировании серверов, клиентов и других продуктов DB2.

Каталог установки для этой категории - `doc/htmlcd/%L/start`.

Таблица 33. Информация Начинаем работу

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Quick Beginnings for DB2 Clients (Быстрый старт IBM DB2 Universal Database для клиентов DB2)	GC09-4832 (GH43-0222)	db2itx80

Таблица 33. Информация Начинаем работу (продолжение)

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
IBM DB2 Universal Database Quick Beginnings for DB2 Servers (Быстрый старт IBM DB2 Universal Database для серверов DB2)	GC09-4836 (GH43-0221)	db2isx80
IBM DB2 Universal Database Quick Beginnings for DB2 Personal Edition	GC09-4838	db2i1x80
IBM DB2 Universal Database Installation and Configuration Supplement (Дополнение по установке и настройке IBM DB2 Universal Database)	GC09-4837 (GH43-0220)	db2iyx80
IBM DB2 Universal Database Quick Beginnings for DB2 Data Links Manager	GC09-4829	db2z6x80

Информация по обучающим программам

Обучающие программы знакомят вас с функциями DB2 и обучают выполнению различных задач.

Каталог установки для этой категории - `doc/htmlcd/%L/tutr`.

Таблица 34. Информация по обучающим программам

Название	Номер формы (в скобках - для русской версии)	Имя файла PDF
Business Intelligence Tutorial: Introduction to the Data Warehouse	Номера формы нет	db2tux80
Business Intelligence Tutorial: Extended Lessons in Data Warehousing	Номера формы нет	db2tax80
Development Center Tutorial for Video Online using Microsoft Visual Basic	Номера формы нет	db2tdx80
Information Catalog Center Tutorial	Номера формы нет	db2aix80
Video Central for e-business Tutorial	Номера формы нет	db2twx80
Visual Explain Tutorial	Номера формы нет	db2tvx80

Информация о дополнительных компонентах

Информация в этой категории описывает, как работать с дополнительными компонентами DB2.

Каталог установки для этой категории - `doc/htmlcd/%L/opt`.

Таблица 35. Информация о дополнительных компонентах

Название	Номер формы	Имя файла PDF
IBM DB2 Life Sciences Data Connect Planning, Installation, and Configuration Guide	GC27-1235	db2lsx80
IBM DB2 Spatial Extender User's Guide and Reference	SC27-1226	db2sbx80
IBM DB2 Universal Database Data Links Manager Administration Guide and Reference	SC27-1221	db2z0x80
IBM DB2 Universal Database Net Search Extender Administration and Programming Guide **Примечание:** Этот документ в виде HTML не устанавливается с компакт-диска документации HTML.	SH12-6740	Нет

Замечания по выпуску

В замечаниях по выпуску предоставляется дополнительная информация, относящаяся конкретно к вашему выпуску продукта и уровню FixPak. В них также содержится сводная информация по обновлениям к документации, включаемым в каждый выпуск и пакет FixPak.

Таблица 36. Замечания по выпуску

Название	Номер формы	Имя файла PDF
Замечания по выпуску DB2	Смотрите примечание.	Смотрите примечание.
Замечания по установке DB2	Доступны только на компакт-диске продукта.	Доступны только на компакт-диске продукта.

Примечание: HTML-версию Замечаний по выпуску можно вызвать через Информационный центр или с компакт-диска продукта. Чтобы посмотреть файл ASCII на платформах UNIX, откройте файл `Release.Notes`. Он расположен в каталоге `DB2DIR/Readme/%L`, где `%L` - национальная версия, а `DB2DIR`:

- `/usr/opt/db2_08_01` - в AIX
- `/opt/IBM/db2/V8.1` - в других операционных системах UNIX

Задачи, связанные с данной темой:

- “Печать книг DB2 из файлов PDF” на стр. 304
- “Заказ печатных копий книг DB2” на стр. 305
- “Обращение к электронной справке” на стр. 306
- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 310
- “Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML” на стр. 311

Печать книг DB2 из файлов PDF

Можно напечатать книги по DB2 из файлов PDF с компакт-диска *Документация по DB2 в формате PDF*. При помощи Adobe Acrobat Reader можно напечатать книгу целиком или же определенный диапазон страниц.

Предварительные требования:

У вас должен быть Adobe Acrobat Reader. Его можно получить на сайт Adobe по адресу www.adobe.com

Процедура:

Чтобы напечатать книгу DB2 из файла PDF:

1. Вставьте компакт-диск *Документация по DB2 в формате PDF* в дисковод. В операционных системах UNIX смонтируйте компакт-диск *Документация по DB2 в формате PDF*. Подробности о том, как смонтировать компакт-диск в операционных системах UNIX, смотрите в книге *Quick Beginnings* (Быстрый старт).
2. Запустите Adobe Acrobat Reader.
3. Откройте файл PDF из одного из следующих мест:
   - В операционных системах Windows:

     Из каталога *x*`:\doc\`*язык*, где *x* - буква дисковода компакт-дисков, а *язык* - двухсимвольный код территории, соответствующий вашему языку (например, RU для русского).
   - В операционных системах UNIX:

     Из каталога `/`*cdrom*`/doc/`*%L* на компакт-диске, где */cdrom* - точка установки компакт-диска, а *%L* - имя требуемой национальной версии.

Задачи, связанные с данной темой:

- “Заказ печатных копий книг DB2” на стр. 305
- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 310
- “Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML” на стр. 311

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 295

Заказ печатных копий книг DB2

Процедура:

Чтобы заказать печатные книги:

- Обратитесь к авторизованному дилеру или торговому представителю IBM. Локального представителя IBM можно найти во каталоге контактных адресов IBM (IBM Worldwide Directory of Contacts) по адресу www.ibm.com/planetwide
- Позвоните по телефону `1-800-879-2755` в США или `1-800-IBM-4YOU` в Канаде.
- С Web-страницы Центра публикаций IBM (IBM Publications Center): www.ibm.com/shop/publications/order

Печатные копии руководств DB2 можно также получить, заказав у поставщика IBM пакеты документации (Doc Packs) для вашего продукта DB2. Пакеты документации - это избранные руководства из библиотеки DB2, облегчающие освоение приобретенного вами продукта DB2. Те же руководства доступны в формате PDF на компакт-диске *Документация по DB2 в формате PDF*, их содержание совпадает с содержанием компакт-диска *Документация по DB2 в формате HTML*.

Задачи, связанные с данной темой:

- “Печать книг DB2 из файлов PDF” на стр. 304
- “Поиск тем при обращении к Информационному центру DB2 из браузера” на стр. 307
- “Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML” на стр. 311

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 295

Обращение к электронной справке

Электронная справка, поставляемая со всеми компонентами DB2, доступна в трех вариантах:

- Справка по окну и записной книжке
- Справка командной строки
- Справка по операторам SQL

В справке по окну и записной книжке объясняются задачи, выполняемые в окне или записной книжке, и описываются органы управления. Эта справка бывает двух типов:

- Справка, вызываемая кнопкой **Справка**
- Всплывающие подсказки

Кнопка **Справка** позволяет обращаться к обзорной информации и информации о предварительных условиях. Всплывающие подсказки описывают органы управления в окне или записной книжке. Справка окна и записной книжки доступна из центров и компонентов DB2, поддерживающих пользовательский интерфейс.

Справка командной строки состоит из справки по командам и справки по сообщениям. Справка по командам объясняет синтаксис команд процессора командной строки. Справка по сообщениям описывает причины появления сообщений об ошибках и необходимые действия в ответ на ошибки.

Справка по операторам SQL состоит из справки SQL и справки SQLSTATE. Система DB2 возвращает SQLSTATE - значения, описывающие ошибки, которые могут возникнуть при выполнении оператора SQL. Справка по SQLSTATE объясняет синтаксис операторов SQL (состояния SQL и коды классов).

Примечание: Справка по SQL недоступна для операционных систем UNIX.

Процедура:

Чтобы обратиться к электронной справке:

- Для справки по окну и записной книжке нажмите кнопку **Справка** или щелкните по интересующему вас органу управления и затем нажмите клавишу **F1**. Если на странице **Общие** записной книжки **Параметры инструментов** включен переключатель **Автоматически выводить всплывающие подсказки**, всплывающие подсказки по органам управления будут появляться также при наведении на них указателя мыши.
- Для справки командной строки откройте процессор командной строки и введите:

– Для справки по командам:

```
? команда
```

где *команда* - ключевое слово для команды целиком.

Например, `? catalog` выводит справку по всем командам CATALOG, а `? catalog database` выводит справку по команде CATALOG DATABASE.

- Для справки по сообщениям:

```
? XXXnnnnn
```

где *XXXnnnnn* - идентификатор существующего сообщения.

Например, `? SQL30081` выводит справку по сообщению SQL30081.

- Для справки по оператору SQL введите в командной строке DB2:

```
? sqlstate или ? код класса
```

где *sqlstate* - допустимый пятизначный код SQL, а *код класса* - первые две цифры sqlstate.

Например, `? 08003` выводит справку по состоянию SQL 08003, а `? 08` выводит справку по коду класса 08.

Задачи, связанные с данной темой:

- “Поиск тем при обращении к Информационному центру DB2 из браузера” на стр. 307
- “Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML” на стр. 311

Поиск тем при обращении к Информационному центру DB2 из браузера

Обращение к Информационному центру DB2 из браузера дает доступ к информации, необходимой для полного использования всех возможностей DB2 Universal Database и DB2 Connect. Информационный центр DB2 содержит также сведения по основным возможностям и компонентам DB2, включая репликацию, хранилище данных, метаданные и модули расширения DB2.

При обращении из браузера Информационный центр DB2 будет состоять из следующих основных элементов:

Дерево навигации

Дерево навигации расположено в левом фрейме окна браузера. Его можно разворачивать и сворачивать для показа и скрытия тем, глоссария и главного указателя Информационного центра DB2.

Панель инструментов навигации

Панель инструментов навигации расположена в правом фрейме окна браузера. Она содержит кнопки, позволяющие вести поиск в Информационном центре DB2, скрывать дерево навигации и искать текущую тему в этом дереве.

Фрейм содержимого

Фрейм содержимого - это правый нижний фрейм окна браузера. Если щелкнуть по ссылке в дереве навигации, по результату поиска или же перейти по ссылке из другой темы или главного указателя, во фрейме содержимого выводятся темы Информационного центра DB2.

Предварительные требования:

Для доступа к Информационному центру DB2 из браузера необходим один из следующих браузеров:

- Microsoft Explorer Версии 5 или новее
- Netscape Navigator Версии 6.1 или новее

Ограничения:

Информационный центр DB2 содержит только те наборы тем, которые вы установили с компакт-диска *Документация по DB2 в формате HTML*. Если при попытке перехода к теме по ссылке ваш браузер возвратил ошибку `Файл не найден`, необходимо установить дополнительные наборы тем с компакт-диска *Документация по DB2 в формате HTML*.

Процедура:

Чтобы найти тему по ключевым словам:

1. Нажмите на панели инструментов навигации кнопку **Поиск**.
2. В верхнем текстовом поле ввода окна Поиск введите один или несколько терминов, отражающих интересующую вас область, и нажмите кнопку **Поиск**. В поле **Результаты** будет выведен список тем, ранжированных в порядке точности соответствия условиям поиска. Число рядом с каждым результатом поиска отражает точность соответствия (чем больше число, тем лучше соответствие).

   Ввод дополнительных слов для поиска повышает точность запроса, сокращая количество возвращаемых тем.
3. В поле **Результаты** щелкните по заголовку интересующей вас темы. Информация по этой теме будет выведена во фрейме содержимого.

Чтобы найти тему в дереве навигации:

1. Щелкните по значку с книгой у интересующего вас тематического раздела в дереве навигации. Под значком появится список подкатегорий этого раздела.

2. Щелкая по значкам с книгой, раскрывайте далее эти подкатегории, пока не дойдете до категории с нужными сведениями. Заголовки категорий, содержащих ссылки на темы справки, при наведении на них указателя мыши принимают вид подчеркнутой ссылки. Отдельные темы в дереве навигации обозначаются значком страницы.
3. Щелкните по ссылке на нужную тему. Информация по этой теме будет выведена во фрейме содержимого.

Чтобы найти тему или термин в главном указателе:

1. Щелкните по категории “Указатель” в дереве навигации. Категория примет вид дерева навигации со списком расположенных в алфавитном порядке ссылок.
2. Щелкните в этом дереве навигации по ссылке на первый символ термина, относящегося к интересующей вас теме. Во фрейме содержимого появится список терминов, начинающихся с этого символа. Термины, которым соответствует несколько вхождений указателя, будут отмечены значком книги.
3. Щелкните по значку у интересующего вас термина. Под этим термином появится список подчиненных терминов и тем справки. Темы обозначаются значком страницы с подчеркнутым заголовком.
4. Щелкните по заголовку нужной темы. Информация по теме будет выведена во фрейме содержимого.

Понятия, связанные с данным:

- “Доступность” на стр. 317
- “Информационный центр DB2 при обращении из браузера” на стр. 320

Задачи, связанные с данной темой:

- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 310
- “Обновление документации HTML, установленной на вашем компьютере” на стр. 312
- “Устранение ошибок при поиске в документации DB2 с помощью Netscape 4.x” на стр. 314
- “Поиск в документации DB2” на стр. 315

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 295

Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления

Информационный центр DB2 обеспечивает быстрый доступ к информации о программном продукте DB2. Он доступен во всех операционных системах, где доступны инструменты управления DB2.

При обращении из инструментов управления в Информационном центре DB2 выводятся шесть типов информации.

Задачи Основные задания, которые вы можете выполнить в DB2.

Основные понятия
Основные понятия DB2.

Справочник
Справочная информацию по таким элементам DB2, как ключевые слова, команды и API.

Устранение неисправностей
Сообщения об ошибках и информация, которая поможет вам при возникновении проблем с DB2.

Примеры
Ссылки на тексты HTML примеров программ, поставляемых с DB2.

Обучающие программы
Пошаговая помощь для освоения возможностей DB2.

Предварительные требования:

Некоторые ссылки в Информационном центре DB2 указывают на сайты в Интернете. Чтобы посмотреть содержимое таких ссылок, надо соединиться с Интернетом.

Процедура:

Чтобы найти информацию о продукте при обращении к Информационному центру DB2 из инструментов:

1. Запустите Информационный центр DB2 одним из следующих способов:
   - На панели графических инструментов управления щелкните по значку **Информационный центр**. Этот пункт можно также выбрать в меню **Справка**.
   - Введите в командной строке **db2ic**.
2. Щелкните по вкладке типа информации, связанного с информацией, которую вы ищете.
3. Разверните дерево и щелкните по интересующей вас теме. Информационный центр запускает браузер для вывода этой информации.

4. Чтобы найти информацию, не просматривая списки, щелкните по значку **Поиск** справа от списка.

   Когда Информационный центр запустит браузер для вывода информации, вы можете выполнять поиск по всему тексту, щелкнув по значку **Поиск** на навигационной панели.

Понятия, связанные с данным:

- “Доступность” на стр. 317
- “Информационный центр DB2 при обращении из браузера” на стр. 320

Задачи, связанные с данной темой:

- “Поиск тем при обращении к Информационному центру DB2 из браузера” на стр. 307
- “Поиск в документации DB2” на стр. 315

Просмотр технической документации непосредственно с компакт-диска Документация по DB2 в формате HTML

Все темы в формате HTML, которые можно установить с компакт-диска *Документация по DB2 в формате HTML*, можно также читать непосредственно с этого компакт-диска. Поэтому просмотр документации возможен и без ее установки.

Ограничения:

Поскольку справка по инструментам устанавливается с компакт-диска продукта DB2, а не с компакт-диска *Документация по DB2 в формате HTML*, для просмотра справки необходимо установить этот продукт DB2.

Процедура:

1. Вставьте в дисковод компакт-диск *Документация по DB2 в формате HTML*. В операционных системах UNIX смонтируйте компакт-диск *Документация по DB2 в формате HTML*. Подробности о том, как смонтировать компакт-диск в операционных системах UNIX, смотрите в книге *Quick Beginnings* (Быстрый старт).
2. Запустите ваш браузер и откройте нужный файл:
   - Для операционных систем Windows:

     ```
     e:\program files\IBM\SQLLIB\doc\htmlcd\%L\index.htm
     ```

     где *e* - дисковод компакт-дисков, а %L - необходимая вам национальная версия документации, например, **ru_RU** для русского языка.
   - Для операционных систем UNIX:

     ```
     /cdrom/program files/IBM/SQLLIB/doc/htmlcd/%L/index.htm
     ```

где */cdrom/* - положение, где монтируется компакт-диск, а %L необходимая вам национальная версия документации, например, **ru_RU** для русского языка.

Задачи, связанные с данной темой:

- “Поиск тем при обращении к Информационному центру DB2 из браузера” на стр. 307
- “Копирование файлов с компакт-диска Документация по DB2 в формате HTML на Web-сервер” на стр. 313

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 295

Обновление документации HTML, установленной на вашем компьютере

Теперь есть возможность обновлять файлы HTML, установленные с компакт-диска *Документация по DB2 в формате HTML*, по мере поступления обновлений от IBM. Это можно сделать одним из следующих способов:

- С помощью Информационного центра (если у вас установлены инструменты управления DB2 с графическим интерфейсом).
- С помощью загрузки и применения пакета обновлений FixPak для документации HTML DB2.

Примечание: Эти изменения затронут НЕ программный код DB2, а лишь документацию HTML, установленную с компакт-диска *Документация по DB2 в формате HTML*.

Процедура:

Чтобы изменить вашу локальную документацию с помощью Информационного центра:

1. Запустите Информационный центр DB2 одним из следующих способов:
   - На панели графических инструментов управления щелкните по значку **Информационный центр**. Этот пункт можно также выбрать в меню **Справка**.
   - Введите в командной строке **db2ic**.
2. Убедитесь, что у вашего компьютера есть доступ в Интернет; при необходимости программа обновления будет загружать последние пакеты документации FixPak с сервера IBM.
3. Чтобы начать обновление, выберите в меню **Информационный центр —> Обновить локальную документацию**.
4. Если требуется, введите информацию о вашем прокси-сервере, чтобы соединиться с Интернетом.

При наличии свежего пакета документации FixPak Информационный центр загрузит и применит его.

Чтобы загрузить и применить пакет документации FixPak вручную:

1. Убедитесь, что ваш компьютер соединен с Интернетом.
2. Откройте в вашем браузере страницу поддержки DB2: www.ibm.com/software/data/db2/udb/winos2unix/support.
3. Перейдите по ссылке для Версии 8 и найдите ссылку ″Documentation FixPaks″ (Пакеты документации FixPak).
4. Определите, устарела ли версия вашей локальной документации, сравнив уровень пакета FixPak с уровнем установленной у вас документации. Текущая документация на вашем компьютере имеет следующий уровень: **DB2 v8.1 GA**.
5. Если доступна более новая версия документации, загрузите пакет FixPak для вашей операционной системы. Один пакет FixPak используется для всех платформ Windows, другой пакет FixPak - для всех платформ UNIX.
6. Примените пакет FixPak:
   - Для операционных систем Windows: Пакет документации FixPak - это самораспаковывающийся zip-архив. Поместите загруженный пакет FixPak в пустой каталог и запустите его там. Будет создан исполняемый файл **setup**, при запуске которого начинается установка пакета FixPak.
   - Для операционных систем UNIX: Пакет документации FixPak - это упакованный файл tar.Z. Распакуйте и разархивируйте этот файл. При этом будет создан каталог `delta_install` со сценарием **installdocfix**. Запустите этот сценарий, чтобы установить пакет документации FixPak.

Задачи, связанные с данной темой:

- “Копирование файлов с компакт-диска Документация по DB2 в формате HTML на Web-сервер” на стр. 313

Ссылки, связанные с данной темой:

- “Обзор технической информации DB2 Universal Database” на стр. 295

Копирование файлов с компакт-диска Документация по DB2 в формате HTML на Web-сервер

Вся библиотека с информацией DB2 поступает к вам на компакт-диске *Документация DB2 в формате HTML*; для облегчения доступа к ней ее можно установить на Web-сервере. Для этого просто скопируйте эту документацию на нужных вам языках на ваш Web-сервер.

Примечание: При обращении к документации HTML с Web-сервера через низкоскоростное соединение загрузка может идти медленно.

Процедура:

Чтобы скопировать на Web-сервер файлы с компакт-диска *Документация по DB2 в формате HTML*, используйте соответствующий путь источника:

- Для операционных систем Windows:
  `E:\program files\IBM\SQLLIB\doc\htmlcd\%L\*.*`

  где *E* - буква дисковода компакт-дисков, а *%L* - идентификатор языка.
- Для операционных систем UNIX:
  `/cdrom/program files/IBM/SQLLIB/doc/htmlcd/%L/*.*`

  где *cdrom* - точка монтирования компакт-диска, а *%L* - идентификатор языка.

Задачи, связанные с данной темой:

- “Поиск в документации DB2” на стр. 315

Ссылки, связанные с данной темой:

- “Поддерживаемые DB2 языки интерфейса, национальные версии и кодовые страницы” в книге *Quick Beginnings for DB2 Servers*
- “Обзор технической информации DB2 Universal Database” на стр. 295

Устранение ошибок при поиске в документации DB2 с помощью Netscape 4.x

Большинство проблем при поиске связаны с поддержкой Java, обеспечиваемой браузерами. В этой задаче описываются возможные обходные приемы для этих проблем.

Процедура:

При работе с Netscape 4.x обычно возникает проблема отсутствия или неверного местонахождения класса защиты. Попытайтесь применить описанный ниже прием, в особенности если на консоли Java браузера появилась следующая строка:

```
Невозможно найти класс java/security/InvalidParameterException
```

- В операционных системах Windows:

  Скопируйте с *компакт-диска документации HTML DB2* файл `x:\program files\IBM\SQLLIB\doc\htmlcd\locale\InvalidParameterException.class`, где *x* - буква дисковода компакт-дисков, а *locale* - нужная национальная версия, в подкаталог `java\classes\java\security\` каталога установки вашего браузера Netscape.

  Примечание: Возможно, надо будет создать подкаталоги `java\security\`.
- В операционных системах UNIX:

Скопируйте с компакт-диска *Документация по DB2 в формате HTML* файл */cdrom*/program files/IBM/SQLLIB/doc/htmlcd/*locale*/InvalidParameterException.class, где *cdrom* - точка монтирования компакт-диска, а *locale* - нужная национальная версия, в подкаталог java/classes/java/security/ каталога установки вашего браузера Netscape.

Примечание: Возможно, надо будет создать подкаталоги java/security/.

Если ваш браузер Netscape по-прежнему не может вывести окно ввода поиска, попытайтесь сделать следующее:

- Закройте все экземпляры браузеров Netscape, чтобы в компьютере не выполнялся программный код Netscape. Затем откройте новый экземпляр браузера Netscape и попытайтесь выполнить поиск снова.
- Очистите кэш браузера.
- Попробуйте использовать другую версию Netscape или другой браузер.

Задачи, связанные с данной темой:

- “Поиск в документации DB2” на стр. 315

Поиск в документации DB2

Необходимую вам информацию можно найти в библиотеке документации DB2. Если щелкнуть по значку поиска на навигационной панели инструментов Информационного центра DB2 (при обращении из браузера), откроется всплывающее окно поиска. Загрузка результатов поиска может занять некоторое время в зависимости от скорости вашего компьютера и сети.

Предварительные требования:

Требуется Netscape Версии 6.1 или новее или же Microsoft Internet Explorer Версии 5 или новее. В вашем браузере должна быть включена поддержка Java.

Ограничения:

Ограничения при поиске документации:

- Поиск не регистрозависим.
- Логические условия поиска не поддерживаются.
- Поиск с символами подстановки и частичный поиск не поддерживается. Так, при поиске *java** (или *java*) это вхождение будет восприниматься просто как строка символов *java** (или *java*), и, например, вхождение *javadoc* не будет найдено.

Процедура:

Для поиска документации DB2:

1. Щелкните по значку **Поиск** на панели инструментов навигации.
2. В верхнем текстовом поле ввода окна Поиск введите (через пробел) один или несколько терминов, отражающих интересующую вас область, и нажмите кнопку **Поиск**. В поле **Результаты** будет выведен список тем, ранжированных в порядке точности соответствия условиям поиска. Число рядом с каждым результатом поиска отражает точность соответствия (чем больше число, тем лучше соответствие).

   Ввод дополнительных слов для поиска повышает точность запроса, сокращая количество возвращаемых тем.
3. В списке **Результаты** щелкните по заголовку интересующей вас темы. Информация по этой теме будет выведена во фрейме содержимого Информационного центра DB2.

Примечание: При выполнении поиска его первый результат (с высшим рангом соответствия) автоматически загружается во фрейм браузера. Чтобы просмотреть содержимое других результатов поиска, щелкните по нужному результату в списке результатов.

Задачи, связанные с данной темой:

- “Устранение ошибок при поиске в документации DB2 с помощью Netscape 4.x” на стр. 314

Электронная информации об устранении неисправностей DB2

В выпуске DB2® UDB Версии 8 больше нет *Руководства по устранению неисправностей*. Информация по устранению неисправностей, ранее содержавшаяся в этом руководстве, теперь включена в другие публикации по DB2. Это позволяет давать вам наиболее свежую доступную информацию. Чтобы найти информацию по утилитам и функциям устранения неисправностей DB2, вызовите Информационный центр DB2 из любого инструмента DB2.

Если вы сталкиваетесь с проблемами и вам нужна помощь в поиске причин и решений, обратитесь на сайт поддержки DB2 (DB2 Online Support). Этот сайт содержит большую, постоянно обновляемую базу данных публикаций DB2, технических замечаний, записей APAR (о проблемах с продуктом), пакетов FixPaks и прочих ресурсов. Для решения ваших проблем можно воспользоваться поиском по сайту.

Сайт поддержки DB2 можно вызвать по адресу www.ibm.com/software/data/db2/udb/winos2unix/support, а также нажатием кнопки **Электронная поддержка** в Информационном центре DB2. На этом сайте теперь доступна также часто обновляемая информация, например, список внутренних кодов ошибок DB2.

Понятия, связанные с данным:

- “Информационный центр DB2 при обращении из браузера” на стр. 320

Задачи, связанные с данной темой:

- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 310

Доступность

Функции доступности помогают пользователям с физическими недостатками, например с ограниченной подвижностью или недостаточным зрением, с успехом пользоваться программными продуктами. В DB2® Universal Database Версии 8 применяются следующие основные функции доступности:

- DB2 позволяет использовать клавиатуру вместо мыши для работы с любыми функциями. Смотрите раздел “Ввод с клавиатуры и навигация”.
- DB2 позволяет настраивать размер и цвет шрифтов. Смотрите раздел “Доступность и дисплей”.
- DB2 позволяет использовать как визуальные, так и звуковые средства оповещения. Смотрите раздел “Альтернативные средства предупреждения” на стр. 318.
- DB2 поддерживает возможности доступности в программах, которые используют API доступности Java™. Смотрите раздел “Совместимость с технологиями для людей с физическими недостатками” на стр. 318.
- DB2 поставляется с документацией в формате, обеспечивающем доступность. Смотрите раздел “Удобный формат документации” на стр. 318.

Ввод с клавиатуры и навигация

Ввод с клавиатуры

Можно работать с инструментами DB2, используя только клавиатуру. Для выполнения операций вместо мыши можно использовать также клавиши или сочетания клавиш.

Фокус ввода с клавиатуры

В системах на основе UNIX фокус ввода с клавиатуры выделяется на экране; тем самым указывается активная область окна, в которую будут вводиться символы при нажатии клавиш.

Доступность и дисплей

В инструментах DB2 используются средства, улучшающие пользовательский интерфейс и облегчающие работу для пользователей со слабым зрением. К них относится поддержка настраиваемых свойств шрифтов.

Параметры шрифтов

Инструменты DB2 позволяют вам при помощи записной книжки Свойства инструментов выбрать цвет, размер и тип шрифта, используемого в меню и для диалоговых окон.

Независимость от цвета

Чтобы использовать любые функции этого продукта, вам не требуется различать цвета.

Альтернативные средства предупреждения

Вы можете задать, в каком виде получать оповещения: в виде звуковых или визуальных сигналов.

Совместимость с технологиями для людей с физическими недостатками

Интерфейс инструментов DB2 поддерживает API доступности Java, что позволяет использовать программы чтения экрана и другие технологии для пользователей с физическими недостатками.

Удобный формат документации

Документация для продуктов семейства DB2 доступна в формате HTML. Это позволяет просматривать документацию, используя предпочтения экрана, заданные для вашего браузера. Это позволяет также использовать программы чтения с экрана и другие технологии для людей с физическими недостатками.

Обучающие программы DB2

Обучающие программы DB2® помогают освоить различные аспекты DB2 Universal Database. Эти программы содержат уроки с пошаговыми указаниями по разработке программ, настройке производительности запросов SQL, работе с хранилищами данных, управлением метаданными и разработке Web-служб, использующих DB2.

Прежде, чем вы начнете:

Прежде чем обращаться к обучающим программам по приведенным ниже ссылкам, надо установить эти программы с компакт-диска *Документация по DB2 в формате HTML*.

Если вы не хотите устанавливать обучающие программы, можно просматривать их HTML-версии непосредственно с компакт-диска *Документация по DB2 в формате HTML*. На компакт-диске *Документация по DB2 в формате PDF* доступны также версии этих обучающих программ в формате PDF.

В некоторых уроках используются примеры данных или кодов программ. Описание необходимых условий для выполнения задач разных обучающих программ смотрите отдельно в каждой программе.

Обучающие программы DB2 Universal Database:

Если вы установили обучающие программы с компакт-диска *Документация по DB2 в формате HTML*, можно для просмотра материала щелкнуть по его заголовку в приведенном ниже списке.

Обучающая программа Business Intelligence Tutorial: Начальные сведения о Центре хранилищ данных
: Выполнение вводных задач работы с хранилищами данных при помощи Центра хранилищ данных.

Обучающая программа Business Intelligence Tutorial: Дополнительные уроки по хранилищам данных
: Выполнение дальнейших задач работы с хранилищами данных при помощи Центра хранилищ данных.

Обучающая программа по Центру разработки для Video Online с помощью Microsoft® Visual Basic
: Построение компонентов программ при помощи дополнительного модуля Development Center для Microsoft Visual Basic.

Обучающая программа по Центру каталогов данных
: Создание каталога данных для поиска и использования метаданных и управление им при помощи Центра каталогов данных.

Обучающая программа по Video Central для электронной коммерции
: Разработка и внедрение усовершенствованных программ DB2 Web Services с использованием продуктов WebSphere®.

Обучающая программа по Visual Explain
: Анализ, оптимизация и настройка операторов SQL для улучшения производительности при помощи Наглядного объяснения.

На этом языке нужная вам тема недоступна

На этом языке нужная вам тема недоступна.

Поэтому в отдельном окне браузера выведена английская версия этой темы.

Информационный центр DB2 при обращении из браузера

Информационный центр DB2® дает доступ ко всей информации, необходимой для полного использования возможностей DB2 Universal Database™ и DB2 Connect™ в вашей работе. Информационный центр DB2 также содержит сведения по основным возможностям и компонентам DB2, включая репликацию, хранилища данных, Центр каталогов данных, Life Sciences Data Connect и модули расширения DB2.

Информационный центр DB2 при обращении из браузера Netscape Navigator Версии 6.1 или новее или Microsoft Internet Explorer Версии 5 или новее поддерживает перечисленные ниже возможности. Для некоторых из них требуется включить поддержку Java или JavaScript:

Регулярно обновляемая документация
Постоянное обновление тем путем загрузки новейших файлов HTML.

Поиск Поиск по всем темам, установленным на вашей рабочей станции, после щелчка по значку **Поиск** на панели инструментов навигации.

Интегрированное дерево навигации
Поиск любой темы в библиотеке DB2 в одном дереве навигации. По типу содержащейся в нем информации дерево навигации организовано так:

- Задачи содержат пошаговые инструкции по достижению цели.
- Понятия помогают раскрыть содержание вопроса.
- Справочные темы содержат подробную информацию по вопросу, в том числе синтаксис операторов и команд, справку по сообщениям, требования.

Главный указатель
Доступ к информации, установленной с компакт-диска *Документация по DB2 в формате HTML* производится из главного указателя. Термины в указателе располагаются в алфавитном порядке.

Главный глоссарий
В главном глоссарии даются определения терминов, используемых Центром информации DB2. Термины в глоссарии располагаются в алфавитном порядке.

Задачи, связанные с данной темой:

- “Поиск тем при обращении к Информационному центру DB2 из браузера” на стр. 307
- “Поиск информации о продукте при обращении к Информационному центру DB2 из инструментов управления” на стр. 310
- “Обновление документации HTML, установленной на вашем компьютере” на стр. 312

Приложение D. Замечания

IBM может предлагать описанные продукты, услуги и возможности не во всех странах. Сведения о продуктах и услугах, доступных в настоящее время в вашей стране, можно получить в местном представительстве IBM. Любые ссылки на продукты, программы или услуги IBM не означают явным или неявным образом, что можно использовать только продукты, программы или услуги IBM. Разрешается использовать любые функционально эквивалентные продукты, программы или услуги, если при этом не нарушаются права IBM на интеллектуальную собственность. Однако ответственность за оценку и проверку работы любых продуктов, программ и услуг других фирм лежит на пользователе.

Фирма IBM может располагать патентами или рассматриваемыми заявками на патенты, относящимися к предмету данного документа. Получение этого документа не означает предоставления каких-либо лицензий на эти патенты. Запросы по поводу лицензий следует направлять в письменной форме по адресу:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

По поводу лицензий, связанных с использованием наборов двухбайтных символов (DBCS), обращайтесь в отдел интеллектуальной собственности IBM в вашей стране/регионе или направьте запрос в письменной форме по адресу:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

Следующий абзац не применяется в Великобритании или в любой другой стране/регионе, где подобные заявления противоречат местным законам: КОРПОРАЦИЯ INTERNATIONAL BUSINESS MACHINES ПРЕДСТАВЛЯЕТ ДАННУЮ ПУБЛИКАЦИЮ “КАК ЕСТЬ” БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ПРЕДПОЛАГАЕМЫЕ ГАРАНТИИ СОВМЕСТИМОСТИ, РЫНОЧНОЙ ПРИГОДНОСТИ И СООТВЕТСТВИЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. В некоторых странах для определенных сделок подобные оговорки не допускаются; таким образом, это утверждение может не относиться к вам.

Данная информация может содержать технические неточности и типографские опечатки. Периодически в информацию вносятся изменения, они будут включены в новые издания этой публикации. Фирма IBM может в любое время без уведомления вносить изменения и усовершенствования в продукты и программы, описанные в этой публикации.

Любые ссылки в данной информации на Web-сайты, не принадлежащие IBM, приводятся только для удобства и никоим образом не означают поддержки IBM этих Web-сайтов. Материалы этих Web-сайтов не являются частью данного продукта IBM, и вы можете использовать их только на собственную ответственность.

IBM может использовать или распространять присланную вами информацию любым способом, как фирма сочтет нужным, без каких-либо обязательств перед вами.

Если обладателю лицензии на данную программу понадобятся сведения о возможности: (i) обмена данными между независимо разработанными программами и другими программами (включая данную) и (ii) совместного использования таких данных, он может обратиться по адресу:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

Такая информация может быть предоставлена на определенных условиях (в некоторых случаях к таким условиям может относиться оплата).

Лицензированная программа, описанная в данном документе, и все лицензированные материалы, доступные с ней, предоставляются IBM на условиях IBM Customer Agreement (Соглашения IBM с заказчиком), Международного соглашения о лицензиях на программы IBM или эквивалентного соглашения.

Приведенные данные о производительности измерены в контролируемой среде. Таким образом, результаты, полученные в других операционных средах, могут существенно отличаться от них. Некоторые показатели измерены получены в системах разработки и нет никаких гарантий, что в общедоступных системах эти показатели будут теми же. Более того, некоторые результаты могут быть получены путем экстраполяции. Реальные результаты могут отличаться от них. Пользователи должны проверить данные для своих конкретных сред.

Информация о продуктах других фирм получена от поставщиков этих продуктов, из их опубликованных объявлений или из других общедоступных

источников. Фирма IBM не проверяла эти продукты и не может подтвердить точность измерений, совместимость или прочие утверждения о продуктах других фирм. Вопросы о возможностях продуктов других фирм следует направлять поставщикам этих продуктов.

Все утверждения о будущих планах и намерениях IBM могут быть изменены или отменены без уведомлений, и описывают исключительно цели фирмы.

Эта информация может содержать примеры данных и отчетов, иллюстрирующие типичные деловые операции. Чтобы эти примеры были правдоподобны, в них включены имена лиц, названия компаний и товаров. Все эти имена и названия вымышлены и любое их сходство с реальными именами и адресами полностью случайно.

ЛИЦЕНЗИЯ НА КОПИРОВАНИЕ:

Эта информация может содержать примеры прикладных программ на языках программирования, иллюстрирующих приемы программирования для различных операционных платформ. Разрешается копировать, изменять и распространять эти примеры программ в любой форме без оплаты фирме IBM для целей разработки, использования, сбыта или распространения прикладных программ, соответствующих интерфейсу прикладного программирования операционных платформ, для которых эти примера программ написаны. Эти примеры не были всесторонне проверены во всех возможных условиях. Поэтому IBM не может гарантировать их надежность, пригодность и функционирование.

Каждая копия программ примеров или программ, созданных на их основе, должна содержать следующее замечание об авторских правах:

© (*название вашей фирмы*) (*год*). Части этого кода построены на основе примеров программ IBM Corp. © Copyright IBM Corp. *_вставьте год или годы_*. Все права защищены.

Товарные знаки

Следующие термины, используемые по крайней мере в одном из документов библиотеки документации DB2 UDB, являются товарными знаками корпорации International Business Machines в Соединенных Штатах и/или в других странах.

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

Следующие термины, используемые по крайней мере в одном из документов библиотеки документации DB2 UDB, являются товарными знаками или зарегистрированными товарными знаками других компаний:

Microsoft, Windows, Windows NT и логотип Windows - товарные знаки Microsoft Corporation в Соединенных Штатах и в других странах.

Intel и Pentium - товарные знаки Intel Corporation в Соединенных Штатах и/или других странах.

Java и все товарные знаки на основе Java - товарные знаки Sun Microsystems, Inc. в Соединенных Штатах и/или в других странах.

UNIX - зарегистрированный товарный знак The Open Group в Соединенных Штатах и в других странах.

Названия других компаний, продуктов и услуг могут быть товарными знаками или марками сервиса других фирм.

Индекс

H

O

П

Р

Ф

Х

Ч

Э

Ю

Я

Как связаться с IBM

В Соединенных Штатах позвоните по одному из следующих номеров:

- 1-800-237-5511, чтобы обратиться в службу поддержки заказчиков
- 1-888-426-4343, чтобы узнать о доступных формах обслуживания.
- 1-800-IBM-4YOU (426-4968), чтобы обратиться в отдел маркетинга и продаж DB2

В Канаде позвоните по одному из следующих номеров:

- 1-800-IBM-SERV (1-800-426-7378), чтобы обратиться в службу поддержки заказчиков
- 1-800-465-9600, чтобы узнать о доступных формах обслуживания.
- 1-800-IBM-4YOU (1-800-426-4968), чтобы обратиться в отдел маркетинга и продаж DB2

Адрес отделения IBM в вашей стране или регионе можно найти на странице IBM Directory of Worldwide Contacts в Интернете по адресу www.ibm.com/planetwide

Информация о продукте

Информацию о продуктах DB2 Universal Database можно получить по телефону или в Интернете по адресу www.ibm.com/software/data/db2/udb

Этот сайт содержит свежую информацию по технической библиотеке, заказу книг, загружаемые клиенты, группы новостей, пакеты FixPaks, новости и ссылки на ресурсы в Интернете.

Если вы находитесь в США, позвоните по одному из следующих номеров:

- 1-800-IBM-CALL (1-800-426-2255), чтобы заказать продукты или получить общую информацию.
- 1-800-879-2755, чтобы заказать публикации.

Информацию о том, как связаться с IBM из других стран, смотрите на странице IBM Worldwide по адресу www.ibm.com/planetwide

Код изделия: CT17VRU

Напечатано в Дании

(1P) P/N: CT17VRU

SH43-0200-00

Spine information:

IBM® DB2 Universal Database™

Руководство администратора: Планирование

Версия 8