

IBM® DB2® Universal Database



# Руководство администратора: Планирование

*Версия 7*



IBM® DB2® Universal Database



# Руководство администратора: Планирование

*Версия 7*

Перед тем как использовать данный документ и продукт, описанный в нем, прочтите общие сведения под заголовком “Приложение F. Замечания” на стр. 301.

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

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

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

© Copyright International Business Machines Corporation 1993, 2001. Все права защищены.

---

# Содержание

Об этой книге . . . . .	vii
Для кого предназначена эта книга . . . . .	viii
Структура этой книги . . . . .	viii
Краткий обзор других томов Руководства администратора . . . . .	ix
Руководство администратора: Реализация . . . . .	ix
Руководство администратора: производительность . . . . .	xi

---

## Часть 1. Мир DB2 Universal Database . . . . . 1

### Глава 1. Управление DB2 Universal Database . . . . . 3

---

## Часть 2. Концепции баз данных . . . 5

### Глава 2. Основные понятия реляционных баз данных . . . . . 7

Обзор объектов базы данных . . . . .	7
Экземпляры . . . . .	8
Базы данных . . . . .	9
Группы узлов . . . . .	9
Таблицы . . . . .	9
Производные таблицы . . . . .	9
Индекс . . . . .	10
Схемы . . . . .	11
Таблицы системного каталога . . . . .	12
Обзор объектов хранения . . . . .	12
Табличные пространства . . . . .	12
Контейнеры . . . . .	16
Пул буферов . . . . .	17
Обзор системных объектов . . . . .	18
Параметры конфигурации . . . . .	18
Логические правила для данных . . . . .	20
Восстановление базы данных . . . . .	24
Реорганизация таблиц в базе данных . . . . .	25
Обзор защиты DB2 . . . . .	26
Аутентификация . . . . .	26
Авторизация . . . . .	27
Обзор аутентификации и авторизации для баз данных объединения . . . . .	28

### Глава 3. Системы объединения . . . . . 31

Включение поддержки системы объединения . . . . .	34
---	----

### Глава 4. Параллельные системы баз данных . . . . . 35

Группы узлов и разделение данных . . . . .	36
Типы параллелизма . . . . .	37
Параллелизм ввода/вывода . . . . .	38
Параллелизм запросов . . . . .	38
Параллелизм утилит . . . . .	41
Аппаратные среды . . . . .	41
Один раздел с одним процессором . . . . .	42
Один раздел с несколькими процессорами . . . . .	43
Многораздельные конфигурации . . . . .	45
Сводка по вариантам параллелизма для каждой аппаратной среды . . . . .	49

### Глава 5. О хранилищах данных . . . . . 51

Что такое хранилище данных? . . . . .	51
Тематические области . . . . .	52
Источники хранилища . . . . .	52
Потребители хранилища . . . . .	52
Агенты хранилища и узлы агентов . . . . .	52
Шаги и процессы . . . . .	53
Задачи хранилищ . . . . .	55

### Глава 6. О модуле Spatial Extender . . . . . 57

Назначение Spatial Extender . . . . .	57
Данные о географических объектах . . . . .	58
Как данные представляют географические объекты . . . . .	58
Характер пространственных данных . . . . .	59
Откуда берутся пространственные данные . . . . .	60

---

## Часть 3. Проектирование баз данных . . . . . 63

### Глава 7. Логическое проектирование базы данных . . . . . 65

Решите, какие данные записывать в базу данных . . . . .	65
Определите таблицы для каждого типа отношений . . . . .	67

Отношения один-с-многими и многие-с-одним . . . . .	67
Связи Многие-с-многими . . . . .	68
Отношение один-с-одним . . . . .	69
Дайте определения столбцов для всех таблиц	69
Выберите один или несколько столбцов в качестве первичного ключа . . . . .	72
Выберите ключевые столбцы . . . . .	73
Определите столбцы идентификации . . . . .	74
Обеспечьте, чтобы совпадающие значения представляли один объект . . . . .	75
Рассмотрите возможность нормализации таблиц . . . . .	76
Первая нормальная форма . . . . .	77
Вторая нормальная форма . . . . .	77
Третья нормальная форма . . . . .	79
Четвертая нормальная форма . . . . .	81
Планирование введения ограничений . . . . .	82
Ограничения уникальности . . . . .	82
Реляционная целостность . . . . .	83
Проверочные ограничения таблицы . . . . .	88
Триггеры . . . . .	88
Другие особенности проектирования базы данных . . . . .	89
<b>Глава 8. Физическая структура базы данных . . . . .</b>	<b>91</b>
Каталоги баз данных . . . . .	91
Файлы баз данных . . . . .	92
Оценка пространства, требуемого для таблиц	93
Таблицы системного каталога . . . . .	94
Данные пользовательских таблиц . . . . .	95
Данные длинных полей . . . . .	97
Данные больших объектов . . . . .	97
Индексное пространство . . . . .	98
Дополнительные требования к пространству	101
Пространство для файлов журнала . . . . .	101
Временное рабочее пространство . . . . .	102
Проектирование групп узлов . . . . .	102
Особенности проектирования групп узлов	104
Проектирование и выбор табличных пространств . . . . .	111
SMS - Пространство, управляемое системой . . . . .	115
Табличное пространство, управляемое базой данных . . . . .	119
Особенности структуры табличных пространств . . . . .	122
Особенности проектирования объединенных баз данных . . . . .	135

<b>Глава 9. Проектирование распределенных баз данных . . . . .</b>	<b>137</b>
Использование в транзакции одной базы данных . . . . .	138
Использование в транзакции нескольких баз данных . . . . .	139
Изменение одной базы данных . . . . .	139
Изменение нескольких баз данных . . . . .	140
Другие особенности конфигурирования . . . . .	145
Прикладные программы хоста или AS/400, обращающиеся к серверу DB2 Universal Database в локальной сети при многоузловом изменении . . . . .	146
Процесс двухфазного принятия . . . . .	148
Восстановление после ошибок двухфазного принятия . . . . .	151
Ресинхронизация неоднозначных транзакций, если AUTORESTART=OFF . . . . .	153
<b>Глава 10. Планирование использования менеджеров транзакций . . . . .</b>	<b>155</b>
Модель распределенной обработки транзакций X/Open . . . . .	156
Прикладная программа . . . . .	156
Менеджер транзакций . . . . .	158
Менеджеры ресурсов . . . . .	159
Задание базы данных в качестве менеджера ресурсов . . . . .	160
Использование строк xa_open и xa_close	160
Новый формат строки xa_open для DB2	
Версии 7 . . . . .	160
Значения параметров TPM и TP_MON_NAME . . . . .	162
Формат строки xa_open для прежних версий DB2 . . . . .	166
Изменения на серверах баз данных хоста или AS/400 . . . . .	166
Особенности соединения с базой данных	166
Принятие эвристических решений . . . . .	167
Особенности защиты . . . . .	170
Особенности конфигурирования . . . . .	171
Поддерживаемая функция XA . . . . .	172
Диагностика ошибок интерфейса XA . . . . .	174
Конфигурирование менеджеров транзакций XA для использования DB2 UDB . . . . .	175
Конфигурирование IBM TXSeries CICS . . . . .	175
Конфигурирование IBM TXSeries Encina	176
Конфигурирование BEA Tuxedo . . . . .	178
Конфигурирование сервера Microsoft Transaction Server . . . . .	180

<b>Глава 11. Высокая доступность и восстановление при отказах. Введение</b>	<b>187</b>
Высокая доступность	187
Высокая доступность с помощью оперативного отделения зеркальной копии и поддержки приостановленного ввода/вывода	190
Создание клона базы данных	191
Использование отдельной зеркальной копии в качестве резервной базы данных	191
Использование отдельной зеркальной копии в качестве резервной копии	192

## **Часть 4. Приложения . . . . . 193**

<b>Приложение А. Правила именования</b>	<b>195</b>
Общие правила именования	195
Правила именования объектов	195
Дополнительная информация об именах схем	197
Дополнительная информация о паролях	198
Использование идентификаторов с ограничителями в качестве имен объектов	199
Сохранение регистрозависимых значений в системе объединения	200

### **Приложение В. Планирование перенастройки баз данных . . . . . 203**

Особенности перенастройки	203
Ограничения перенастройки	204
Защита и авторизация	204
Требования к памяти	204
Несовместимость выпусков	205
Перенастройка базы данных	205

### **Приложение С. Несовместимость выпусков . . . . . 209**

Планируемые несовместимости в DB2 Universal Database	210
Производные таблицы только для чтения в будущей версии DB2 Universal Database	210
PK_COLNAMES и FK_COLNAMES в будущей версии DB2 Universal Database	210
Столбец COLNAMES в будущей версии DB2 Universal Database	211
Несовместимости DB2 Universal Database	
Версии 7	211
Прикладное программирование	211
SQL	213
Утилиты и инструменты	215

Возможности соединений и сосуществование	216
Несовместимости DB2 Universal Database	
Версии 6	216
Производные таблицы системных каталогов	216
Прикладное программирование	223
SQL	227
Защита и настройка баз данных	229
Утилиты и инструменты	230
Возможности соединений и сосуществование	231
Параметры конфигурации	231

### **Приложение D. Поддержка национальных языков (NLS) . . . . . 233**

Национальные версии	233
Поддержка кода страны/региона и кодовой страницы:	233
Национальные настройки для сервера администратора DB2	249
Поддержка переведенных национальных версий на платформах на основе UNIX	249
Наборы файлов Центра управления и документации	252
Определение значений кодовых страниц	253
Наборы символов	254
Набор символов для идентификаторов	255
Кодирование операторов SQL	256
Поддержка CCSID с двумя направлениями письма	256
Последовательность сортировки	260
Значения даты и времени	263
Поддержка Unicode в DB2 UDB	269
Введение	269
Реализация Unicode в DB2 UDB	271

### **Приложение E. Использование библиотеки DB2 . . . . . 281**

Файлы PDF и печатные книги DB2	281
Информация DB2	281
Печать книг PDF	291
Заказ печатных копий	292
Электронная документация DB2	293
Обращение к электронной справке	293
Просмотр информации на экране	295
Использование мастеров DB2	298
Установка сервера документации	299
Поиск электронной информации	300

<b>Приложение F. Замечания . . . . .</b>	<b>301</b>
Товарные знаки . . . . .	304
<b>Индекс . . . . .</b>	<b>307</b>

<b>Как связаться с IBM . . . . .</b>	<b>313</b>
Информация о продукте. . . . .	313



---

## Об этой книге

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

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

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

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

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

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

---

## Для кого предназначена эта книга

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

---

## Структура этой книги

Эта книга содержит сведения на следующие основные темы:

### В мире DB2 Universal Database

- В разделе Глава 1. Управление DB2 Universal Database содержатся предварительные сведения о DB2 Universal Database и ее обзор.

### Концепции баз данных

- В разделе Глава 2. Основные понятия реляционных баз данных приводится обзор объектов баз данных, включая объекты хранения и системные объекты.
- В разделе Глава 3. Системы объединения обсуждаются системы объединения - системы управления базами данных (СУБД), которые дают возможность программам и пользователям выполнять операторы SQL, обращающиеся (в одном операторе) к нескольким СУБД или нескольким базам данных.
- В разделе Глава 4. Параллельные системы баз данных излагаются начальные сведения о типах параллелизма, доступных при работе с DB2.
- В разделе Глава 5. О хранилищах данных дается обзор работы с хранилищами данных и возникающих при этом задач.
- В разделе Глава 6. О модуле Spatial Extender дается понятие о модуле Spatial Extender, объясняется его назначение и обсуждаются обрабатываемые с его помощью данные.

### Проектирование баз данных

- В разделе Глава 7. Логическое проектирование базы данных обсуждаются концепции логической структуры базы данных и даются указания по разработке баз данных.
- В разделе Глава 8. Физическая структура базы данных содержатся указания по разработке физической структуры базы данных, в том числе по вопросам хранения данных.
- В разделе Глава 9. Проектирование распределенных баз данных обсуждается, как обращаться к нескольким базам данных в ходе одной транзакции.
- В разделе Глава 10. Планирование использования менеджеров транзакций обсуждается, как использовать ваши базы данных в среде обработки распределенных транзакций, например, CICS.
- Раздел Глава 11. Высокая доступность и восстановление при отказах. Введение содержит обзор поддержки высокой доступности восстановления после отказов, обеспечиваемой DB2.

### Приложения

- В разделе Приложение А. Правила именования приводятся правила именования баз данных и объектов.
- В разделе Приложение В. Планирование перенастройки баз данных приводится информация о перенастройке баз данных в Версию 7.
- В разделе Приложение С. Несовместимость выпусков рассматриваются несовместимости, возникающие при переходе от выпуска к выпуску, вплоть до Версии 7.
- В разделе Приложение D. Поддержка национальных языков (NLS) дается понятие о поддержке национальных языков в DB2, включая информацию о странах, языках и кодовых страницах.
- В разделе Приложение Е. Использование библиотеки DB2 приводится информация о структуре библиотеки DB2, включая мастера, электронную справку, сообщения и книги.

---

## Краткий обзор других томов Руководства администратора

### Руководство администратора: Реализация

Книга *Administration Guide: Implementation* посвящена реализации вашего проекта базы данных. Отдельные главы и приложения из этого тома кратко описаны здесь:

#### Управление при помощи Центра управления

- Раздел "Управление DB2 при помощи инструментов GUI" описывает инструменты графического пользовательского интерфейса (GUI) для управления базой данных.

#### Реализация вашего проекта

- В разделе "Перед созданием базы данных" описаны предварительные требования для создания базы данных.
- В разделе "Создание базы данных" описаны 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.

### **Перемещение данных**

- Раздел "Утилиты для перемещения данных" - это краткое введение в различные способы перемещения данных; он адресуется к книге *Data Movement Utilities Guide and Reference*.

### **Восстановление**

- "Восстановление базы данных" - краткое введение в основные понятия резервного копирования и восстановления базы данных и повтора транзакций. Более подробные сведения можно найти в книге *Справочное руководство по восстановлению данных и высокой доступности*.

### **Приложения**

- В разделе "Использование служб каталога распределенной вычислительной среды (DCE)" обсуждается информация о том, как пользоваться службами каталога DCE.
- В разделе "Обработчик пользователя для восстановления базы данных" обсуждается, каким можно использовать обработчики пользователя с файлами журнала базы данных, и описываются некоторые примеры обработчиков пользователя.
- В разделе "Выдача команд нескольким серверам разделов баз данных" обсуждается использование сценариев оболочки *db2\_all* и *rah* для отправки команд всем разделам в среде многораздельных баз данных.
- 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 NT" 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 NT или Windows 2000" 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 "Высокоскоростная связь между узлами" описывается, как разрешить использование Virtual Interface Architecture с DB2 Universal Database.
- В разделе "Использование служб каталога протокола LDAP" приводится информация о том, как пользоваться службами каталога LDAP.
- 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.

## **Руководство администратора: производительность**

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

### **Вводные замечания по производительности**

- Раздел "Элементы производительности" знакомит с идеями и особенностями управления производительностью DB2 UDB и ее повышения.
- В разделе "Обзор архитектуры и процессов обработки" излагаются основы архитектуры и процессов DB2 Universal Database.

### **Настройка производительности прикладных программ**

- В разделе "Особенности прикладных программ" описываются некоторые методы повышения производительности базы данных при разработке ваших прикладных программ.
- В разделе "Особенности среды" описываются некоторые методы повышения производительности базы данных при задании параметров среды вашей базы данных.
- В разделе "Статистика системного каталога" описывается, как можно собрать статистику о ваших данных и использовать ее для обеспечения оптимальной производительности.
- В разделе "Основные сведения о компиляторе SQL" описывается, что происходит с оператором SQL при его компиляции с помощью компилятора SQL.

- В разделе "Средства объяснения SQL" описываются средства объяснения, позволяющие исследовать варианты доступа к вашим данным, выбранные компилятором SQL.

### **Настройка и конфигурирование вашей системы**

- В разделе "Производительность работы" дается обзор использования памяти менеджером баз данных и описываются другие особенности, влияющие на производительность во время выполнения.
- В разделе "Использование утилиты ограничения ресурсов" излагаются начальные сведения об использовании утилиты ограничения ресурсов для управления некоторыми аспектами работы с базой данных.
- В разделе "Масштабирование вашей конфигурации" описаны 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" дается обзор вопросов и способов измерения производительности.
- В главе "Конфигурирование DB2" обсуждаются файлы конфигурации менеджера баз данных и баз данных и значения параметров конфигурации.

### **Приложения**

- В приложении "Реестр и переменные среды DB2" описаны значения реестра профиля и переменных среды.
- В приложении "Таблицы и определения объяснения" описаны таблицы, используемые средствами объяснения DB2, и их создание.
- В приложении "Инструменты объяснения SQL" описаны инструменты объяснения DB2: db2expln и dypexpln.
- В приложении "db2exfmt – инструмент форматирования таблиц объяснения" описано использование этого инструмента объяснения DB2 для форматирования данных таблиц объяснения.

---

# Часть 1. Мир DB2 Universal Database





---

# Глава 1. Управление DB2 Universal Database

DB2 обеспечивает гибкость выбора для работы аппаратных конфигураций в широком диапазоне. Она позволяет вам выбрать, как наилучшим образом выполнить аппаратуры и прикладных программ при помощи определенной конфигурацией продукта DB2.

DB2 поддерживается также среды баз данных различных уровней сложности, и для каждой среды есть свои определенные особенности и свои задачи. Они обсуждаются подробно в руководстве *Administration Guide* и других книгах библиотеки DB2 (смотрите приложение “Приложение Е. Использование библиотеки DB2” на стр. 281). В некоторых случаях целые разделы этих книг относятся только к определенным средам. Узнать, с какими главами этого и других томов руководства *Administration Guide* (*Administration Guide: Implementation* и *Administration Guide: Performance*) вам стоит ознакомиться, можно из предисловия к этой книге (“Об этой книге”).

Если вы мало знакомы с системами управления реляционными базами данных (relational database management systems, RDBMS) или с DB2, вам будет полезен раздел “Основные понятия реляционных баз данных”. Если вы знакомы с этими понятиями или не нуждаетесь в таком обзоре, можно пропустить этот раздел и перейти прямо к разделам, подробно описывающим более конкретные темы:

- Системы объединения. В этом разделе описаны системы управления базами данных (СУБД), которые дают возможность программам и пользователям выполнять операторы SQL, обращающиеся (в одном операторе) к нескольким СУБД или базам данных.
- Системы параллельных баз данных. Этот раздел содержит введение в типы параллелизма, доступные в системе DB2. Компоненты задачи, например запрос базы данных, могут запускаться параллельно, что существенно улучшает производительность.
- Распределенная обработка транзакций. В этом разделе обсуждается, как обращаться к нескольким базам данных в одной транзакции и как использовать базы данных в среде распределенной обработки транзакций.

DB2 позволяет удовлетворить ваши специальные нужды в области управления информацией, такие как:

- *Репликация*, позволяющая регулярно копировать информацию в несколько удаленных баз данных. Если вам нужно автоматически копировать изменения в главной базе данных в остальные базы, вы можете воспользоваться возможностями репликации в DB2, чтобы задать, какую информацию нужно копировать, в какие таблицы баз данных и как часто. Если вы хотите использовать возможности репликации DB2, обратитесь к книге *Replication*

*Guide and Reference*. В ней описаны понятия репликации информации в DB2, планирование, конфигурирование среды репликации и управление ей.

- *Хранилище данных*, в котором вы можете хранить "информативные данные" - данные, извлеченные из рабочих данных и преобразованные для принятия решений конечными пользователями. Например, средствами хранилища данных можно скопировать данные продаж из рабочей базы данных, выполнить вычисления для составления по этой информации сводных данных и записать полученные сводные данные в другую базу данных, отдельную от рабочей. Можно строить запросы к этой отдельной базе данных (*хранилищу*), не затрагивая рабочих баз данных. Подробную информацию о хранилищах данных смотрите в руководстве *Data Warehouse Center Administration Guide*.
- *Географическая информационная система* (ГИС), которую можно создать с помощью модуля Spatial Extender. ГИС - это совокупность объектов, информации и прикладных программ, позволяющая генерировать и анализировать географическую информацию о географических объектах. В Spatial Extender географический объект может быть представлен строкой в таблице или производной таблице или частью такой строки. Подробную информацию об использовании Spatial Extender смотрите в книге *Spatial Extender User's Guide and Reference*.

В руководстве *Administration Guide: Planning* также описано проектирование баз данных и особенности проектирования логических и физических баз данных для DB2. Изложены и другие вопросы планирования, такие как планирование перенастройки баз данных, описание несовместимостей, которые могли бы затронуть ваши прикладные программы (*несовместимость* - это часть DB2 Universal Database, которая работает иначе, чем в предыдущем выпуске DB2; в случае использования в существующей прикладной программе, несовместимость будет приводить к неожиданным результатам, вынуждать изменение прикладной программы или снижать производительность) и использование поддержки национальных языков.

В книге *Administration Guide: Implementation* описываются подробности реализации проекта ваших баз данных. Изложены такие темы, как создание и изменение баз данных, защита баз данных и управление системой DB2 с помощью Центра управления, графический пользовательский интерфейс DB2.

В книге *Administration Guide: Performance* описываются темы и вопросы, касающиеся установки, проверки и повышения производительности вашей программы и самой системы DB2.

---

## **Часть 2. Концепции баз данных**



---

## Глава 2. Основные понятия реляционных баз данных

В этой главе рассматриваются следующие темы:

- “Обзор объектов базы данных”
- “Обзор объектов хранения” на стр. 12
- “Обзор системных объектов” на стр. 18
- “Логические правила для данных” на стр. 20
- “Восстановление базы данных” на стр. 24
- “Реорганизация таблиц в базе данных” на стр. 25
- “Обзор защиты DB2” на стр. 26

---

### Обзор объектов базы данных

В этом разделе дается обзор следующих ключевых объектов базы данных:

- Экземпляры
- Базы данных
- Группы узлов
- Таблицы
- Производные таблицы
- Индексы
- Схемы
- Таблицы системного каталога

На рис. 1 на стр. 8 показаны отношения между некоторыми из этих объектов. Здесь также показано, что таблицы, индексы и длинные данные хранятся в табличных пространствах.

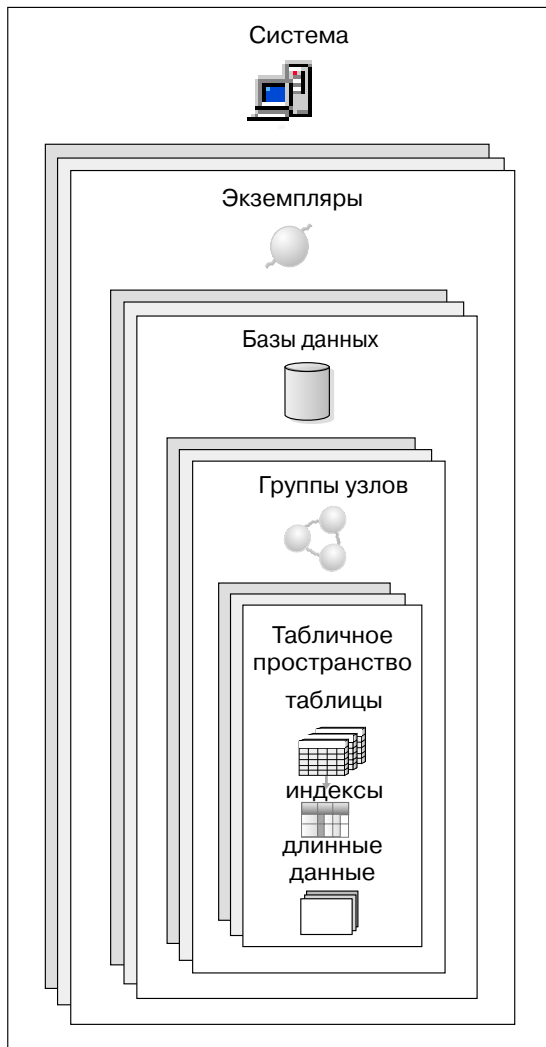


Рисунок 1. Отношения между некоторыми объектами баз данных

## Экземпляры

*Экземпляр* (другое название - *менеджером баз данных*) - это код DB2, предназначен для работы с данными. Он управляет возможными действиями над данными и назначенными для него системными ресурсами. Каждый экземпляр является полнофункциональной средой. Он содержит все разделы баз данных для данной параллельной системы баз данных (смотрите “Глава 4. Параллельные системы баз данных” на стр. 35). 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 узлов

*Группа узлов* - это набор из одного или нескольких разделов базы данных. При создании таблиц для базы данных сначала вы создаете группу узлов, где будут храниться табличные пространства, а затем создаете табличное пространство, где будут храниться таблицы. Дополнительную информацию о группах узлов смотрите в разделе "Группы узлов и разделение данных" на стр. 36. "Глава 4. Параллельные системы баз данных" на стр. 35 содержит определение раздела базы данных. Дополнительную информацию о табличных пространствах смотрите в разделе "Табличные пространства" на стр. 12.

## 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. Дополнительную информацию о табличных пространствах смотрите в разделе "Табличные пространства" на стр. 12. 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.

Доступ к табличным данным осуществляется с помощью SQL (Structured Query Language - язык структурированных запросов, смотрите справочник *SQL Reference*), стандартизированного языка для определения данных и работы с ними в *реляционных базах данных*. *Запрос* используется 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. 2 показаны отношения между таблицами и производными таблицами.

### База данных

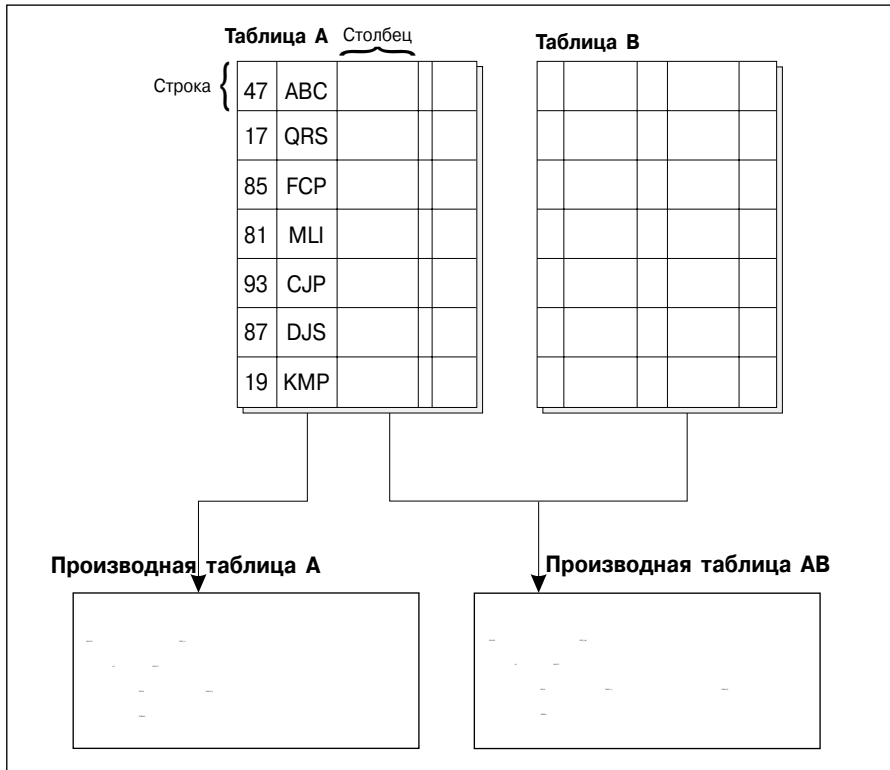


Рисунок 2. Отношение между таблицами и производными таблицами

## Индекс

*Индекс* - это набор ключей, каждый из которых указывает на строки в таблице. Например, таблица A на рис. 3 на стр. 11 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. Более подробно ключи и индексы описаны в разделе “Логические правила для данных” на стр. 20.

На рис. 3 показано отношение между индексом и таблицей.

База данных

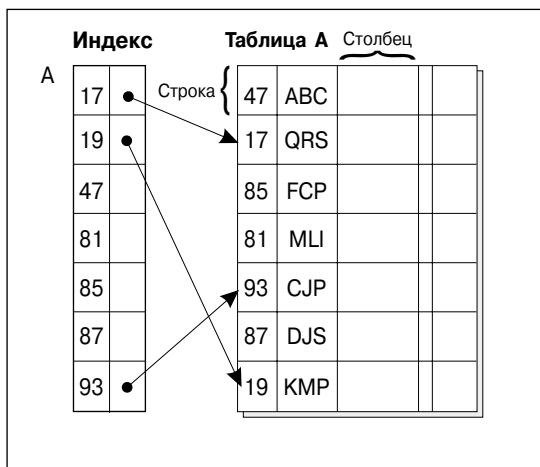


Рисунок 3. Отношение между индексом и таблицей

## 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.

---

## Обзор объектов хранения

Следующие объекты базы данных позволяют вам определить, как данные будут храниться в системе и как можно повысить производительность (связанную с доступом к данным):

- Табличное пространство
- Контейнер
- Пул буферов

## 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:

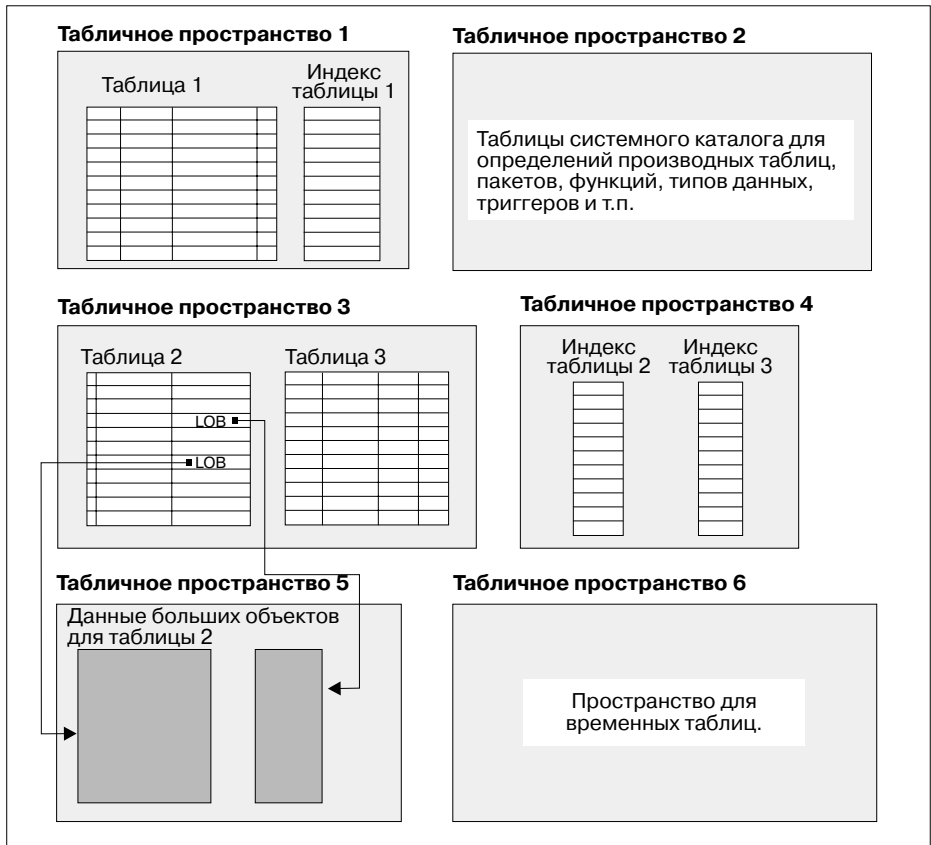


Рисунок 4. Табличные пространства

Табличные пространства размещаются в группах узлов (смотрите раздел “Группы узлов” на стр. 9). Определения и атрибуты табличных пространств записываются в системный каталог базы данных (смотрите раздел “Таблицы системного каталога” на стр. 12).

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.

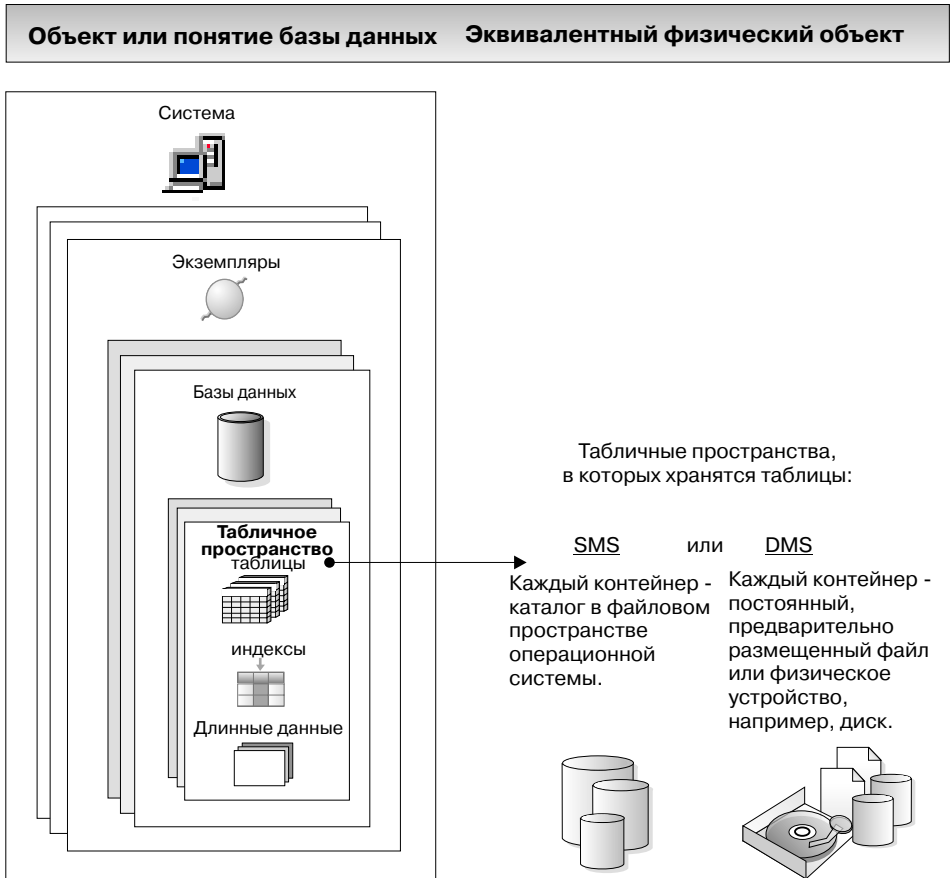


Рисунок 5. Табличные пространства и таблицы

На рис. 6 на стр. 15 показаны три типа табличных пространств: *обычное*, *временное* и *длинное*.

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

Таблицы, содержащие данные длинных полей или данные длинных объектов, например, объектов мультимедиа, размещаются в длинных 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.

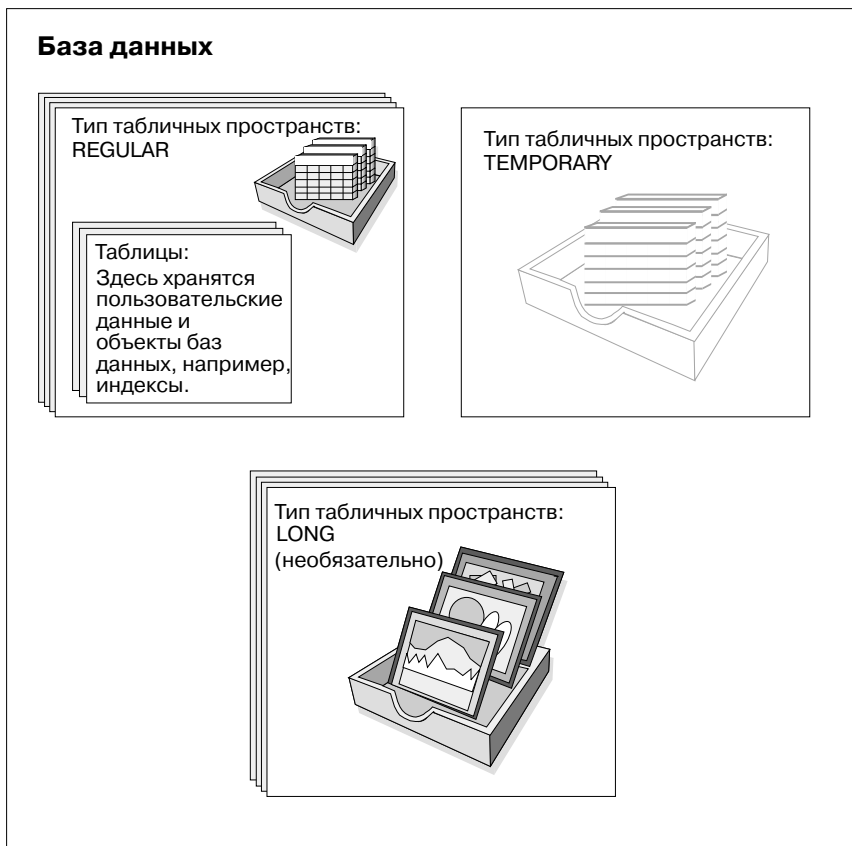


Рисунок 6. Три типа табличных пространств

## 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 показано отношение между таблицами и табличным пространством в базе данных, а также связанные с ними контейнеры и диски.

### База данных

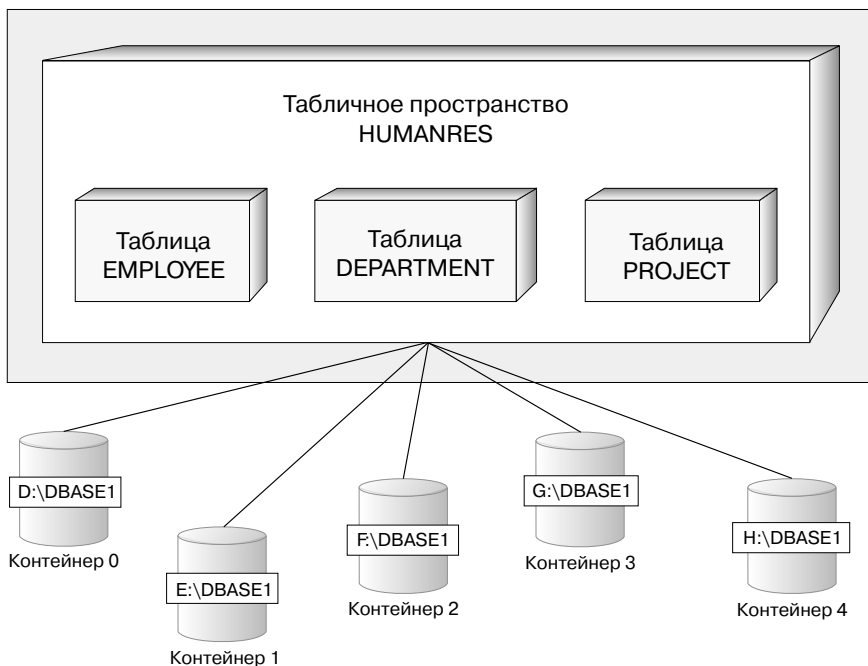


Рисунок 7. Табличные пространства и таблицы в базе данных

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 экстенда*.

## **Пул 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 на стр. 18 показана взаимосвязь между пулом буферов и контейнерами.

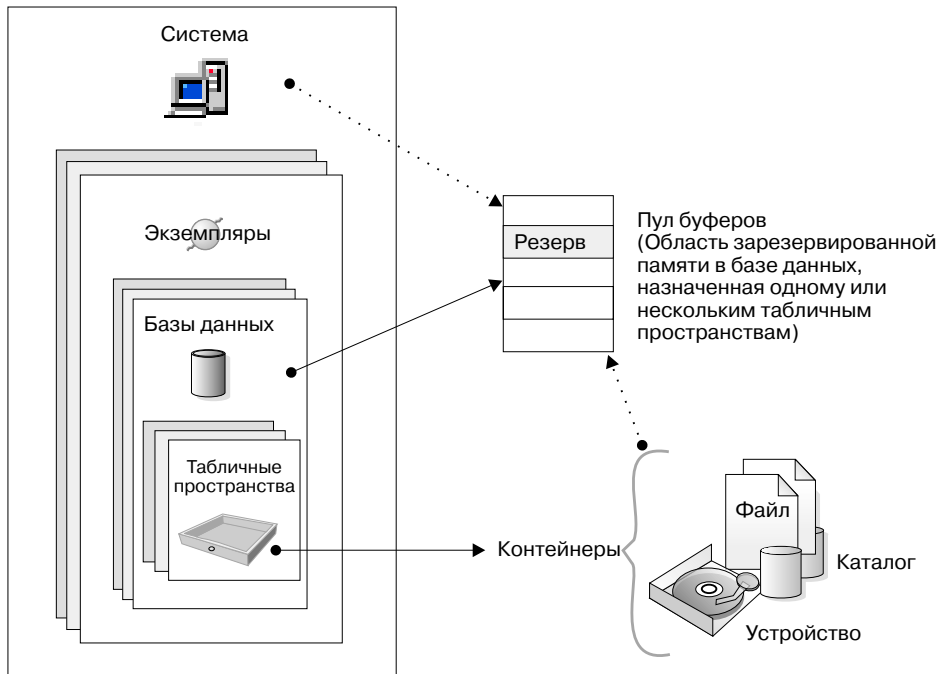


Рисунок 8. Пул буферов и контейнеры

## Обзор системных объектов

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

### Параметры конфигурации

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 и файл конфигурации базы данных для каждой отдельной базы данных (смотрите рис. 9 на стр. 20).



*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.

*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.

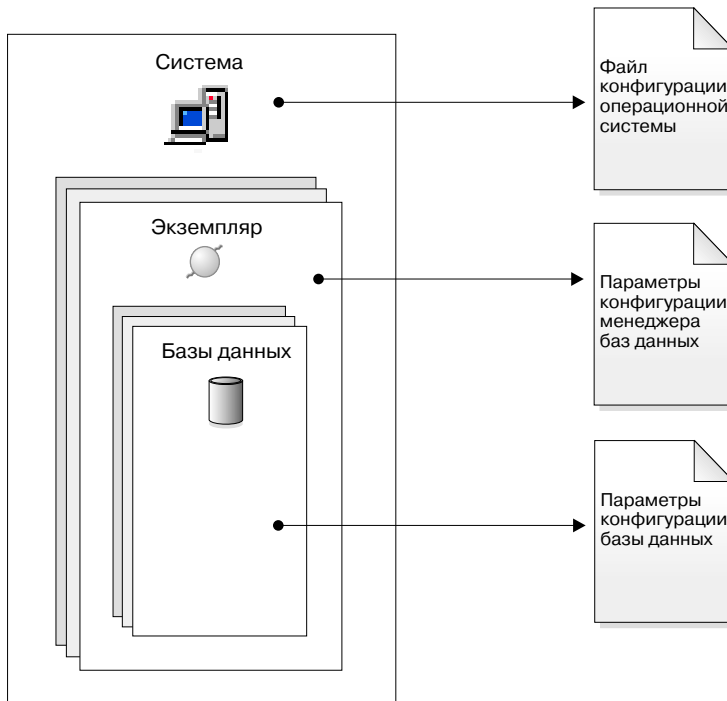


Рисунок 9. Файлы параметров конфигурации

## Логические правила для данных

В любой сфере деятельности данные чаще всего должны подчиняться определенным ограничениям или правилам. Например, номера сотрудников должны быть уникальны. DB2 обеспечивает способ задания таких правил, называемый *ограничениями*.

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.

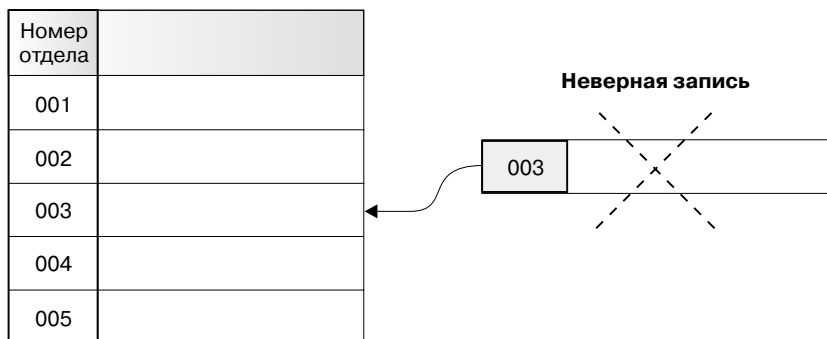


Рисунок 10. Ограничения уникальности предотвращают повторение данных

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.

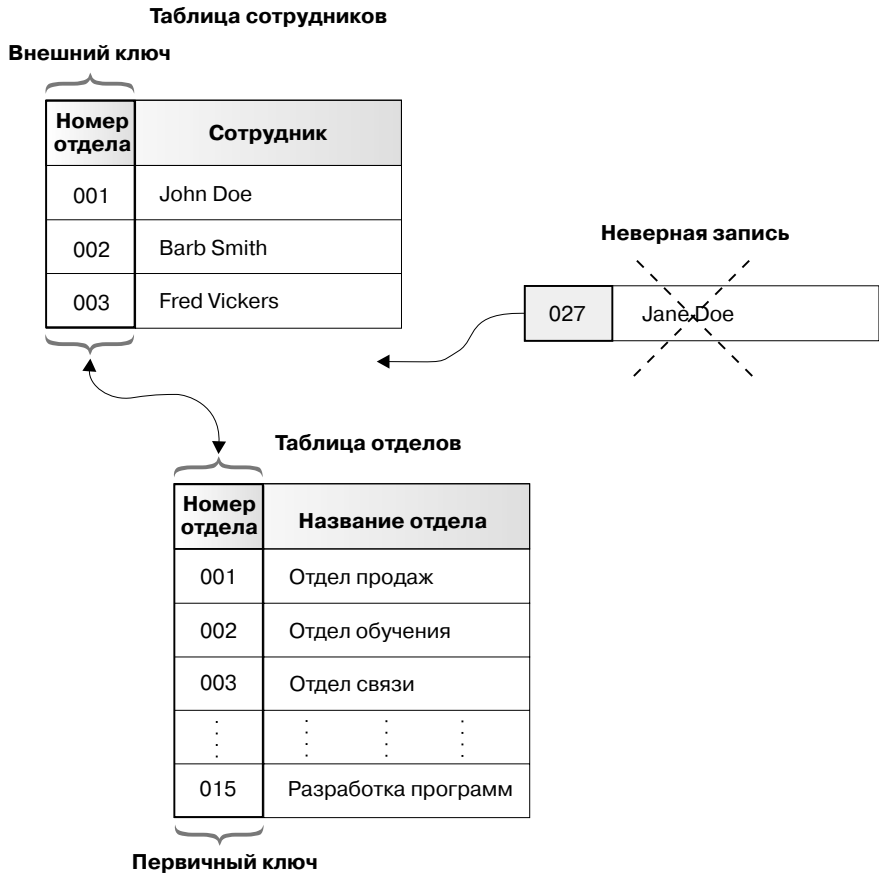


Рисунок 11. Ограничения внешнего и первичного ключей определяют отношения и защищают данные

### 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 ограничением триггеры - более сложное и потенциально более мощное средство. Они определяют набор действий, выполняемых в сочетании с операцией

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. Дополнительную информацию о триггерах смотрите в книге *Application Development Guide*.

---

## Восстановление базы данных

База данных может оказаться непригодной для использования из-за ошибки аппаратных средств, программного обеспечения или общей ошибки; при различных сценариях ошибок могут потребоваться разные действия по восстановлению. Для защиты базы данных от возможных сбоев должна существовать продуманная стратегия.

Подробную информацию о резервном копировании, восстановлении и разработке стратегии копирования и восстановления смотрите в книге *Справочное руководство по восстановлению данных и высокой доступности*.

Идея резервного копирования базы данных - та же, что и для любых других данных: копия данных снимается и 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 база данных повреждена или разрушена, вы можете воссоздать ее из резервной копии. Воссоздание базы данных называется восстановлением.

Если авария базы данных происходит во время выполнения транзакции, при перезапуске базы происходит процесс, называемый восстановлением после отказа. *Восстановление после отказа* - это процесс, при котором база данных возвращается в согласованное и пригодное к использованию состояние. Это достигается откатом незавершенных транзакций и завершением принятых транзакций, которые еще оставались в памяти к моменту отказа.

Если база данных повреждена или разрушена, есть два способа восстановить ее - восстановление версии и восстановление с повтором транзакций.

*Восстановление версии* - это восстановление предыдущей версии базы данных с использованием образа, созданного при резервном копировании. Резервная копия базы данных позволяет восстановить базу данных до состояния на момент создания данной резервной копии. Однако все единицы работы с момента резервного копирования до момента ошибки теряются.

Если вы хотите иметь возможность восстановления до точки после момента снятия резервной копии, надо использовать *восстановление с повтором транзакций*. Для использования метода восстановления с повтором транзакций у вас должны быть готовая резервная копия базы данных и архивированные журналы. (Это достигается заданием параметра конфигурации базы данных *logretain* или *userexit*.)

Каждая база данных включает журналы восстановления, которые используются для восстановления информации при ошибках программ или системы. В сочетании с резервными копиями базы данных их можно использовать для восстановления базы данных до момента, когда случилась ошибка. Файлы журналов создаются автоматически при создании базы данных. Эти файлы нельзя изменять непосредственно.

Еще один важный для восстановления объект - файл хронологии восстановлений. Файл хронологии восстановлений содержит сводную информацию о резервном копировании, которую можно использовать, когда нужно восстановить всю базу данных или ее часть до заданного момента времени. Он используется для отслеживания событий, связанных с восстановлением, например, резервного копирования и операций восстановления и загрузки.

**Примечание:** Вся информация о резервном копировании и восстановлении, а также соответствующая информация из справочников Command Reference и API Reference, собрана в книге *Справочное руководство по восстановлению данных и высокой доступности*. Книга *Справочное руководство по восстановлению данных и высокой доступности* - это основной источник информации по резервному копированию и восстановлению.

---

## Реорганизация таблиц в базе данных

После большого числа изменений таблица может стать фрагментированной, что приведет к ухудшению производительности. Если после сбора статистики производительность заметно не улучшится, может помочь реорганизация данных таблицы. При реорганизации таблицы данные перестраиваются согласно заданному индексу в физическую последовательность, а свободное пространство, образовавшееся при фрагментации, удаляется. Такая реорганизация может повысить скорость доступа к данным, тем самым улучшив производительность.

Перед реорганизацией таблицы рекомендуется вызвать команду REORGCHK и собрать статистику по данной таблице. Эта команда поможет определить, нужна ли реорганизация таблицы. Информацию о команде REORGCHK смотрите в книге *Command Reference*.

---

## Обзор защиты DB2

Для защиты данных и ресурсов, связанных с сервером баз данных, в DB2 используется 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.

### 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 операционных системах на основе UNIX такое средство защиты есть в самой операционной системе. DCE Security Services - самостоятельный продукт, предоставляющий средство защиты для распределенной среды. В операционных системах Windows 9x и Windows 3.1 средств защиты нет.

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 при помощи 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. В DCE Security Services *ID авторизации* DB2 содержится в реестре DCE и извлекается оттуда после успешного завершения аутентификации.
- 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 получит список максимум из 64 групп. Если пользователь входит больше, чем в 64 группы, в список групп DB2 будут добавлены только первые 64 группы, которые будут отображены на допустимые имена авторизации DB2. Никакие ошибки не возвращаются, а все группы, начиная с 65, игнорируются DB2.



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.

Администраторы DB2 могут позволять другим пользователям изменять пароли в системах AIX и Windows NT EEE через переменную реестра профиля DB2CHGPWD\_EEE. Значение по умолчанию для этой переменной NOT SET (отключено). DB2CHGPWD\_EEE принимает стандартные логические значения, используемые другими переменными профиля DB2.

Администратор DB2 отвечает за централизованную поддержку паролей для всех узлов и использует для этого либо Windows NT Domain Controller в Windows NT, либо NIS в AIX.

**Примечание:** Если пароли не поддерживаются централизованно, включение переменной DB2CHGPWD\_EEE может привести к несогласованности паролей по узлам. В таком случае при использовании возможности "изменить пароль" пароль будет изменен только на узле, с которым вы соединяетесь.

DB2 UDB в AIX может регистрировать попытки ввода неверных паролей средствами операционной системы и определять, когда клиент превысит число разрешаемых попыток регистрации, заданное параметром LOGINRETRIES.

Дополнительную информацию о системной проверке правильности ввода (это особенно важно, когда к базе данных обращаются удаленные клиенты), смотрите в разделе "Выбор метода аутентификации для сервера" книги *Administration Guide: Implementation*.

## Авторизация

Авторизация - это процесс, при котором DB2 получает информацию о своем аутентифицированном пользователе с указанием операций, которые он может выполнять, и объектов, 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 имени тех группы, к которым он принадлежит, сравнивается 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 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.

Дополнительную информацию смотрите в разделе "Привилегия, полномочия и авторизация" книги *Administration Guide: Implementation*.

## **Обзор аутентификации и авторизации для баз данных объединения**

Так как система баз данных объединения DB2 может получать доступ к информации нескольких систем управления базами данных, для защиты данных могут потребоваться дополнительные шаги.

При планировании подхода к аутентификации учтите, что пользователи могут проходить проверки аутентификации не только на DB2, но и на источниках данных. В системе объединения аутентификация может проводиться на рабочих станциях клиентов DB2, на серверах DB2, на источниках данных (DB2, DB2 for OS/390, другие серверы DRDA, Oracle) или на DB2 (клиент или сервер DB2) в сочетании с источниками данных. Даже в средах DCE могут оказаться необходимыми определенные шаги, если источник данных требует ID

пользователя и пароль. Дополнительную информацию смотрите в разделе "Обработка аутентификации базы данных объединения" книги *Administration Guide: Implementation*.

Точно также в DB2 и на источниках данных пользователи должны проходить и проверку авторизации. Каждый источник данных (DB2, Oracle, DB2 for OS/390 и т.п.) поддерживает защиту объектов под своим собственным управлением. При выполнении пользователем операции с псевдонимом он обязан пройти проверку авторизации для таблицы или производной таблицы, к которой обращаются по этому псевдониму.



---

## Глава 3. Системы объединения

*Система баз данных объединения* или *система объединения* - это система управления базами данных (СУБД), которая обеспечивает для программ и пользователей, выдающих операторы SQL, возможность обращения к нескольким СУБД или к нескольким базам данных в одном операторе. Примером служит объединение между таблицами из двух различных баз данных DB2. Такой тип оператора называется *распределенным требованием*.

Система объединения DB2 Universal Database обеспечивает поддержку распределенных требований по базам данных и СУБД. Можно, например, выполнить операцию UNION между таблицей DB2 и представлением Oracle. К поддерживаемым СУБД относится DB2, продукты семейства DB2 (например, DB2 for OS/390 и DB2 for AS/400) и Oracle.

Система объединения DB2 обеспечивает *прозрачность положений* для объектов баз данных. Если информация (в таблицах и производных таблицах) перемещается, можно исправить ссылки (так называемые *псевдонимы*), не внося никаких изменений в сами программы, запрашивающие информацию. Кроме того, система объединения DB2 предоставляет возможность *работать* с СУБД, которые не полностью поддерживают диалект SQL DB2 или какие-либо возможности оптимизации. Операции, которые нельзя выполнить в таких СУБД (например, рекурсивный SQL), запускаются под DB2.

Системы объединения DB2 работают *полуавтономным* способом: запросы DB2, содержащие ссылки на объекты Oracle, могут передаваться на Oracle, когда программы Oracle обращаются к тому же серверу. Система объединения DB2 не монополизирует и не ограничивает доступ (помимо ограничений целостности и блокировок) к объектам Oracle и других СУБД.

Система объединения DB2 состоит из экземпляра DB2 UDB, базы данных, которая будет служить *базой данных объединения*, и одного или нескольких *источников данных*. База данных объединения содержит записи каталога, идентифицирующие источники данных и их характеристики. Источник данных включает в себя СУБД и данные. Программы обращаются к базе данных объединения, как и к любой другой базе данных DB2. Среда базы данных объединения показана на рис. 12 на стр. 32.

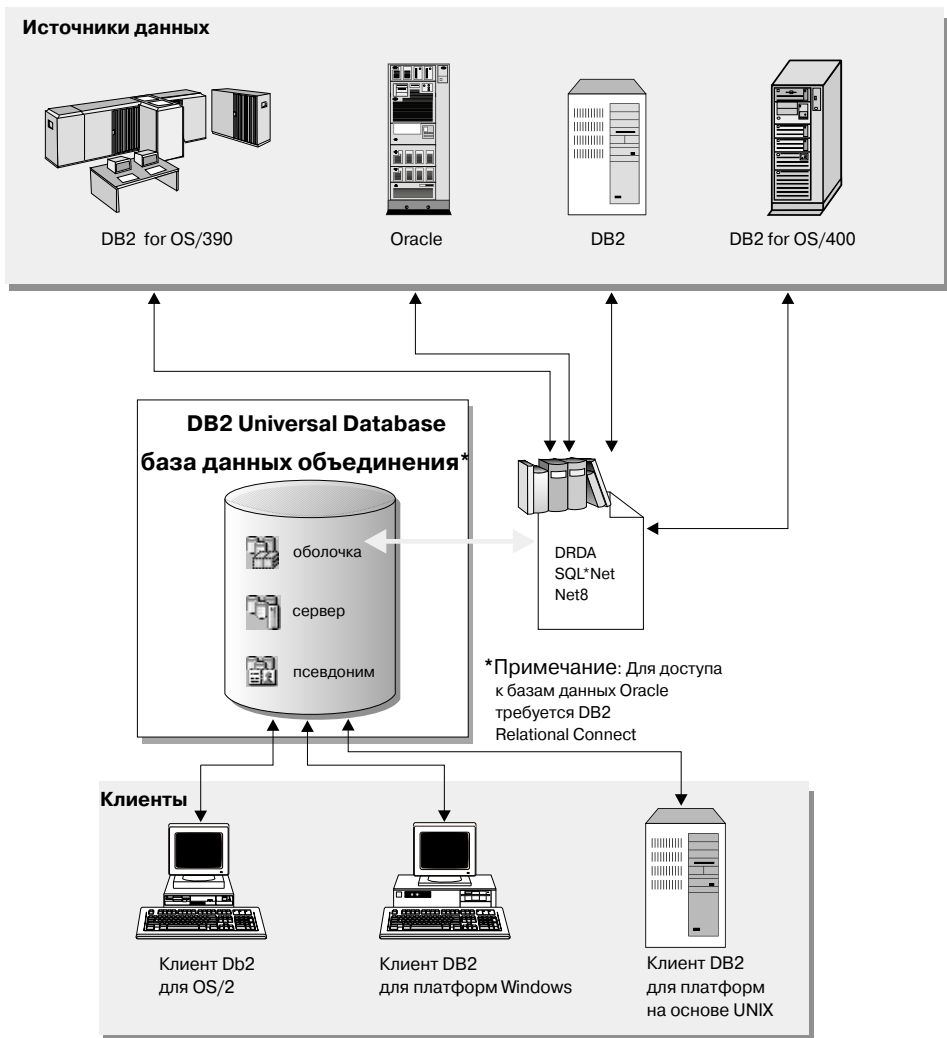


Рисунок 12. Система баз данных объединения

Записи каталога объединения DB2 содержат информацию об объектах источника данных: их имена, содержащуюся в них информацию и условия, на которых их можно использовать. Поскольку этот каталог DB2 хранит информацию об объектах из многих СУБД, он называется *глобальным каталогом*. В этом каталоге хранятся атрибуты объектов. Реальные СУБД, к которым обращаются, модули, используемые для связи с источником данных, и объекты данных СУБД (например, таблицы), к которым потребуется доступ, хранятся вне базы данных. (Одно исключение: база данных объединения может сама являться источником данных для системы объединения.) Объекты

объединения можно создать с помощью Центра управления или операторов SQL DDL. Необходимые объекты базы данных объединения:

### **Оболочки**

Идентифицируют модули (DLL, библиотеки и так далее), используемые для доступа к конкретному классу или категории источника данных.

### **Серверы**

Определяют источники данных. Данные сервера включают имя оболочки, имя сервера, тип сервера, версию сервера, информацию авторизации и опции сервера.

### **Псевдонимы**

Идентификаторы, хранимые в базе данных объединения - ссылки на конкретные объекты источника данных (таблицы, алиасы, производные таблицы). Программы используют псевдонимы в запросах так же, как имена таблиц и производных таблиц.

В зависимости от конкретных потребностей вы можете создавать дополнительные объекты:

- Отображения пользователей - для целей аутентификации
- Отображения типов данных - для настройки отношений между типом источника данных и типом DB2
- Отображения функций - для отображения локальных функций в функции источника данных
- Спецификации индексов - для повышения производительности.

Когда система объединения сконфигурирована, к информации источников данных можно обращаться так, как если бы она находилась в одной большой базе данных. Пользователи и программы посылают запросы на одну базу объединения, которая затем получает данные от систем семейства DB2 и Oracle по необходимости. Пользователи и программы задают в запросах псевдонимы; при помощи этих псевдонимов производится обращение к таблицам и производным таблицам на источниках данных. С точки зрения конечного пользователя псевдонимы подобны алиасам.

На производительность системы объединения влияют многие факторы. Самый важный фактор - обеспечить точность и актуальность информации об источниках данных и объектах, хранимой в глобальном каталоге базы данных объединения. Эта информация используется оптимизатором DB2 и может повлиять на решения об операциях для оценки источников данных. Дополнительную информацию о производительности системы объединения смотрите в книге *Administration Guide: Performance*.

Система объединения DB2 работает с некоторыми ограничениями. Распределенные требования ограничиваются операциями только для чтения.

Кроме того, нельзя применять к псевдонимам утилиты (LOAD, REORG, REORGCHK, IMPORT, RUNSTATS и т.п.).

Однако с помощью возможности работы через промежуточный сервер операторы DDL и DML можно передать непосредственно менеджерам баз данных, используя диалект SQL этого источника данных.

Системы объединения допускают параллельные среды. Повышение производительности ограничивается степенью, до которой запрос базы данных объединения может быть семантически разбит на обращения к локальным объектам (таблицам, производным таблицам) и обращения к псевдонимам. Требования данных псевдонимов обрабатываются последовательно; локальные объекты могут обрабатываться параллельно. Например, если дан запрос SELECT \* FROM A, B, C, D, где A и B - локальные таблицы, а C и D - псевдонимы, ссылающиеся на таблицы на источниках данных Oracle, возможен план объединения, при котором таблицы A и B объединяются с использованием параллельного объединения. Затем результаты объединяются последовательно с псевдонимами C и D.

---

## Включение поддержки системы объединения

Базы данных объединения поддерживаются в DB2 Enterprise Edition (EE) и DB2 Enterprise - Extended Edition (EEE). Чтобы включить поддержку системы объединения:

1. При установке выберите для опции установки *Распределенное объединение для баз данных DB2* значение DB2 EE или EEE.
2. Если в состав системы объединения входят базы данных Oracle, установите DB2 Relational Connect. Дополнительную информацию смотрите в руководстве *Дополнение по установке и настройке*.
3. Задайте для параметра конфигурации менеджера баз данных *federated* значение "YES".
4. Создайте оболочки, серверы и псевдонимы (дополнительную информацию смотрите в разделе "Создание базы данных" в книге *Administration Guide: Implementation*).
5. При необходимости создайте дополнительные объекты или задайте опции (дополнительную информацию смотрите в разделе "Реализация проекта" в книге *Administration Guide: Implementation*).



---

## Глава 4. Параллельные системы баз 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. (Термин "узел" - node - использовался в DB2 Parallel Edition for AIX Версии 1.)

*Однораздельная база данных* - это база данных с единственным разделом. В этом разделе хранятся все данные базы данных. В этом случае группы узлов (смотрите раздел "Группы узлов" на стр. 9), даже если они заданы, не дают дополнительных возможностей.

*Многораздельная база данных* - это база данных с несколькими разделами. Таблицы могут находиться в одном или нескольких разделах базы данных. Если таблица находится в группе узлов, 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.

---

## Группы узлов и разделение данных

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

На рис. 13 на стр. 37 показан пример базы данных с пятью разделами, в которой:

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

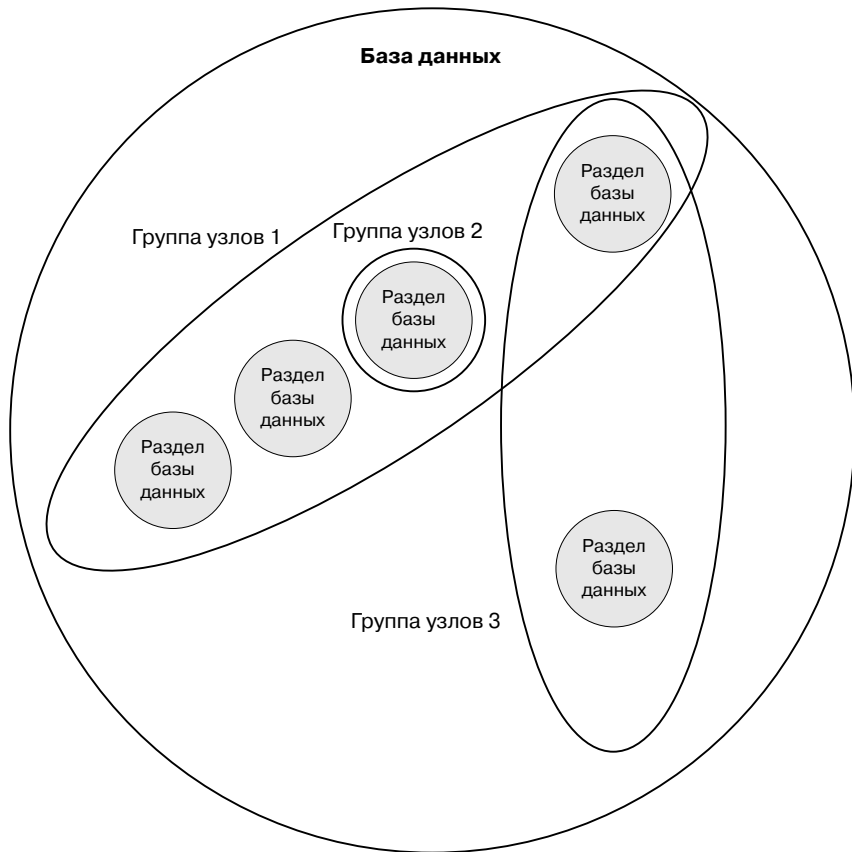


Рисунок 13. Группы узлов в базе данных

Новую группу узлов можно создать при помощи оператора `CREATE NODEGROUP`. Дополнительную информацию смотрите в справочнике *SQL Reference*. Данные делятся между всеми разделами группы узлов. При использовании многораздельных групп узлов следует учитывать некоторые особенности их разработки. Дополнительную информацию смотрите в разделе “Проектирование групп узлов” на стр. 102.

---

## Типы параллелизма

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. В этом разделе описаны следующие поддерживаемые DB2 типы параллелизма:

- ввод/вывод
- Параллелизм запросов
- Параллелизм утилит

## Параллелизм ввода/вывода

Если в табличном пространстве есть несколько контейнеров, менеджер базы 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 ввода/вывода - компонент каждой аппаратной среды, описанной в разделе “Аппаратные среды” на стр. 41. В Табл. 3 на стр. 50 описано, в каких аппаратных средах возможен параллелизм ввода/вывода.

## 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.

Именно в данном смысле термин *параллелизм запросов* используется во всей этой книге.

### Внутрираздельный параллелизм

*Внутрираздельный параллелизм* - это возможность разбивать запрос на несколько частей. (Некоторые утилиты также поддерживают этот тип параллелизма. Смотрите раздел “Параллелизм утилит” на стр. 41.)

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. 14 на стр. 39 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.

В Табл. 3 на стр. 50 описано, в каких аппаратных средах возможен внутрираздельный параллелизм.

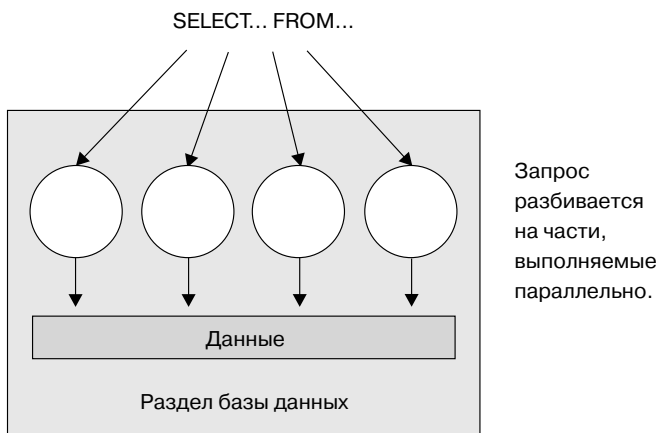


Рисунок 14. Внутрираздельный параллелизм

### 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 также поддерживают этот тип параллелизма. Смотрите раздел “Параллелизм утилит” на стр. 41.)

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. 15 на стр. 40 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 узлов.

В Табл. 3 на стр. 50 описано, в каких аппаратных средах возможен межраздельный параллелизм.

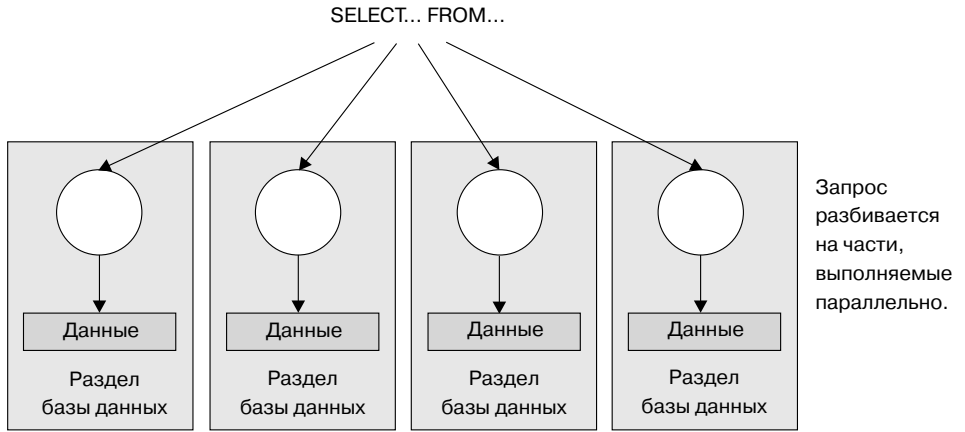


Рисунок 15. Межраздельный параллелизм

### Одновременный внутрираздельный и межраздельный параллелизм

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

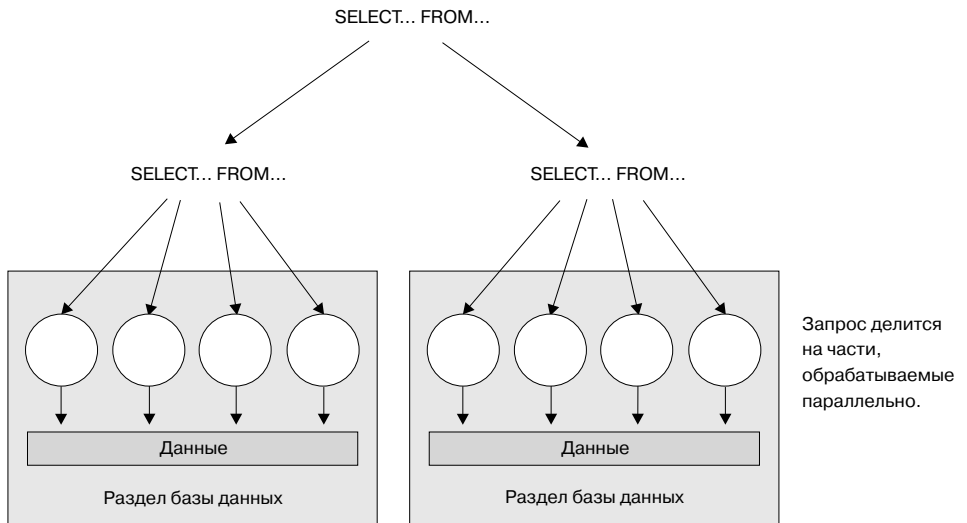


Рисунок 16. Одновременный межраздельный и внутрираздельный параллелизм

## Параллелизм утилит

Утилиты DB2 могут использовать преимущества внутрираздельного параллелизма. Кроме того, они могут использовать межраздельный параллелизм; если заданы разделы многораздельной базы данных, утилиты запускаются в каждом разделе параллельно.

Утилит загрузки может использовать преимущества внутрираздельного 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. Дополнительную информацию о подключении поддержки параллелизма для утилиты загрузки смотрите в книге *Data Movement Utilities Guide and Reference*.

В среде многораздельных баз данных утилита AutoLoader использует преимущества внутрираздельного параллелизма, межраздельного параллелизма и параллелизма ввода/вывода применением параллельных вызовов команды LOAD на каждом разделе базы данных, где находятся таблицы. Дополнительную информацию об утилите AutoLoader смотрите в книге *Data Movement Utilities Guide and Reference*.

Во время создания индекса просмотр и последовательная сортировка данных происходят параллельно. При создании индекса DB2 использует как параллелизм ввода/вывода, так и внутрираздельный 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 внутрираздельный 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. Дополнительную информацию о подключении поддержки параллелизма для этих утилит смотрите в описании команд BACKUP DATABASE и RESTORE DATABASE в книге *Command Reference*.

---

## Аппаратные среды

В этом разделе описываются следующие аппаратные среды:

- Однораздельная с одним процессором (однопроцессорная система)
- Однораздельная 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 времен ответов.

### **Один раздел с одним процессором**

Эта среда состоит из памяти, диска и единственного процессора (смотрите рис. 17 на стр. 43). 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.

В Табл. 3 на стр. 50 приведены типы параллелизма, наиболее подходящие для использования преимуществ этой аппаратной конфигурации.



## Однопроцессорный компьютер



Рисунок 17. Один раздел с одним процессором

### Емкость 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 однораздельной среде с несколькими процессорами.

### Один раздел с несколькими процессорами

Эта среда обычно состоит из нескольких процессоров одинаковой мощности на одном компьютере (смотрите рис. 18 на стр. 44) 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.

В Табл. 3 на стр. 50 приведены типы параллелизма, наиболее подходящие для использования преимуществ этой аппаратной конфигурации.

### Компьютер SMP

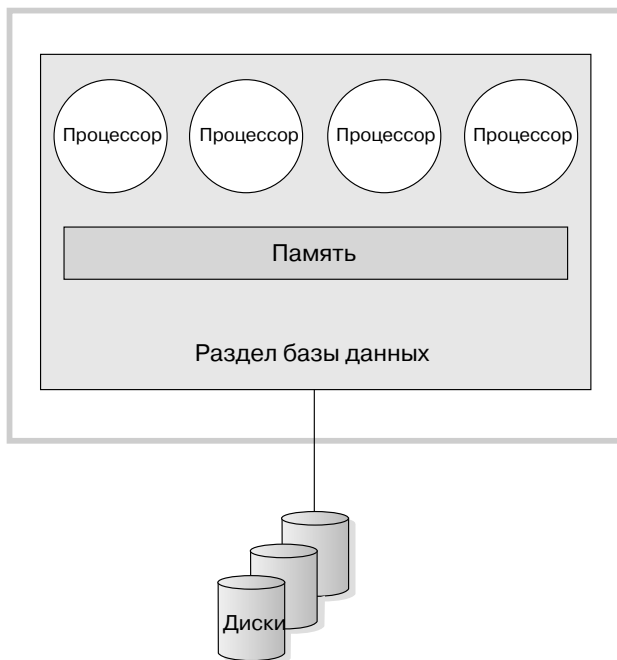


Рисунок 18. Симметричная мультипроцессорная система с однораздельной базой данных

### Емкость 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

### **Несколько однопроцессорных разделов**

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. 19 на стр. 46). Все компьютеры соединяются средствами связи. Эта среда может называться по-разному, например, кластером, кластером однопроцессорных систем, средой с 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.

В Табл. 3 на стр. 50 приведены типы параллелизма, наиболее подходящие для использования преимуществ этой аппаратной конфигурации.

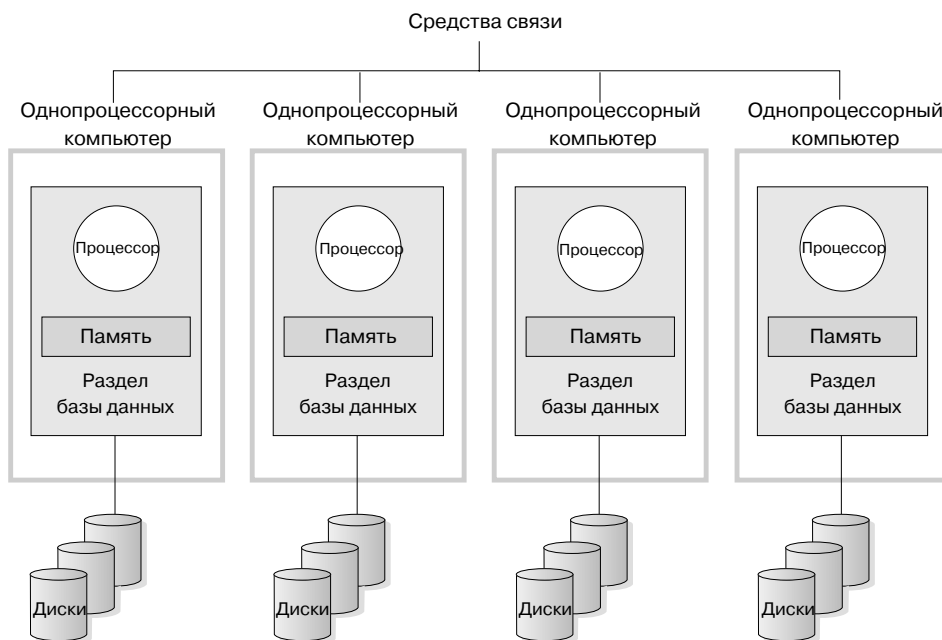


Рисунок 19. Система широкомасштабной параллельной обработки

**Емкость 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.

### Несколько многопроцессорных разделов

Альтернатива конфигурации, в которой каждый раздел имеет единственный процессор - это конфигурация, где раздел использует несколько процессоров. Такую среду называют *кластером SMP* (смотрите рис. 20 на стр. 47).

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.

В Табл. 3 на стр. 50 приведены типы параллелизма, наиболее подходящие для использования преимуществ этой аппаратной конфигурации.

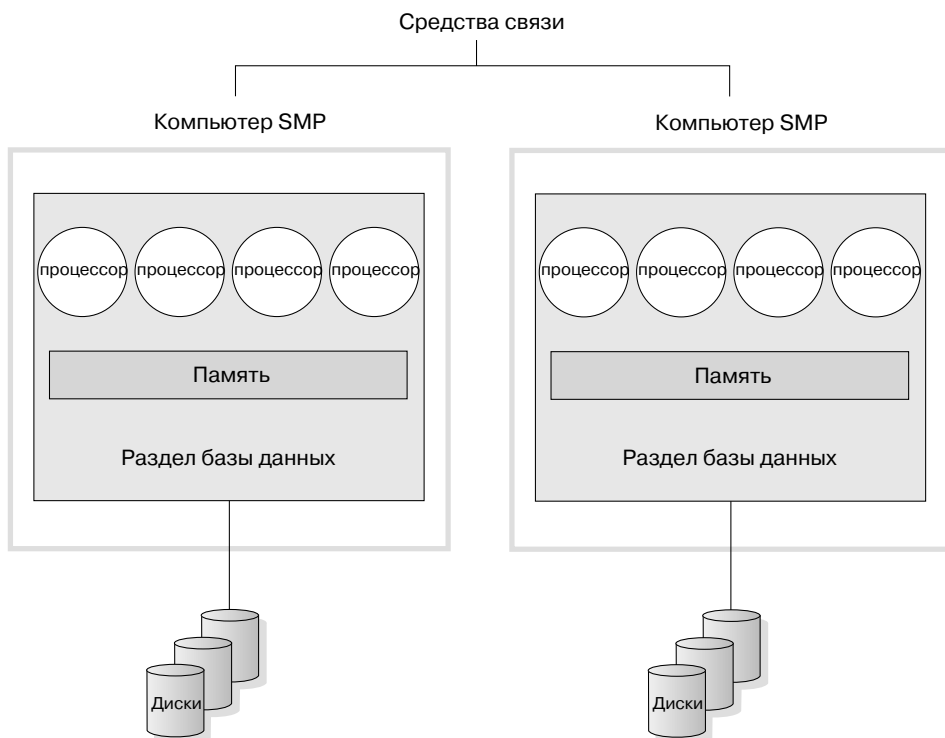


Рисунок 20. Кластер сред SMP

**Емкость 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 базы данных. На рис. 21 на стр. 48 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.

## Большой компьютер SMP

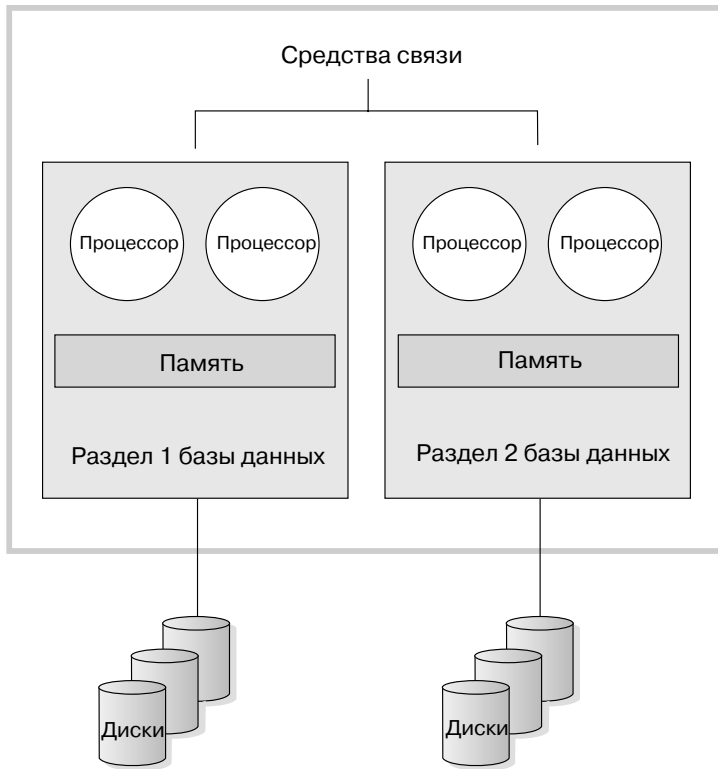


Рисунок 21. Многораздельная база данных, симметричная мультипроцессорная система

На рис. 22 на стр. 49 показано, как распространить конфигурацию, показанную на рис. 21, на несколько компьютеров, чтобы увеличить мощность обработки.

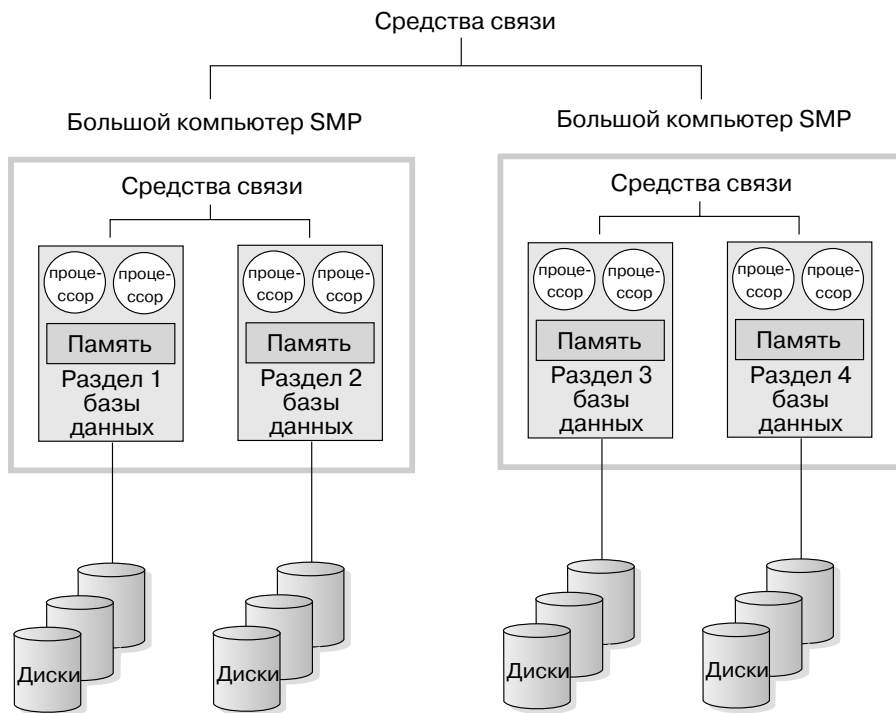


Рисунок 22. Многораздельная база данных, совместная кластеризация симметричных мультипроцессорных систем

В Табл. 3 на стр. 50 приведены типы параллелизма, наиболее подходящие для использования преимуществ этой аппаратной среды.

**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. Дополнительную информацию смотрите в разделе “Глава 11. Высокая доступность и восстановление при отказах. Введение” на стр. 187.

### Сводка по вариантам параллелизма для каждой аппаратной среды

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
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
<p><b>Примечание:</b> (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).</p>			



---

## Глава 5. О хранилищах данных

DB2 Universal Database содержит Центр хранилищ данных, компонент, который автоматизирует работу хранилищ. Центр хранилищ данных можно использовать для определения данных, помещаемых в хранилище. Затем Центр хранилищ можно применить для автоматического регулярного обновления данных в хранилище.

В этом разделе описывается работа и задачи хранилища данных. Более подробную информацию о хранилищах данных и информацию об использовании Центра хранилищ данных можно найти в книге *Data Warehouse Center Administration Guide* и в справке Центра хранилищ 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 времени года для поиска аномалий или планирования будущих продаж.

Однако если аналитики обращаются к рабочим данным непосредственно, возникает несколько проблем:

- У них 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 *информационные данные* – данные, извлеченные из рабочих данных и преобразованные для принятия решений конечными пользователями. Например, средствами хранилища данных можно скопировать данные продаж из рабочей базы данных, выполнить вычисления для составления по этой информации сводных данных и записать полученные сводные данные в другую базу данных,

отдельную от рабочей. Конечные пользователи могут строить запросы к этой отдельной базе данных (*хранилищу*), не затрагивая рабочих баз данных.

В следующем разделе описываются объекты (тематические области, источники хранилища, потребители хранилища, узлы агентов, шаги и процессы), 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 можно добавить процессы, связанные с продажами, в тематическую области Продажи. Подобным образом определения, связанные с данными маркетинга, можно добавить 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 и для получения эти данных. Источником может быть практически любой реляционный или нереляционный источник (таблица, производная таблица или файл), который подключен к вашему хранилищу.

## **Потребители хранилища**

*Потребители хранилища* - это таблицы или файлы баз данных, содержащие данные, преобразованные для удобства использования их конечными пользователями. Потребители хранилища могут, как и источники, давать данные для шагов Центра хранилищ данных.

## **Агенты хранилища и узлы агентов**

*Агенты* Центра хранилищ данных управляют потоком данных между источниками данных и потребителями данных. Агенты доступны в операционных системах Windows NT, AIX, OS/2, OS/390, OS/400 и SUN 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 через 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 том же компьютере, что и сервер хранилища. *Удаленный агент хранилища* устанавливается на другом компьютере, 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 одного физического компьютера может быть только одно имя хоста TCP/IP. Однако на одном компьютере можно определить несколько узлов агентов. Логическое имя служит для идентификации каждого узла агента.

*Узел агента по умолчанию*, называемый Default VW AgentSite - это локальный агент в Windows NT, который Центр хранилищ данных определяет во время установки управляющей базы данных хранилища.

## Шаги и процессы

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

- Структуру выходной таблицы или файла.
- Механизм (SQL или программу) для заполнения выходной таблицы или файла.
- Расписание, по которому заполняется выходная таблица или файл.

Шаги перемещают и преобразуют данные с помощью операторов SQL или вызовов программ. При запуске шага выполняется передача данных между источником хранилища и потребителем хранилища и все необходимые преобразования этих данных.

*Процесс* состоит из серий шагов, выполняющих задачи по преобразованию и перемещению данных. В общем случае, процесс заполняет потребитель хранилища в базе данных хранилища, извлекая данные из одного или нескольких источников, которыми могут быть таблицы или файлы базы данных. Однако можно определить процесс для запуска программ, который не задает ни источников, ни потребителей.

Шаг можно запускать по требованию или запланировать его запуск на заданное время. Можно спланировать одновременный 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 завершения (успешного или неудачного) какого-либо другого шага. Если составляется расписание процесса, первый шаг в процессе запускается во время, указанное в расписании.

При выполнении шаг или процесс может сохранять данные:

- Заменяя все данные в потребителе хранилища новыми.

- Добавля новые данные к существующим.
- Добавля отдельный вариант данных.

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 отдельные шаги. Каждый шаг будет выполнять отдельную задачу, например извлекать данные из баз данных или преобразовывать их в нужный формат. Затем нужно воспользоваться другим шагом для заполнения таблицы назначения, содержащей преобразованные данные.

В следующих разделах описываются разные типы шагов, которые вы найдете в Центре хранилищ данных. Дополнительную информацию о шагах смотрите в книге *Data Warehouse Center Administration Guide*.

### **Шаги SQL**

Шаг SQL использует оператор SQL SELECT для извлечения данных из источника хранилища и генерирует оператор INSERT для вставки данных в таблицу назначения хранилища.

### **Шаги программ**

Программные шаги бывают различных типов: программы DB2 for AS/400, DB2 for OS/390, DB2 for UDB, Visual Warehouse 5.2 DB2, 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 программу, которую должен запускать Центр хранилищ данных. Шаг пользовательской программы может запускаться агентом хранилища:

- Во время заполнения потребителя хранилища.
- После заполнения потребителя хранилища.
- Самостоятельно.

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

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

---

## Задачи хранилищ

При создании хранилища данных решаются следующие задачи:

- Определение тематической области, идентифицирующей и группирующей процессы, которые будут использоваться в хранилище.
- Исследование данных источника (или рабочих данных) и определение источников хранилища.
- Создание базы данных для использования в качестве хранилища и определение потребителей хранилища.
- Задание способов перемещения и преобразования данных источника в формат для базы данных хранилища с помощью определения процесса.
- Проверка определенных вами шагов и составление расписания их автоматического запуска.
- Управление хранилищем с использованием задаваемой защиты и мониторинга базы данных.
- Создание каталога данных в хранилище, если у вас есть пакет DB2 Warehouse Manager. Каталог данных - это база данных, содержащая деловые метаданные. Метаданные помогают пользователям идентифицировать и находить данные и информацию, доступную для них в организации. Конечные пользователи хранилища могут проводить поиск в этом каталоге, чтобы определять, к каким таблицам направлять запросы.
- 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). Например, если ваша фирма производит прохладительные напитки, 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.

- Объединение таблицы фактов и таблиц ассоциаций для объединения подробных сведений из таблиц ассоциаций с информацией о заказах. Например, можно объединить таблицу ассоциаций продуктов с таблицей фактов, чтобы добавить к ней информацию о том, как каждый продукт включался в пакеты заказов.

Об этих и других задачах можно узнать подробнее, воспользовавшись учебником *Business Intelligence Tutorial*, просмотрев краткий обзор *DB2 Universal Database Quick Tour* или прочитав книгу *Data Warehouse Center Administration Guide*.

---

## Глава 6. О модуле Spatial Extender

В этом разделе кратко описан модуль Spatial Extender, объяснены его задачи и рассказано о данных, которые он обрабатывает. Подробную информацию об использовании Spatial Extender смотрите в книге *Spatial Extender User's Guide and Reference*.

---

### Назначение Spatial Extender

Spatial Extender используется для создания *географической информационной системы* (GIS): комплекса объектов, данных и программ, позволяющих генерировать и анализировать пространственные данные о географических объектах. К *географическим объектам* относятся естественные и искусственные объекты, образующие земную поверхность. Они составляют природную (реки, леса, горы и пустыни) и культурную (города, населенные пункты, предприятия, ориентиры на местности и т.п.) среду.

К *пространственным данным* относятся:

- Положение географических особенностей по отношению окрестностям (например, точки в городе, где расположены больницы и клиники, или близость населенных пунктов к сейсмическим зонам)
- Пути связи между географическими объектами (например, информация, что определенная речная система находится в некотором районе, или что мосты в данном районе пересекают ее притоки)
- Измерения, относящиеся к одной или нескольким географическим особенностям (например, расстояние от здания предприятия до границ его участка или длина периметра заповедника)

Пространственные данные, как сами по себе, так и в сочетании с традиционным выводом базы данных, могут помочь в разработке проектов и принятии деловых и политических решений. Предположим, что управляющему по социальному обеспечению округа нужно проверить, что кандидаты и получатели социальной помощи действительно проживают на территории обслуживания этого округа. Spatial Extender может извлечь эти сведения из местоположения обслуживаемой территории и адресов кандидатов и получателей.

Или предположим, что владелец сети ресторанов желает распространить свой бизнес на близлежащие города. Чтобы определить, где открыть новые рестораны, ему нужно получить ответы на такие вопросы как, например: Где в этих городах концентрируется тип людей, которые будут часто посещать мои рестораны? Где находятся главные магистрали? Где самые низкие показатели преступности? Где находятся рестораны моих конкурентов? Чтобы ответить на

такие вопросы, Spatial Extender может дать наглядное представление пространственных данных, а лежащая в основе система управления базой данных может сгенерировать пометки и текст для объяснения вывода.

---

## Данные о географических объектах

В этом разделе представлен обзор данных, которые вы генерируете, храните и обрабатываете для получения пространственной информации. Рассматриваемые темы:

- Как данные описывают географические объекты
- Характер пространственных данных
- Способы получения пространственных данных

### Как данные представляют географические объекты

В Spatial Extender географический объект может быть представлен строкой в таблице или производной таблице или частью такой строки. Например, рассмотрим следующие географические объекты: офисные и жилые здания. На рис. 23 каждая строка таблицы BRANCHES описывает филиал банка. Другой вариант - каждая строка таблицы CUSTOMERS в целом описывает клиента банка, но часть этой строки (а именно столбцы с адресами клиентов) можно рассматривать как представление места жительства клиента.

В этих таблицах содержатся данные, идентифицирующие и описывающие филиалы банков и клиентов. Такие данные называются *данными атрибутов*.

#### Таблица филиалов BRANCHES

ID	NAME	ADDRESS	CITY	STATE	ZIP
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141

#### Таблица клиентов CUSTOMERS

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141	A	A

*Рисунок 23. Строка таблицы, описывающая географический объект; строка таблицы, данные об адресе в которой описывают географический объект. Строка данных в таблице BRANCHES описывает филиал банка. Данных об адресах в таблице CUSTOMERS описывают место жительства клиента.*

Подмножество данных атрибутов (значения, описывающие филиалы банков и адреса клиентов) могут быть преобразованы в поля, содержащие пространственные данные. Например, на рисунке указан следующий адрес одного из филиалов банка: 92467 Airzone Blvd., San Jose CA 95141. Адрес клиента - 9 Concourt Circle, San Jose CA 95141. Spatial Extender может



преобразовать эти адреса в значения, указывающие положение филиала банка и дома клиента относительно окрестностей. На рис. 24 показаны таблицы BRANCHES и CUSTOMERS с новыми столбцами, содержащими такие значения.

**Таблица филиалов BRANCHES**

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	

**Таблица клиентов CUSTOMERS**

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141		A	A

Рисунок 24. Таблицы с добавленными пространственными столбцами. В каждой таблице столбец LOCATION будет содержать координаты, соответствующие этим адресам.

Если адреса и подобные идентификаторы используются в качестве отправной точки для пространственной информации, они называются *исходными данными*. Так как полученные из них значения поставляют пространственную информацию, эти значения называются *пространственными данными*. Следующий раздел рассказывает о пространственных данных и знакомит со связанными с ними типами данных.

## Характер пространственных данных

Многие пространственные данные состоят из координат. *Координата* - это число, обозначающее положение относительно начала отсчета. Например, широта - это координата, обозначающая положение относительно экватора. Долгота - координата, обозначающая положение относительно гринвичского меридиана. Таким образом, положение Йелоустонского национального парка обозначается его широтой (44,45 градусов к северу от экватора) и долготой (110,40 градусов к западу от гринвичского меридиана).

Широта, долгота, их начала отсчета и другие относящиеся к ним параметры вместе называются *системой координат*. Существуют системы координат на основе иных значений, отличающихся от широты и долготы. В этих системах координат есть собственные измерения положения, начала отсчета и дополнительные отличительные параметры.

Простейший элемент пространственных данных состоит из двух координат, определяющих положение одиночной географической точки. Термин *элемент данных* относится к значению или значениям, занимающим ячейку в реляционной таблице. Другой элемент данных может описываться несколькими координатами, которые определяют линейный путь, например дорогу или реку.

Третий вид состоит из координат, определяющих периметр области, например, границы земельного участка или поймы.

Каждый элемент этих пространственных данных является примером одного из типов пространственных данных. Тип данных для двух координат, обозначающих положение - `ST_Point`, тип данных для координат, обозначающих линейные пути - `ST_LineString` и тип данных для координат, обозначающих периметры - `ST_Polygon`. Эти типы вместе с другими типами пространственных данных являются структурированными типами, принадлежащими к отдельной иерархии.

## Откуда берутся пространственные данные

Пространственные данные можно получить:

- Из данных атрибутов
- Из других пространственных данных
- Посредством импорта

### Использование данных атрибутов в качестве исходных данных

Получение данных из данных атрибутов (например, из адресов) называется *геокодированием*. На рис. 24 на стр. 59 показаны два столбца (один в таблице `BRANCHES`, а второй - в таблице `CUSTOMERS`), предназначенные для пространственных данных. Представим, что модуль `Spatial Extender` выполнил геокодирование адресов в этих таблицах и поместил результаты (координаты, соответствующие адресам) в эти столбцы. Этот результат показан на рис. 25.

#### Таблица филиалов `BRANCHES`

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	1653 3094

#### Таблица клиентов `CUSTOMERS`

ID	LAST NAME	FIRST NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	CHECKING	SAVINGS
59-6396	Kriner	Endela	9 Concourt Circle	San Jose	CA	95141	953 1527	A	A

*Рисунок 25. Таблицы с пространственными данными, полученными из исходных данных. Столбец `LOCATION` в таблице `CUSTOMERS` содержит координаты, полученные геокодировщиком из адреса в столбцах `ADDRESS`, `CITY`, `STATE` и `ZIP`. Аналогично столбец `LOCATION` в таблице `BRANCHES` содержит координаты, полученные геокодировщиком из адреса в столбцах `ADDRESS`, `CITY`, `STATE` и `ZIP` данной таблицы.*

Для геокодирования данных атрибутов и помещения полученных пространственных данных в столбцы `Spatial Extender` использует функцию, которая называется *геокодировщик*.

## Использование в качестве исходных данных других пространственных данных

Пространственные данные можно сгенерировать не только из данных атрибутов, но также и из других пространственных данных. Предположим, что банку, филиалы которого определены в в таблице BRANCHES, нужно знать, сколько клиентов находятся в пределах пяти миль от каждого филиала. Прежде, чем банк сможет получить эту информацию из базы данных, он должен дать определение зоны, лежащей внутри пятимильного радиуса вокруг каждого филиала. Такое определение может создать функция Spatial Extender ST\_Buffer. Используя в качестве вводных данных координаты каждого филиала, эта функция может сгенерировать координаты, разграничивающие периметры нужных зон. На рис. 26 показана таблица BRANCHES с информацией, поставляемой функцией ST\_Buffer.

**Таблица филиалов BRANCHES**

ID	NAME	ADDRESS	CITY	STATE	ZIP	LOCATION	SALES_AREA
937	Airzone-Multern	92467 Airzone Blvd	San Jose	CA	95141	1653 3094	1002 2001, 1192 3564, 2502 3415, 1915 3394, 1002 2001

*Рисунок 26. Таблица, содержащая новые пространственные данные, полученные из существующих пространственных данных. Координаты в столбце SALES\_AREA были получены функцией ST\_Buffer из координат в столбце LOCATION.*

Помимо функции ST\_Buffer модуль Spatial Extender обеспечивает несколько других функций, которые могут получать новые пространственные данные из существующих пространственных данных.

## Импорт пространственных данных

Третий способ получения пространственных данных - их импорт из файлов одного из форматов, поддерживаемых Spatial Extender. Эти файлы содержат данные, которые обычно применяются для карт: данные о переписи, о затопляемых поймах, сейсмоопасных районах и т.п. Воспользовавшись такими данными в сочетании с вашими пространственными данными, можно расширить доступную вам географическую информацию. Например, если отделу по социальной работе требуется определить опасности, представляющие угрозу для проживающего населения, можно воспользоваться функцией ST\_Buffer для определения зоны данного округа, а затем импортировать данные по затопляемым поймам и сейсмоопасным районам, чтобы посмотреть, какие из этих проблемных областей пересекаются с данной зоной.



---

## Часть 3. Проектирование баз данных



---

## Глава 7. Логическое проектирование базы данных

В этом разделе описываются следующие этапы проектирования базы данных:

- “Решите, какие данные записывать в базу данных”
- “Определите таблицы для каждого типа отношений” на стр. 67
- “Дайте определения столбцов для всех таблиц” на стр. 69
- “Выберите один или несколько столбцов в качестве первичного ключа” на стр. 72
- “Обеспечьте, чтобы совпадающие значения представляли один объект” на стр. 75
- “Рассмотрите возможность нормализации таблиц” на стр. 76
- “Планирование введения ограничений” на стр. 82
- “Другие особенности проектирования базы данных” на стр. 89.

Цель при проектировании базы данных - создать представление вашей среды, которое будет легко понять и можно 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 устранил аномалии, которые могут возникать при изменении базы данных.

Приведенные действия - это часть процесса *логического* проектирования базы данных. Проектирование базы данных - это нелинейный процесс; скорее всего, при разработке структуры базы данных некоторые шаги придется повторять.

*Физическая* реализация структуры базы данных описана в разделах “Глава 8. Физическая структура базы данных” на стр. 91 и “Реализация вашего проекта” в книге *Administration Guide: Implementation*.

---

### Решите, какие данные записывать в базу 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 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. (Описание базы данных примера смотрите в справочнике *SQL Reference*.)

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
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
- Отделы подотчетны другим отделам
  - Отдел C01 подотчетен отделу A00
  - Отдел B01 подотчетен отделу A00
- Сотрудники работают над проектами
  - Долорес и Хитер заняты в проекте IF1000
- 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.

---

## Определите таблицы для каждого типа отношений

В базе данных может быть определено несколько типов отношений. Рассмотрим возможные отношения между сотрудниками и отделами. 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-с-многими*. В этом разделе рассмотрены следующие типы отношений:

- "Отношения один-с-многими и многие-с-одним"
- "Связи Многие-с-многими" на стр. 68
- "Отношение один-с-одним" на стр. 69

### Отношения один-с-многими и многие-с-одним

Чтобы определить таблицы для каждого отношения один-с-многими и многие-с-одним:

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-с-одним

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

### Связи Многие-с-многими

Многозначное для обеих сторон отношение - это отношение многие-со-многими. 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-со-многими может быть выражена таблицей со столбцами для каждого объекта ("сотрудников" и "проектов"), как показано в следующем примере.

Следующий пример показывает, как отношение многие-с-многими (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

### Отношение один-с-одним

Отношения один-с-одним 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-с-одним отражается в таблице:

Таблица DEPARTMENT:

DEPTNO	MGRNO
A00	000010
B01	000020
D11	000060

---

### Дайте определения столбцов для всех таблиц

Чтобы определить столбец в реляционной таблице:

1. Выберите имя для столбца.

У каждого столбца должно быть уникальное в своей таблице имя. Выбор имен для столбцов подробно описан в разделе "Приложение А. Правила именования" на стр. 195.

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. Информацию о встроенных и пользовательских типах DB2 смотрите в справочнике *SQL Reference*.

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.

Чтобы лучше понять, как поддерживаются большие объекты, посмотрите справочник *SQL Reference*.

*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.

Чтобы лучше понять, что такое пользовательские типы, структурированные типы, ссылочные типы и пользовательские функции, посмотрите справочник *SQL Reference*.

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 не известно.

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 не применимо), или использовать по умолчанию другое значение, назначаемое менеджером баз данных или программой.

Значение NULL и значения по умолчанию подробно описаны в справочнике *SQL Reference*.

---

## Выберите один или несколько столбцов в качестве первичного ключа

*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. (Смотрите также раздел "Определите столбцы идентификации" на стр. 74.)

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

## Выберите ключевые столбцы

Для определения столбцов, которые разумно выбрать ключевыми, найдите наименьшее число столбцов, которое однозначно определяет объект. Может

найти несколько подходящих ключей. В Табл. 2 на стр. 22 можно выбирать различные ключи. Каждый из столбцов EMPNO, PHONENO и LASTNAME однозначно определяет сотрудника.

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 примера 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.

## Определите 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, может достичь гораздо более высокого уровня одновременности, так как счетчик не будет блокироваться транзакциями. Одна не принятая транзакция, увеличившая счетчик, не мешает 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 восстанавливаемо. Если происходит ошибка, значение счетчика восстанавливается по журналам, таким образом уникальность значений остается гарантированной.
- Значения столбца идентификации могут кэшироваться для лучшей производительности.

---

## **Обеспечьте, чтобы совпадающие значения представляли один объект**

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).

---

## Рассмотрите возможность нормализации таблиц

*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 это 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 (смотрите раздел “Первая нормальная форма” на стр. 77).

#### *Вторая*

Каждый столбец, не входящий в ключ, зависит от ключа (смотрите раздел “Вторая нормальная форма” на стр. 77).

#### *Третья*

Каждый неключевой столбец независим от других неключевых столбцов; он зависит только от ключа (смотрите раздел “Третья нормальная форма” на стр. 79).

#### Четвертая

Ни в одной строке не содержится нескольких независимых многозначных фактов об одном объекте (смотрите раздел “Четвертая нормальная форма” на стр. 81).

### 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 8. Таблица, нарушающая первую нормальную форму

PART (Первичный ключ)	WAREHOUSE
P0010	Склад А, Склад В, Склад С
P0020	Склад В, Склад D

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

Таблица 9. Таблица, соответствующая первой нормальной форме

PART (Первичный ключ)	WAREHOUSE (Первичный ключ)	QUANTITY
P0010	Склад А	400
P0010	Склад В	543
P0010	Склад С	329
P0020	Склад В	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 10. Таблица, нарушающая вторую нормальную форму

<b>PART</b> (Первичный ключ)	<b>WAREHOUSE</b> (Первичный ключ)	<b>QUANTITY</b>	<b>WAREHOUSE_ADDRESS</b>
P0010	Склад А	400	1608 New Field Road
P0010	Склад В	543	4141 Greenway Drive
P0010	Склад С	329	171 Pine Lane
P0020	Склад В	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 11. Таблица PART\_STOCK, соответствующая второй нормальной форме

<b>PART</b> (Первичный ключ)	<b>WAREHOUSE</b> (Первичный ключ)	<b>QUANTITY</b>
P0010	Склад А	400
P0010	Склад В	543
P0010	Склад С	329
P0020	Склад В	200
P0020	Склад D	278

Таблица 12. Таблица WAREHOUSE, соответствует второй нормальной форме

<b>WAREHOUSE</b> (Первичный ключ)	<b>WAREHOUSE_ADDRESS</b>
Склад А	1608 New Field Road

Таблица 12. Таблица WAREHOUSE, соответствует второй нормальной форме (продолжение)

WAREHOUSE (Первичный ключ)	WAREHOUSE_ADDRESS
Склад В	4141 Greenway Drive
Склад С	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.

Чтобы лучше понять вопросы, связанные с производительностью, посмотрите раздел "Настройка производительности программы" в книге *Administration Guide: Performance*.

### 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. 14 на стр. 80). 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 13. 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 14. 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 15. 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 16. 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 17. 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 18. Таблица EMPLOYEE\_SKILL, соответствующая четвертой нормальной форме

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

Таблица 19. 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.

---

## Планирование введения ограничений

*Ограничение* - это правило, которому подчиняется менеджер базы данных. В этом разделе рассмотрено четыре типа работы с ограничениями:

- |  |  |
|--|--|
| <b>Ограничения уникальности</b>        | Проверка уникальности значений ключа в таблице. Любые изменения, вносимые в столбец первичного ключа, проверяются на соблюдение уникальности.              |
| <b>Реляционная целостность</b>         | Накладывает реляционные ограничения на операции вставки, изменения и удаления. В этом состоянии базы данных допустимы любые значения любых внешних ключей. |
| <b>Проверочные ограничения таблицы</b> | Проверяет, что измененные данные не нарушают условий, указанных при создании или переопределении таблицы.  |
| <b>Триггеры</b>                        | Определяет набор действий, которые надо выполнить при вызове операций изменения, удаления или вставки для данной таблицы.                                  |

### Ограничения уникальности

*Ограничение уникальности* - это правило, которое гарантирует уникальность значений ключа внутри таблицы. Каждый ключевой столбец в ограничении уникальности должен быть определен как NOT NULL. Ограничения уникальности определяются в операторах CREATE TABLE или ALTER TABLE с помощью условий PRIMARY KEY или UNIQUE.

У таблицы может быть сколько угодно ограничений уникальности, но только одно ограничение уникальности можно определить как первичный ключ для таблицы. Более того, таблица не может иметь несколько ограничений уникальности на одном и том же наборе столбцов.

Если определено ограничение уникальности, менеджер базы данных создает (если надо) индекс уникальности и назначает его первичным индексом или системным индексом уникальности. Ограничение реализуется через индекс уникальности. Когда ограничение уникальности установлено, проверка уникальности при изменении многих строк откладывается до конца изменения.



Ограничение уникальности также можно использовать как родительский ключ в реляционном ограничении.

## Реляционная целостность

Менеджер баз данных поддерживает реляционную целостность с помощью *реляционных ограничений*, которые требуют, чтобы каждое значение данного атрибута или столбца таблицы существовало также в какой-то другой таблице или столбце. Например, реляционное ограничение может требовать, чтобы каждый сотрудник из таблицы EMPLOYEE работал в одном из отделов, который есть в таблице DEPARTMENT. Сотрудник не может работать в несуществующем отделе.

Можно включить реляционные ограничения в базу данных, чтобы гарантировать поддержание реляционной целостности и дать оптимизатору возможность использовать информацию об этих специальных отношениях для более эффективной обработки запросов. При планировании реляционной целостности учтите все отношения между таблицами базы данных. Обозначить отношение можно, определив первичный ключ и реляционные ограничения.

Рассмотрим следующие связанные таблицы:

Таблица 20. Таблица DEPARTMENT

DEPTNO (Первичный ключ)	DEPTNAME	MGRNO
A00	Spiffy Computer Service Div.	000010
B01	Плановый отдел	000020
C01	Информационный центр	000030
D11	Производственный отдел	000060

Таблица 21. 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

Многие понятия, полезные для понимания реляционной целостности, будут обсуждаться на примере этих таблиц.

*Ключ уникальности* - это столбец или набор столбцов, в котором значения не повторяются в различных строках. Вы можете определить один из ключей уникальности как первичный ключ таблицы. Ключ уникальности называется *родительским ключом*, если на него ссылается внешний ключ.

*Первичный ключ* - это ключ уникальности, указанный в определении таблицы. У каждой таблицы может быть только один первичный ключ. У приведенных выше таблиц DEPARTMENT и EMPLOYEE первичными ключами являются столбцы DEPTNO и EMPNO соответственно.

*Внешний ключ* - это столбец или набор столбцов таблицы, ссылающийся на ключ уникальности или первичный ключ той же самой или другой таблицы. Внешний ключ используется для образования отношений с ключом уникальности или первичным ключом, чтобы поддерживать реляционную целостность между таблицами. Столбец WORKDEPT в таблице EMPLOYEE - это внешний ключ, так как он ссылается на первичный ключ - столбец DEPTNO - в таблице DEPARTMENT.

*Составной ключ* - это ключ, состоящий из нескольких столбцов. И первичный, и внешний ключи могут быть составными. Например, если отделы однозначно идентифицируются комбинацией из номера подразделения и номера отдела, понадобится два столбца, чтобы создать ключ для таблицы DEPARTMENT.

*Родительский ключ* - это первичный ключ или ключ уникальности реляционного ограничения. *Ограничение первичного ключа* - это родительский ключ реляционного ограничения по умолчанию, если набор столбцов родительского ключа не указан.

*Родительская таблица* - это таблица, содержащая родительский ключ, связанный хотя бы с одним внешним ключом той же самой или другой таблицы. Таблица может быть родительской в произвольном числе связей. Например, таблица DEPARTMENT с первичным ключом DEPTNO является родительской для таблицы EMPLOYEE с внешним ключом WORKDEPT.

*Родительская строка* - это строка родительской таблицы, значение родительского ключа которой совпадает хотя бы с одним значением внешнего ключа зависимой таблицы. Строка родительской таблицы не обязательно является родительской строкой. Четвертая строка (D11) таблицы DEPARTMENT является родительской строкой для третьей и шестой строк таблицы EMPLOYEE. Вторая строка (B01) таблицы DEPARTMENT не является родительской ни для какой строки.

*Зависимая таблица* - это таблица, содержащая один или более внешних ключей. Зависимая таблица сама может быть родительской таблицей. Таблица может

быть зависимой в произвольном числе связей. Таблица EMPLOYEE содержит внешний ключ WORKDEPT, и является зависимой от таблицы DEPARTMENT с первичным ключом DEPTNO.

*Зависимая строка* - это строка зависимой таблицы, имеющая непустое значение внешнего ключа, совпадающее со значением родительского ключа. Значение внешнего ключа представляет собой ссылку из зависимой строки на родительскую строку. Так как внешний ключ может принимать значения NULL, строка зависимой таблицы не обязательно является зависимой строкой.

Таблицу называют *потомком* другой таблицы, если это зависимая таблица, или если она потомок зависимой таблицы. Таблица-потомок содержит внешний ключ, по которому можно найти родительский ключ некоторой таблицы.

*Реляционный цикл* - это путь, связывающий таблицу саму с собой. Таблица, непосредственно связанная сама с собой, называется *автореферентной*. Если бы в таблице EMPLOYEE был столбец MGRID, содержащий EMPNO начальника для каждого сотрудника, она была бы автореферентной. Столбец MGRID был бы внешним ключом для таблицы EMPLOYEE.

Автореферентная таблица выступает одновременно как родительская и зависимая в одной взаимосвязи. Автореферентная строка - это строка, которая сама для себя и родительская, и зависимая. Ограничение, имеющее место в этой ситуации, называется автореферентным. Например, если значение внешнего ключа в строке автореферентной таблицы совпадает со значением ключа уникальности в этой строке, строка является автореферентной.

*Реляционное ограничение* - это требование, чтобы непустые значения указанного внешнего ключа допускались, только если они также встречаются среди значений ключа уникальности указанной таблицы. Назначение реляционных ограничений - это гарантия поддержания связей в базе данных и соблюдения правил ввода данных.

### **Последствия для операций SQL**

Введение реляционных ограничений приводит к особым последствиям для некоторых операций SQL, зависящим от того, является ли таблица родительской или зависимой. В этом разделе описано, как поддержание реляционной целостности влияет на операции SQL INSERT, DELETE, UPDATE и DROP.

DB2 *не* накладывает реляционные ограничения на системы автоматически. Следовательно, если вы хотите наложить реляционные ограничения на систему, в ваших программах должна быть реализована необходимая логика.

Далее описаны следующие темы:

- “Правила INSERT” на стр. 86
- “Правила DELETE” на стр. 86

- “Правила UPDATE” на стр. 87.

**Правила INSERT:** Вставить строку в родительскую таблицу можно в любой момент, не предпринимая никаких действий в отношении зависимой таблицы. Однако в зависимую таблицу можно вставить строку, если в родительской таблице есть строка со значением родительского ключа, равным значению внешнего ключа вставляемой строки, кроме случая, когда значение внешнего ключа пусто (NULL). Значение составного ключа считается пустым, если все его компоненты пусты (равны NULL).

Это правило накладывается при задании внешнего ключа.

При попытке вставить строку в таблицу с реляционными ограничениями операция INSERT не будет выполнена, если какое-либо непустое значение внешнего ключа не встречается в родительском ключе. Если при попытке вставить несколько строк операция INSERT не удалась для одной строки, не будет вставлена ни одна строка.

**Правила DELETE:** Когда из родительской таблицы удаляется строка, DB2 проверяет, нет ли в зависимых таблицах зависимых строк с соответствующими значениями внешнего ключа. Если нашлись зависимые строки, возможно несколько действий. Какое действие будет выполнено, определяет заданное вами при создании зависимой таблицы правило *удаления*.

Правила удаления для зависимой таблицы (таблицы, содержащей внешний ключ) при удалении первичного ключа:

**RESTRICT**

Не допускает удаления строк из родительской таблицы, если у них есть зависимые строки. Если надо удалить как родительские, так и зависимые строки, сначала удалите зависимые строки. Удаление родительских строк первыми нарушает реляционное ограничение и поэтому не допускается.

**NO ACTION**

Наличие родительской строки для каждого потомка проверяется после применения всех реляционных ограничений.

**CASCADE**

Означает, что удаление строки из родительской таблицы влечет автоматическое удаление всех связанных с ней строк зависимой таблицы. Это правило полезно, если строка зависимой таблицы не представляет интереса при отсутствии строки родительской таблицы.

Удаление родительской строки влечет автоматическое удаление зависимых строк,

ссылающихся на первичный ключ. Нет необходимости удалять сначала зависимые строки. Если у некоторых из зависимых строк есть свои зависимые строки, применяется правило удаления для этих отношений. DB2 управляет каскадным удалением.

## SET NULL

Внешнему ключу зависимых строк при удалении строки из родительской таблицы присваивается значение NULL. Другие части строки не изменяются.

Если при создании таблицы правила удаления не определены явно, применяется правило NO ACTION.

Любая таблица, которая может быть вовлечена в операцию удаления, называется связанной по удалению. Для отношения связи по удалению есть следующие ограничения.

- Таблица не может быть связана сама с собой по удалению через реляционный цикл из нескольких таблиц.
- Если таблица связана по удалению с другой таблицей через несколько отношений зависимости, у этих отношений должны быть одинаковые правила удаления - либо CASCADE, либо NO ACTION.
- Если автореферентная таблица зависит от другой таблицы в отношении CASCADE, правило удаления автореферентного отношения должно быть тоже CASCADE.

Удалять строки из зависимой таблицы можно в любой момент, не предпринимая никаких действий для родительской таблицы. В отношении отдел - сотрудник, например, увольнение сотрудника и удаление его строки из таблицы сотрудников никак не повлияет на таблицу отделов. (В обратном отношении - сотрудник - отдел - ID начальника отдела является внешним ключом, ссылающимся на родительский ключ таблицы сотрудников. В таком случае увольнение начальника *влияет* на таблицу отделов.)

**Правила UPDATE:** DB2 не допускает изменения ключа уникальности родительской строки. Когда внешний ключ в зависимой таблице изменяется, и новое значение непусто, оно должно соответствовать какому-нибудь значению родительского ключа родительской таблицы данной связи. Если операция UPDATE нарушает какие-либо реляционные ограничения, возникает ошибка и никакие строки не изменяются.

Когда изменяется значение в столбце родительского ключа:

- 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.

Чтобы изменить значение родительского ключа в родительской строке, сначала нужно удалить отношения со всеми дочерними строками в зависимых таблицах одним из следующих способов:

- Стерев дочерние строки; или,
- Изменив внешние ключи в зависимых таблицах, чтобы они содержали другие допустимые значения.

Если зависимостей от значения ключа для строки нет, строка больше не является родительской в реляционном отношении и ее можно изменить.

Если часть внешнего ключа изменяется и ни одна часть этого внешнего ключа не пуста, новое значение внешнего ключа должно встречаться и как значение ключа уникальности родительской таблицы. Если от данного ключа уникальности не зависит никакой внешний ключ, то есть строка, содержащая этот ключ уникальности, *не* является родительской, часть ключа уникальности можно изменить. Однако в этом случае для изменения можно выбрать только одну строку, так как вы работаете с ключом уникальности и повторение строк недопустимо.

## Проверочные ограничения таблицы

Логические правила, отражаемые в вашей структуре, можно реализовывать с помощью проверочных ограничений таблицы. *Проверочные ограничения таблицы* указывают условия, которые применяются к каждой строке таблицы. Эти ограничения автоматически активируются, когда к таблице применяется оператор изменения или вставки. Проверочные ограничения определяются в операторах CREATE TABLE или ALTER TABLE.

Проверочное ограничение таблицы можно использовать для проверки достоверности. Например, значения номера отдела должны лежать в диапазоне от 10 до 100; название должности сотрудника может быть только "Продавец", "Начальник" или "Клерк"; сотрудник, работающий в компании более 8 лет, должен зарабатывать не меньше 40500 долларов.

Информацию о влиянии проверочных ограничений на команды IMPORT и LOAD можно найти в руководстве *Data Movement Utilities Guide and Reference*.

## Триггеры

*Триггер* - это определенный набор действий, которые выполняются при каждом выполнении операций удаления, вставки или изменения над данной таблицей. Можно определить триггеры для соблюдения логических правил. Также триггеры можно использовать для автоматического обновления данных сводки

или аудита. Так как триггеры хранятся в базе данных, вам не придется программировать эти действия в каждой прикладной программе. Триггер программируется один раз и хранится в базе данных; он автоматически вызывается DB2, если это требуется, когда программа использует базу данных. Это гарантирует, что относящиеся к данным логические правила всегда будут выполнены. Если логическое правило поменялось, придется изменять только триггер.

Оператор SQL с триггером может вызывать пользовательские функции (UDF). Это позволяет вызывать при активации триггера операции, не описываемые SQL. Например, можно в качестве оповещения отправить электронную почту. Подробную информацию о триггерах смотрите в разделе "Создание триггера" книги *Administration Guide: Implementation* и в руководстве *Application Development Guide*.

---

## Другие особенности проектирования 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).

Есть другие вопросы, которые, возможно, вы захотите отразить в вашей структуре, такие как *действия аудита, хронологические данные, сводные таблицы, защита, контроль типов данных и возможность параллельной обработки*.

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 осуществлять с помощью возможности аудита DB2. Дополнительную информацию смотрите в разделе "Аудит действий DB2" в книге *Administration Guide: Implementation*.

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 будут полезны сводные таблицы для

каждого отдела или подразделения, основанные на данных исходной таблицы. Подробную информацию о сводных таблицах смотрите в разделе "Создание сводных таблиц" книги *Administration Guide: Implementation*.

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. Дополнительную информацию о защите и авторизациях смотрите в разделе "Управление доступом к базе данных" книги *Administration Guide: Implementation*.

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.

Дополнительную информацию о реализации структуры с типизированными строками и таблицами смотрите в разделе "Реализация вашего проекта" в руководстве *Administration Guide: Implementation*. Информацию о перемещении данных между типизированными таблицами, находящимися в иерархической структуре, смотрите в руководстве *Data Movement Utilities Guide and Reference*.

По мере роста вашего бизнеса вам могут понадобиться дополнительные емкость и производительность, обеспечиваемые DB2 Enterprise - Extended Edition. В этой среде ваша база данных распределена по нескольким компьютерам или системам, каждая из которых ответственна за хранение и выдачу некоторой порции всей базы данных. Все разделы (или узлы) при обработке операций утилит или SQL работают параллельно.

Вопросы, связанные с параллельными операциями, рассматриваются на протяжении всей этой книги.



---

## Глава 8. Физическая структура базы данных

После создания логической структуры базы данных (“Глава 7. Логическое проектирование базы данных” на стр. 65) 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.

Рассматриваются следующие темы:

- “Каталоги баз данных”
- “Оценка пространства, требуемого для таблиц” на стр. 93
- “Дополнительные требования к пространству” на стр. 101
- “Проектирование групп узлов” на стр. 102
- “Проектирование и выбор табличных пространств” на стр. 111
- “Особенности проектирования объединенных баз данных” на стр. 135

---

### Каталоги баз данных

При создании базы данных DB2 создает отдельный каталог для хранения управляющих файлов (например, файлов заголовков журналов) и для контейнеров, выделяемых табличным пространствам по умолчанию. Связанные с базой данных объекты не всегда хранятся в каталоге базы данных; они могут храниться на различных местах, в частности, на различных устройствах.

База данных создается в экземпляре, определяемом переменной среды DB2INSTANCE, или в экземпляре, к которому вы явно подключились при помощи команды ATTACH. Введение в использование экземпляров смотрите в разделе “Использование нескольких экземпляров менеджера баз данных” книги *Administration Guide: Implementation*.

В системах на основе UNIX используется следующая схема именования:

```
заданный_путь/$DB2INSTANCE/NODEnnnn/SQL00001
```

В операционных системах OS/2 и Windows используется следующая схема именования:

```
D:\$DB2INSTANCE\NODEnnnn\SQL00001
```

где

- заданный\_путь - необязательное заданное пользователем место установки экземпляра.

- NODEnnnn - идентификатор узла в среде распределенных баз данных. Первый узел - NODE0000.
- "D:" - буква диска, задающая том, на котором расположен корневой каталог.

SQL00001 содержит объекты, связанные с первой созданной базой данных; последующим базам данных даются последовательные номера: SQL00002 и так далее.

Эти подкаталоги создаются в каталоге с именем, совпадающим с именем экземпляра менеджера баз данных, к которому вы были подключены при создании базы данных. (В OS/2 и операционных системах Windows эти подкаталоги создаются в корневом каталоге тома, обозначенного буквой диска.) Эти подкаталоги экземпляров и баз данных создаются в пути, указанном в команде CREATE DATABASE, и менеджер баз данных управляет ими автоматически. В зависимости от платформы каждый экземпляр может принадлежать владельцу экземпляра, имеющему полномочия системного администратора (SYSADM) для базы данных данного экземпляра.

Чтобы избежать возможных проблем, не создавайте каталогов с той же схемой именования и не производите никаких действий с каталогами, созданными менеджером баз данных.

## Файлы баз данных

С базой данных связаны следующие файлы:

<b>Имя файла</b>	<b>Описание</b>
<b>SQLDBCON</b>	В этом файле хранятся параметры и флаги настройки базы данных. Информацию об изменении параметров конфигурации базы данных смотрите в книге <i>Administration Guide: Performance</i> .

### **SQLOGCTL.LFH**

Этот файл помогает отслеживать все файлы журнала базы данных и управлять ими.

<b>Syyyyyy.LOG</b>	Файлы журнала базы данных, пронумерованные от 0000000 до 9999999. Количество этих файлов определяется параметрами конфигурации базы данных <i>logprimary</i> и <i>logsecond</i> . Размер отдельных файлов определяется параметром конфигурации базы данных <i>logfilsiz</i> .
--------------------	---

При циклической регистрации файлы с теми же номерами используются повторно. При архивной регистрации номера файлов увеличиваются по мере того, как журналы архивируются и выделяются новые журналы. При достижении 9999999 нумерация снова начинается с нуля.

По умолчанию эти файлы журнала хранятся в каталоге SQLOGDIR. SQLOGDIR находится в подкаталоге SQLnnnnn.

<b>SQLINSLK</b>	Этот файл помогает обеспечить использование базы данных только одним экземпляром менеджера баз данных.
<b>SQLTMPLK</b>	Этот файл помогает обеспечить использование базы данных только одним экземпляром менеджера баз данных.
<b>SQLSPCS.1</b>	В этом файле содержится определение и текущее состояние всех табличных пространств в базе данных.
<b>SQLSPCS.2</b>	Этот файл - резервная копия SQLSPCS.1. При отсутствии обоих этих файлов доступ к базе данных невозможен.
<b>SQLBP.1</b>	В этом файле содержится определение всех пулов буферов, используемых в базе данных.
<b>SQLBP.2</b>	Этот файл - резервная копия SQLBP.1. При отсутствии обоих этих файлов доступ к базе данных невозможен.
<b>DB2RHIST.ASC</b>	Это файл хронологии базы данных. В нем хранится хронология управляющих операций, производившихся над базой данных, например, резервное копирование и восстановление из резервной копии.
<b>DB2RHIST.BAK</b>	Этот файл - резервная копия DB2RHIST.ASC.

#### Примечания:

1. Не изменяйте эти файлы непосредственно. Обращаться к ним можно только через документированные API и инструменты, использующие эти API, включая процессор командной строки и Центр управления DB2.
2. Не перемещайте эти файлы.
3. Не удаляйте эти файлы.
4. Единственный поддерживаемый способ резервного копирования базы данных или табличного пространства - через API **sqlubkp** (Резервное копирование базы данных), а также через использующие этот API процессор командной строки и Центр управления DB2.

---

## Оценка пространства, требуемого для таблиц

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 приблизительно 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. Информацию об этой утилите смотрите в руководстве *Command Reference*.

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" на стр. 95
- "Данные длинных полей" на стр. 97
- "Данные больших объектов" на стр. 97
- "Индексное пространство" на стр. 98

Здесь не рассматриваются требования размера, 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

## Таблицы 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 экстенда табличного пространства, содержащего таблицы каталога. Например, если используется табличное пространство DMS с размером экстенда 32, для табличного пространства каталога будет выделены 20 Мбайт. Дополнительную информацию смотрите в разделе “Проектирование и выбор табличных пространств” на стр. 111.

**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.

## Данные пользовательских таблиц

По умолчанию табличные данные хранятся на страницах по 4 Кбайта. Каждая страница (независимо от ее размера) содержит 76 дополнительных байтов для менеджера баз данных. Для хранения пользовательских данных (или строк) останется 4020 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 VARCHARIC, BLOB, CLOB и DBCLOB. В строках в странице табличных данных содержится дескриптор для этих столбцов. (Информацию об оценке требований размера для табличных объектов, содержащих данные этих типов, смотрите в разделах “Данные длинных полей” на стр. 97 и “Данные больших объектов” на стр. 97.)

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. Дополнительную информацию об этом операторе смотрите в справочнике *SQL Reference*.

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

$$\text{число\_записей\_на\_страницу} = \text{ОКРУГЛИТЬ В МЕНЬШУЮ СТОРОНУ}(4020 / (\text{средний размер строки} + 10))$$

и

$\text{количество\_страниц} = (\text{количество\_записей}/\text{записей\_на\_страницу}) * 1,1$

где средний размер строки - сумма средних размеров столбцов (информацию о размере отдельных столбцов смотрите в руководстве *SQL Reference*, оператор CREATE TABLE), 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 Кбайта длина строки может достигать 4005 байт.
- При размере страницы 8 Кбайт длина строки может достигать 8101 байта.
- При размере страницы 16 Кбайт длина строки может достигать 16293 байт.
- При размере страницы 32 Кбайта 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. Дополнительную информацию об импорте данных в таблицы и о файлах данных IXF смотрите в руководстве *Data Movement Utilities Guide and Reference*.

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.

## **Данные длинных полей**

Данные длинных полей хранятся в отдельном табличном объекте, структура которого отличается от других типов данных (смотрите разделы "Данные пользовательских таблиц" на стр. 95 и "Данные больших объектов").

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 VARCHARIC) хранятся так, чтобы упростить повторное использование свободного места. Информация о выделенном и свободном месте хранится на страницах размещения по 4 Кбайта, периодически вставляемых между страниц 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 VARCHARIC следует использовать CHAR, GRAPHIC, VARCHAR или VARCHARIC.

## **Данные больших объектов**

Данные больших объектов хранятся в двух отдельных табличных объектах, структура которых отличается от остальных типов данных (смотрите разделы "Данные пользовательских таблиц" на стр. 95 и "Данные длинных полей").

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 данных 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`.

## Индексное пространство

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

$$(\text{средний размер ключа индекса} + 8) * \text{количество строк} * 2$$

где:

- "Средний размер ключа индекса" - это число байт в каждом столбце из ключа индекса. Информацию о том, как вычислить число байт для столбцов с различными типами данных, смотрите в описании оператора `CREATE TABLE` в справочнике *SQL Reference*. (При оценке среднего размера столбца для столбцов `VARCHAR` и `VARGRAPHIC` используйте текущий средний размер данных плюс один байт. 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:

$$(\text{средний размер ключа индекса} + 8) * \text{количество строк} * 3,2$$



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 четыре байта. 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 экстенд.

- Грубая оценка среднего числа ключей на конечную страницу:

$$\frac{(0,9 * (U - (M*2))) * (D + 1)}{K + 6 + (4 * D)}$$

где:

- U - доступное место на странице, приблизительно равно размеру страницы минус 100. Для страницы размером 4096 U равно 3996.
- $M = U / (8 + \text{минимальныйРазмерКлюча})$
- D = среднее число дубликатов на значение ключа
- K = *среднийРазмерКлюча*

Помните, что к значениям *минимальныйРазмерКлюча* и *среднийРазмерКлюча* надо добавить по одному байту 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  $(100 - \text{процент\_свободного\_места})/100$ .

- Более точная оценка среднего числа ключей на конечную страницу:  
 $L = \text{число конечных страниц} = X / (\text{среднее число ключей на конечную страницу})$

где X - общее число строк в таблице.

Исходный размер индекса оценивается примерно так:

$$(L + 2L / (\text{среднее число ключей на конечную страницу})) * \text{размер\_страницы}$$

Для табличных пространств DMS сложите размеры всех индексов для данной таблицы и округлите результат в большую сторону до ближайшего числа, кратного размеру экстенда 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:

$$\frac{(0,9 * (U - (M*2))) * (D + 1)}{K + 12 + (8 * D)}$$

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 / (8 + *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:

$$T * \text{размер\_страницы}$$

## Дополнительные требования к пространству

Необходимо также следующее дополнительное пространство:

- “Пространство для файлов журнала”
- “Временное рабочее пространство” на стр. 102

### Пространство для файлов журнала

Объем пространства (в байтах), необходимого для файлов журнала, колеблется от

$$( \text{число\_первичных} * ( \text{размер\_файла\_журнала} + 2 ) * 4096 ) + 8192$$

до:

$$( ( \text{число\_первичных} + \text{число\_вторичных} ) * ( \text{размер\_файла\_журнала} + 2 ) * 4096 ) + 8192$$

где:

- *число\_первичных* - число первичных файлов журнала, определенное в файле конфигурации базы данных
- *число\_вторичных* - число вторичных файлов журнала, определенное в файле конфигурации базы данных
- *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
- 8192 - размер (в байтах) файла управления журналом.

Дополнительную информацию о параметрах конфигурации *logprimary*, *logsecond* и *logfilsiz* смотрите в книге *Administration Guide: Performance*.

**Примечание:** Общее пространство активного журнала не должно превышать 32 Гбайт.

Верхний предел пространства файлов журнала зависит от фактического числа вторичных файлов журнала, требуемых для работы менеджера баз данных. Этот предел может никогда не потребоваться или требоваться только в отдельные периоды повышенной активности.

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.

## Временное рабочее пространство

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.

---

## Проектирование групп узлов

*Группа узлов* - это именованное множество из одного или нескольких узлов, определенных как принадлежащие некоторой базе данных. Все разделы базы данных, являющиеся частью конфигурации системы базы данных, должны быть заранее определены в *файле конфигурации разделов*, который называется

db2nodes.cfg. Группа узлов может содержать от одного раздела базы данных до всего множества разделов базы данных, определенных для системы базы данных.

Новая группа узлов создается при помощи оператора CREATE NODEGROUP; ее можно изменить при помощи оператора ALTER NODEGROUP. В группе узлов можно добавить или отбросить один или несколько разделов базы данных. До изменения группы узлов разделы базы данных должны быть определены в файле db2nodes.cfg. Внутри групп узлов находятся табличные пространства. Внутри табличных пространств находятся таблицы.

При создании или изменении группы узлов 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 узлов будет храниться заданная строка данных. Дополнительную информацию о картах разделения смотрите в разделе “Карты разделения” на стр. 105. Дополнительную информацию о ключах разделения смотрите в разделе “Ключи разделения” на стр. 107.

В однораздельной базе данных не требуется ни ключа разделения, ни карты разделения. При использовании однораздельной базы данных нет необходимости проектировать группы узлов. *Раздел базы данных* - это часть базы данных, заполненная пользовательскими данными, индексами, файлами конфигурации и журналами транзакций. Менеджер баз данных использует группы узлов, созданные по умолчанию при создании базы данных. IBMGROUP - группа узлов по умолчанию для табличного пространства, содержащего системные каталоги. IBMTEMPGROUP - группа узлов по умолчанию для системных временных табличных пространств. IBMDEFAULTGROUP - группа узлов 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 узлов, созданной пользователем, но не в IBMTEMPGROUP.

Если вы используете многораздельную группу узлов, учтите следующие особенности проектирования:

- В многораздельной группе узлов индекс уникальности можно создать, только если он является надмножеством ключа разделения.
- В зависимости от числа разделов в базе данных может быть одна или несколько однораздельных групп узлов и одна или несколько многораздельных групп узлов.
- Каждому разделу базы данных должен быть назначен уникальный номер раздела. Один раздел базы данных может находиться в одной или в нескольких группах узлов.

- 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 узлов, в которые не включен раздел базы данных в группе узлов IBMCATGROUP.

Небольшие таблицы следует помещать в однораздельные группы узлов, 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 однораздельной группе узлов. Таблицы считаются совместно размещенными, если они находятся в многораздельной группе узлов, 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 узлов; 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 узлов из 16 разделов, чем в группе узлов из 32 разделов.

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

## **Особенности проектирования групп узлов**

Необходимость в разделении базы данных зависит от ее логической структуры и объема обрабатываемых данных. В этом разделе описываются следующие темы, связанные с разделением базы данных:

- “Разделение данных”
- “Карты разделения” на стр. 105
- “Ключи разделения” на стр. 107
- “Совместное размещение таблиц” на стр. 109
- “Совместимость с разделами” на стр. 109
- “Реплицируемые сводные таблицы” на стр. 110

### **Разделение 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 узлов, 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.

Не обязаны разделять все таблицы между всеми разделами в базе данных. DB2 поддерживает *частичное рассеяние*, что значит, что таблицы и их табличные пространства можно распределить по подмножеству разделов базы данных в системе (то есть по группе узлов).

Второй возможный вариант для размещения таблиц на каждом разделе базы данных - использование таблиц сводок и последующая их репликация. Вы можете создать таблицу сводок, содержащую всю необходимую информацию, а затем реплицировать ее на все узлы. Дополнительную информацию смотрите в разделе “Реплицируемые сводные таблицы” на стр. 110.

### **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 узлов или один элемент для однораздельных групп узлов. Для однораздельной группы узлов в карте разделения есть только один элемент, содержащий номер раздела базы данных, в котором хранятся все строки таблицы базы данных. Для многораздельных групп узлов циклически задаются номера разделов группы узлов. 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 узлов IBMDEFAULTGROUP этой базы данных будет иметь вид:

0 1 2 3 0 1 2 ...

Если создать в этой базе данных группу узлов, использующую разделы 1 и 2, карта разделения для этой группы узлов 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. 27 показана строка со значением ключа разделения (с1, с2, с3), отображаемая в раздел 2, который, в свою очередь, указывает на раздел базы данных n5.

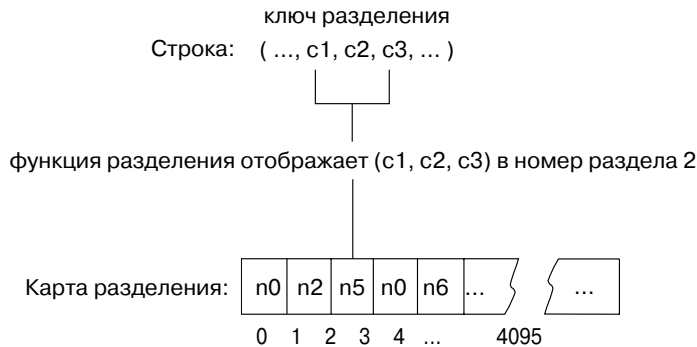


Рисунок 27. Распределение данных при помощи карты разделения

Карта разделения - это гибкий способ управления местом хранения 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. Дополнительную информацию об этой утилите смотрите в разделе "Перераспределение данных между разделами базы данных" книги *Administration Guide: Performance*.

Чтобы получить для просмотра копию карты разделения, можно воспользоваться API получения информации о разделении таблицы (**sqlugtpi**). Дополнительную информацию об этом API смотрите в руководстве *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 узлов, 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 однораздельной группой узлов, 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 однораздельных группах узлов. Ключи разделения можно добавить или отбросит позднее при помощи оператора ALTER TABLE. Изменение ключа разделения допустимо только для таблицы, табличное пространство которой связано с однораздельной группой узлов.

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 узлов. Совместное размещение таблиц определяется ключом разделения для каждой таблицы в табличном пространстве, связанном с группой узлов. Таблицы считаются совместно размещенными, если:

- Таблицы расположены в табличных пространствах, находящихся в одной группе узлов
- 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. Дополнительную информацию о совместимости с разделами смотрите в разделе “Совместимость с разделами” на стр. 109. Дополнительную информацию о совместном размещении таблиц смотрите в разделе “Совместное размещение таблиц” на стр. 109.

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 узлов. 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 VARCHARIC, 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.

Если переменная реестра DB2\_UPDATE\_PART\_KEY имеет значение NO, вы не можете изменить значение столбца, входящего в ключ разделения, в строке таблицы.

*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 узлов создается карта разделения. Карта разделения циклически заполняется последовательным перебором номеров разделов. Дополнительную информацию о картах разделения смотрите в разделе “Карты разделения” на стр. 105.
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.

### **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 узлов и их ключи разделения совместимы. Если несколько таблиц помещены в одну группу узлов, для них используется общая карта разделения. Таблицы могут находиться в разных табличных пространствах, но они должны быть связаны с одной группой узлов. Типа данных связанных столбцов во всех ключах разделения должны быть *совместимы с разделами*. Информацию о совместимости с разделами смотрите в разделе “Совместимость 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. Дополнительную информацию смотрите в разделе “Совместно размещенные объединения” книги *Administration Guide: Performance*.

### **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.

### **Реплицируемые сводные таблицы**

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

В среде многораздельных баз данных сводные таблицы можно реплицировать. *Реплицируемые сводные таблицы* можно использовать для улучшения производительности запросов. Реплицируемая сводная таблица основана на таблице, которая может быть создана в однораздельной группе узлов, но вы хотите реплицировать ее по всем разделам базы данных в этой группе узлов. Чтобы создать реплицируемую сводную таблицу, укажите в операторе CREATE TABLE ключевое слово REPLICATED.

Подробную информацию о сводных таблицах смотрите в разделе "Создание сводных таблиц" книги *Administration Guide: Implementation*.

При помощи реплицируемых сводных таблиц можно добиться совместного размещения таблиц, для которых обычно это невозможно. Реплицируемые сводной таблицы 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.

Дополнительную информацию о реплицируемых сводных таблицах смотрите в описании оператора CREATE TABLE в справочнике *SQL Reference*.  
Дополнительную информацию о совместно размещенных объединениях смотрите в разделе "Совместные объединения" книги *Administration Guide: Implementation*.

---

## Проектирование и выбор табличных пространств

Табличное пространство - это модель хранения, соответствующая промежуточному уровню между базой данных и таблицами, хранящимися в этой базе данных. Табличные пространства находятся в группах узлов. Они позволяют напрямую назначать контейнерам местонахождение базы данных и табличных данных. (Контейнером может быть каталог, устройство или файл.) Это может улучшить производительность и целостность, а также сделать конфигурацию более гибкой.

Дополнительную информацию о создании и изменении табличного пространства смотрите в разделе "Создание табличного пространства" и в разделе "Изменение табличного пространства" книги *Administration Guide: Implementation*.

Поскольку табличные пространства размещаются в группах узлов, выбранное для хранения таблицы табличное пространство определяет, как данные для этой таблицы распределяются по разделам базы данных в данной группе узлов. 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. 28 на стр. 112 показано, как таблицы и табличные пространства в базе данных связаны с контейнерами, относящимися к этой базе данных.

## База данных

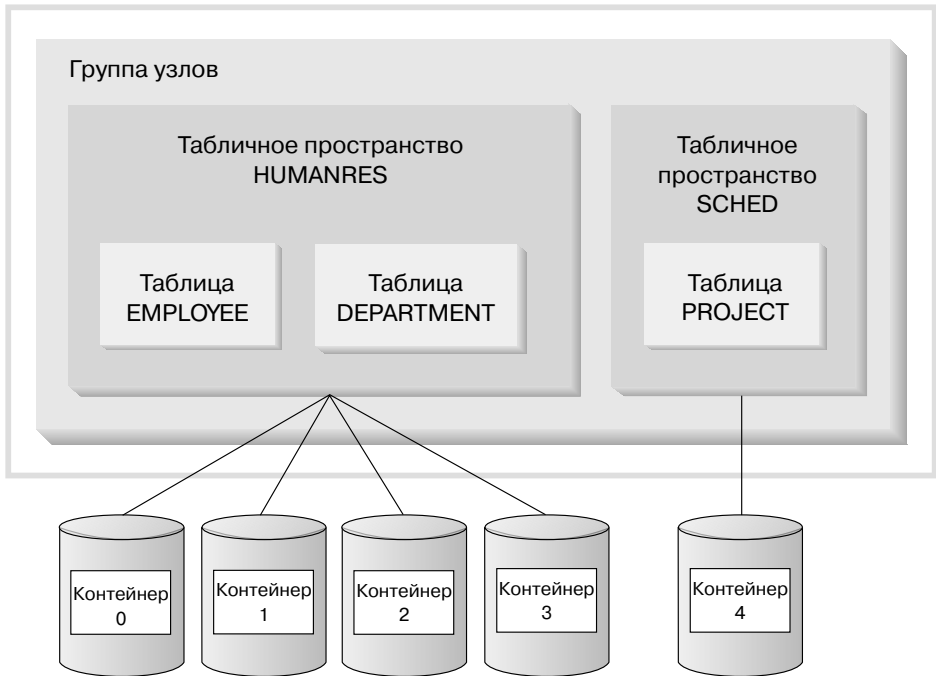


Рисунок 28. Табличные пространства и таблицы в базе данных

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 уравнивать 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. 29 на стр. 113 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.

## Табличное пространство HUMANRES

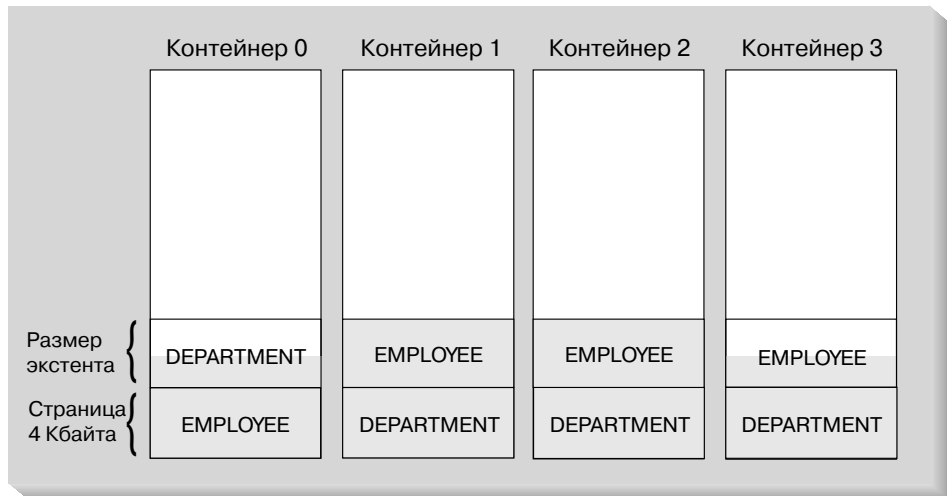


Рисунок 29. Контейнеры и экстенды

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 узлов по умолчанию для этого табличного пространства.
- Одно или несколько *пользовательских табличных пространств*, содержащие все пользовательские таблицы. По умолчанию создается одно табличное пространство - USERSPACE1. IBMDEFAULTGROUP - группа узлов по умолчанию для этого табличного пространства.

При создании таблицы необходимо указывать имя табличного пространства, иначе результаты могут отличаться от ожидаемых. Если не указать имени табличного пространства, таблица будет размещена в соответствии со следующими правилами: Если существуют табличные пространства, созданные пользователем, из них выбирается табличное пространство с наименьшим размером страницы, достаточно большим для этой таблицы. В противном случае будет использовано USERSPACE1, если его размер страницы достаточно велик для этой таблицы. Если нет пользовательских табличных пространств с достаточным размером страницы, таблица не создается.

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*, содержащие 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 узлов 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 пространстве с подходящим размером страницы. Если есть несколько временных табличных пространств с таким размером страницы, табличные пространства будут выбираться циклически.

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. Смотрите также раздел “Рекомендации для временных табличных пространств” на стр. 128.

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:

- “SMS - Пространство, управляемое системой” на стр. 115: Пространство хранения управляется менеджером файлов операционной системы.



- “Табличное пространство, управляемое базой данных” на стр. 119:  
Пространство хранения управляется менеджером баз данных.

## SMS - 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 используется табличное пространство SMS.

С каждой таблицей связан по меньшей мере один физический файл SMS. Список этих файлов и описание их содержимого смотрите в разделе “Физические файлы SMS” на стр. 118.

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. 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. Дополнительную информацию о **db2empfa** смотрите в руководстве *Command Reference*.

Табличные пространства SMS нужно определять в явном виде при помощи 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 узлов 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.

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

- Размер экстенда для табличного пространства.

Размер экстенда можно задать только при создании табличного пространства. Поскольку размер экстенда нельзя изменить позже, важно выбрать для него подходящее значение. Дополнительную информацию смотрите в разделе “Выбор размера экстенда” на стр. 127.

Если при создании табличного пространства не указать размера экстенда, менеджер баз данных создаст табличное пространство с размером экстенда по умолчанию, заданным параметром конфигурации базы данных *dft\_extent\_sz* (дополнительную информацию об этом параметре смотрите в книге *Administration Guide: Performance*). 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 экстенда будет по умолчанию равен 32.

Чтобы выбрать для табличного пространства подходящие значения количества контейнеров и размера экстенда, 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 экстенда. 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 (SQLnnnnn.INX), длинных полей (SQLnnnnn.LF) и LOB (SQLnnnnn.LB и SQLnnnnn.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.

Дополнительную информацию о файлах, используемых в табличном пространстве SMS, смотрите в разделе “Физические файлы SMS” на стр. 118.

## Физические файлы SMS

В контейнере каталога табличного пространства SMS находятся следующие файлы:

Имя файла	Описание
-----------	----------

<b>SQLTAG.NAM</b>	
-------------------	--

В каждом подкаталоге контейнера есть этот файл; они используются менеджером баз данных при соединении с базой данных для проверки ее целостности.

<b>SQLxxxx.DAT</b>	
--------------------	--

Табличный файл. Здесь хранятся все строки таблицы, за исключением данных типов LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB и DBCLOB.

<b>SQLxxxx.LF</b>	
-------------------	--

Файл, содержащий данные типов LONG VARCHAR и LONG VARGRAPHIC (называемые также "данными длинных полей"). Этот файл создается, только если в таблице есть столбцы LONG VARCHAR или LONG VARGRAPHIC.

<b>SQLxxxx.LB</b>	
-------------------	--

Файлы, содержащие данные типов BLOB, CLOB и DBCLOB (называемые также "данными больших объектов"). Эти файлы создаются, только если в таблице есть столбцы BLOB, CLOB или DBCLOB.

<b>SQLxxxx.LBA</b>	
--------------------	--

Файлы, содержащие информацию о размещении и свободном месте в файлах SQLxxxx.LB.

<b>SQLxxxx.INX</b>	
--------------------	--

Файл индекса для таблицы. Все индексы для соответствующей таблицы хранятся в этом файле. Он создается, только если определены какие-нибудь индексы.

**Примечание:** При отбрасывании индекса занимаемое им пространство физически не освобождается из файла индекса (.INX), пока файл индекса не будет удален. Файл индекса удаляется при отбрасывании индексов для данной таблицы (и принятии изменений) или при реорганизации таблицы. Если файл индекса не удаляется, после принятия отбрасывания индекса пространство будет помечено в качестве свободного и затем повторно использовано для будущих создаваемых индексов или для работы с индексами.

<b>SQLxxxx.DTR</b>	
--------------------	--

Временный файл данных для реорганизации файла DAT. При реорганизации таблицы утилита georg (посредством команды REORG TABLE) создает таблицу в одном из системных

временных табличных пространств. Можно задать использование для этих временных табличных пространств вместо контейнеров, используемых для пользовательских таблиц, других контейнеров.

#### **SQLxxxxx.LFR**

Временный файл данных для реорганизации файла LF. При реорганизации таблицы утилиты georg (посредством команды REORG TABLE) создает таблицу в одном из системных временных табличных пространств. Можно задать использование для этих временных табличных пространств вместо контейнеров, используемых для пользовательских таблиц, других контейнеров.

#### **SQLxxxxx.RLB**

Временный файл данных для реорганизации файла LB. При реорганизации таблицы утилиты georg (посредством команды REORG TABLE) создает таблицу в одном из системных временных табличных пространств. Можно задать использование для этих временных табличных пространств вместо контейнеров, используемых для пользовательских таблиц, других контейнеров.

#### **SQLxxxxx.RBA**

Временный файл данных для реорганизации файла LBA. При реорганизации таблицы утилиты georg (посредством команды REORG TABLE) создает таблицу в одном из системных временных табличных пространств. Можно задать использование для этих временных табличных пространств вместо контейнеров, используемых для пользовательских таблиц, других контейнеров.

#### **Примечания:**

1. *Не* изменяйте эти файлы непосредственно. Обращаться к ним можно только через документированные API и инструменты, использующие эти API, включая процессор командной строки и Центр управления DB2.
2. Не перемещайте эти файлы.
3. Не удаляйте эти файлы.
4. Единственный поддерживаемый способ резервного копирования базы данных или табличного пространства - через API **sqlubkp** (Резервное копирование базы данных), а также через использующие этот API процессор командной строки и Центр управления DB2.

### **Табличное 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,

пространство на которых управляется DB2. Администратор решает, какие устройства использовать, и DB2 управляет пространством на этих устройствах. В сущности, такое табличное пространство - реализация файловой системы специального назначения, разработанной для наилучшего соответствия нуждам менеджера баз данных. В определение табличного пространства входит список принадлежащих этому табличному пространству устройств или файлов, в которых можно хранить данные.

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

- *Обычное* табличное пространство для хранения нормальных данных таблиц и индексов
- *Длинное* табличное пространство для хранения данных длинных полей и больших объектов.

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 длинных 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.
- В каждом контейнере зарезервирована одна дополнительная страница, а остальные таблицы будут использоваться по одному экстенду за раз. Используются только целые экстенды, так что для наилучшего управления пространством можно при выделении контейнера определять подходящий размер при помощи следующей формулы:

$$(\text{размер\_экстенда} * n) + 1$$

где *размер\_экстента* - размер каждого экстента в этом табличном пространстве, а *n* - число экстенентов, которое вы хотите хранить в этом контейнере.

- В табличном пространстве резервируются три дополнительных экстенента.
- Для хранения любых данных пользовательских таблиц требуются по меньшей мере два экстенента. (Эти экстененты нужны для обычных данных таблицы, а не для данных индексов, длинных полей или больших объектов, для которых нужны отдельные экстененты.)
- 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.

### **Добавление контейнеров в табличные пространства DMS**

Оператор ALTER TABLESPACE позволяет добавить контейнер в существующее табличное пространство, чтобы увеличить его. После этого табличное пространство перебалансируется по всем контейнерам. Во время перебалансировки доступ к табличному пространству не ограничивается. Если вам нужно добавить несколько контейнеров, их следует добавить в этот момент, либо в одном операторе ALTER TABLESPACE, либо в той же транзакции, чтобы менеджеру баз данных не приходилось перебалансировать контейнеры несколько раз.

Проверить, насколько заполнены контейнеры для табличного пространства, можно при помощи команд LIST TABLESPACE CONTAINERS или LIST TABLESPACES. Добавлять новые контейнеры следует до того, как существующие контейнеры заполнятся почти до конца или же до конца. Новое место на всех контейнерах будет недоступно, пока не завершится перебалансировка.

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.

## **Особенности структуры табличных пространств**

В этой главе рассматриваются следующие темы:

- “Особенности ввода/вывода для табличных пространств”
- “Отображение табличных пространств на пулы буферов” на стр. 124
- “Отображение табличных пространств на группы узлов” на стр. 125
- “Отображение таблиц в табличные пространства” на стр. 125
- “Выбор размера экстенда” на стр. 127
- “Рекомендации для временных табличных пространств” на стр. 128
- “Рекомендации по табличным пространствам каталога” на стр. 130
- “Особенности загрузки” на стр. 130
- “Выбор между табличными пространствами SMS и DMS” на стр. 132
- “Повышение производительности при размещении данных на устройствах RAID” на стр. 133.

## **Особенности ввода/вывода для табличных пространств**

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 - экстенд). 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.

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.

Степень предварительной выборки можно управлять при помощи параметра PREFETCHSIZE оператора CREATE 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, например, 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 экстенда, 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. Нужно определить по меньшей мере по одному чистильщику страниц для каждого реально используемого базой данных диска. Дополнительную информацию по этим темам смотрите в книге *Administration Guide: Performance*.

### **Отображение табличных пространств на пулы 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); 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. Для табличных пространств с одной или несколькими большими таблицами, 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. Дополнительную информацию на эту тему смотрите в разделе "Управление пулом буферов базы данных" книги *Administration Guide: Performance*.

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

### **Отображение табличных пространств на группы узлов**

В среде многораздельных баз данных каждое табличное пространство связано с некоторой группой узлов. Это позволяет применить характеристики табличного пространства к каждому узлу в этой группе узлов. Группа узлов должна существовать (определяется оператором CREATE NODEGROUP); тогда связь между табличным пространством и группой узлов определяется при создании табличного пространства оператором CREATE TABLESPACE.

Связь между табличным пространством и группой узлов нельзя изменить при помощи оператора ALTER TABLESPACE. Можно только изменять задание табличного пространства для отдельных разделов этой группы узлов. В среде однораздельных баз данных все табличные пространства связаны с группой узлов по умолчанию. Группа узлов по умолчанию при определении табличного пространства называется IBMDEFAULTGROUP, если определяется не системное временное табличное пространство; в противном случае используется IBMTEMPGROUP. Дополнительную информацию об операторе CREATE NODEGROUP смотрите в справочнике *SQL Reference*. Дополнительную информацию о группах узлов и физической структуре базы данных смотрите в разделе “Проектирование групп узлов” на стр. 102.

### **Отображение таблиц в табличные пространства**

При определении, как отображать таблицы в табличные пространства, необходимо учитывать:

- Разделение ваших таблиц.

Как минимум, следует убедиться, что выбранное табличное пространство находится в группе узлов с нужным разделением.

- Количество данных в таблице.

Если вы собираетесь хранить в табличном пространстве много маленьких таблиц, подумайте об использовании табличного пространства SMS. Преимущества SMS 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 экстенда.

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 экстенда в зависимости от использования табличного пространства.

(Дополнительную информацию смотрите в разделе “Выбор размера экстенда” на стр. 127.)

- Тип данных в таблице.

Допустим, у вас есть таблицы с редко используемыми хронологическими данными; возможно, конечный пользователь готов терпеть большее время ответа для запросов к этим данным. 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.

### **Выбор размера 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.

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.

Все табличные объекты хранятся отдельно, и для каждого объекта по мере необходимости выделяются новые экстененты. Для каждого табличного объекта имеется объект метаданных, называемый *картой экстенентов*, который описывает все экстененты в этом табличном пространстве, принадлежащие объекту таблицы. Место для карт экстенентов 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. (Информацию о предварительной выборке данных и о ее связи с размером экстента смотрите в книге *Administration Guide: Performance*.)

- Минимально необходимое число экстентов.

Если в контейнерах не хватает места для пяти экстентов табличного пространства, табличное пространство не создается.

### **Рекомендации для 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 "на месте", то есть прямо в табличном пространстве назначения. Конечно, для реорганизации "на месте" необходимо, чтобы в табличном пространстве назначения было достаточно места.

Дополнительную информацию о реорганизации таблиц смотрите в книге *Administration Guide: Performance*.

- 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 8 Кбайт с большим пулом буферов и одно табличное пространство 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 табличного пространства 4 Кбайта, чтобы выполнять реорганизации 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:
  - 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.

- Контейнеры SMS могут использовать буферизацию файловой системы; контейнеры DMS этого не могут.

### **Рекомендации по табличным пространствам 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 экстенда. В зависимости от выбранного размера экстенда можно получить значительный объем выделенного, но неиспользуемого места. При использовании табличного пространства DMS следует выбирать маленький размер экстенда (от двух до четырех страниц); в противном случае используйте табличное пространство SMS.
- В таблицах каталога есть столбцы больших объектов. 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 собственное место для хранения (или кэширования) данных, 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.

### **Особенности нагрузки**

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 и которые обычно возвращают небольшие наборы данных. Если доступ нерегулярный и запрашивается одна или несколько страниц, предварительная выборка невозможна.

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.

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 из всех контейнеров.

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:

$$\text{размер\_страницы} / 255$$

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. Эти ограничения описаны в справочнике *SQL Reference*.

### **Выбор между табличными пространствами 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 базы данных требуется меньше подготовительных усилий, так как вам не требуется определять контейнеры заранее.

#### *Достоинства табличного пространства DMS:*

- Табличное пространство можно расширять, добавляя контейнеры с помощью оператора ALTER TABLESPACE. Существующие данные автоматически перебалансируются по новому набору контейнеров, что сохраняет максимальную эффективность ввода/вывода.
- Таблицу можно разбить на несколько табличных пространств в зависимости от типа хранимых данных:
  - Данные длинных полей и больших объектов
  - Индексы
  - Обычные данные

Это бывает нужно из соображений производительности или увеличения объема данных, которые можно хранить 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 Кбайт, 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 и PTX (IBM NUMA-Q) 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. Дополнительную информацию смотрите в разделе "Особенности производительности для устройств DMS" книги *Administration Guide: Performance*.

### **Повышение производительности при размещении данных на устройствах RAID**

В этом разделе описывается, как повысить производительность, если данные находятся на устройствах RAID (Redundant Array of Independent Disks - дублированный массив независимых дисков). Обычно 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 (смотрите раздел "DB2\_PARALLEL\_IO").
- Чтобы в табличном пространстве выровнялись границы экстенгов, используйте переменную реестра DB2\_STRIPED\_CONTAINERS (смотрите раздел "DB2\_STRIPED\_CONTAINERS" на стр. 134).

### **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 параллельный ввод/вывод для всех
                                     табличных пространств}
db2set DB2_PARALLEL_IO=1,2,4,8     {включает параллельный ввод/вывод для
                                     табличных пространств 1, 2, 4 и 8}
```

Чтобы изменения вступили в силу, после задания переменной реестра нужно остановить DB2 (**db2stop**) и запустить снова (**db2start**).

## DB2\_STRIPED\_CONTAINERS

В настоящее время при создании контейнера табличного пространства DMS (устройства или файла) в начало контейнера записывается односторонняя метка. Остальные страницы доступны DB2 для хранения данных и сгруппированы в блоки размером по одному экстенду.

При использовании в качестве контейнеров табличных пространств устройств RAID рекомендуется создавать табличное пространство с размером экстендов, равным или кратным размеру полосы RAID. Однако из-за односторонней метки контейнера экстенды не будут совпадать с полосами RAID, и при требовании ввода/вывода может понадобиться доступ к большему числу физических дисков, чем было бы оптимально.

Контейнеры табличных пространств DMS можно создавать так, чтобы метка хранилась в отдельном (полном) экстенде. Это позволяет избежать описанной проблемы, но требует выделения лишнего экстенда внутри контейнера. Чтобы создавать такие контейнеры, нужно задать для переменной реестра DB2 DB2\_STRIPED\_CONTAINERS значение "ON", а затем остановить и перезапустить экземпляр:

```
db2set DB2_STRIPED_CONTAINERS=ON
db2stop
db2start
```

При создании любого контейнера DMS (при помощи операторов CREATE TABLESPACE или ALTER TABLESPACE) метки будут занимать целый экстенд. Уже существующие контейнеры не будут изменены.

Чтобы прекратить создавать контейнеры с этим атрибутом, сбросьте значение переменной (значение по умолчанию - NULL), а затем остановите и перезапустите экземпляр:

```
db2set DB2_STRIPED_CONTAINERS=
db2stop
db2start
```

Центр управления и команда LIST TABLESPACE CONTAINERS не показывают, был ли контейнер создан как многополосный. Они показывают только метку "файл" или "устройство", в зависимости от того, как был создан контейнер. Чтобы проверить, был ли контейнер создан как многополосный, 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.

---

## Особенности проектирования объединенных баз данных

При проектировании базы данных объединения следует учитывать следующее:

- Требования к пространству
- Задание приоритетов сети.

Обычно данные, доступные из базы данных объединения, не хранятся в этой базе данных. В системном каталоге хранятся ссылки на таблицы и производные таблицы источника данных, а фактические данные находятся на источнике данных. По существу, для базы данных объединения может потребоваться меньше места, чем обычной. Это общее правило может не действовать, если ваши запросы в связи с различиями системы сортировки или с отсутствием функции на источнике данных приходится выполнять локально. В этом случае таблицы материализуются DB2 для обработки.

Поскольку большинство данных системы объединения обычно находится на одном или нескольких источниках данных, находящихся в сети, возможно, следует изменить ресурсы, назначенные DB2 и системе в сети. Если на системе DB2 выделить сети больше ресурсов, чем собственно менеджеру баз данных, можно повысить производительность.



---

## Глава 9. Проектирование распределенных баз данных

В DB2 транзакции обычно называют *единицами работы*. Единица работы - это восстанавливаемая 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 со 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.

Дополнительную информацию смотрите в следующих темах:

- “Использование в транзакции одной базы данных” на стр. 138
- “Использование в транзакции нескольких баз данных” на стр. 139
- “Другие особенности конфигурирования” на стр. 145
- “Процесс двухфазного принятия” на стр. 148
- “Восстановление после ошибок двухфазного принятия” на стр. 151.

Информацию о создании прикладных программ для работы с распределенными базами данных смотрите в руководствах *Application Development Guide* и *CLI Guide and Reference*.

## Использование 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*.

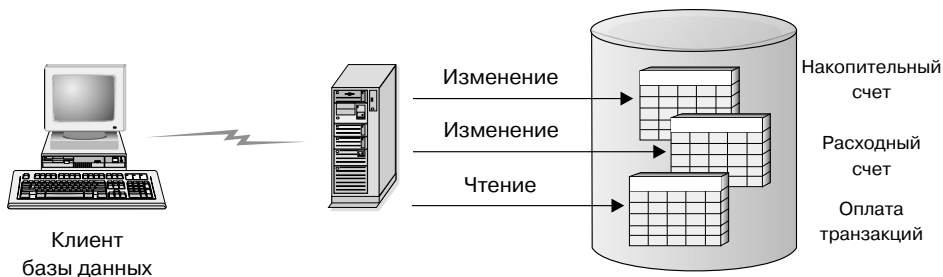


Рисунок 30. Использование в транзакции одной базы данных

На рис. 30 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).

Для конфигурирования такой программы надо:

1. Создать таблицы для накопительного счета, расчетного счета и плана оплаты в одной базе данных (смотрите раздел "Реализация проекта" в руководстве *Administration Guide: Implementation*)
2. Для физически удаленной базы данных: задать для сервера баз данных использование подходящего протокола связи, как описано в книге *Дополнение по установке и настройке*
3. Для физически удаленной базы данных - внести в каталог узел и базу данных, чтобы определить эту базу для сервера, как описано в книгах *Быстрый старт*.



4. Прекомпилировать прикладную программу, задав соединение типа 1; для этого надо задать CONNECT 1 (значение по умолчанию) в команде PRECOMPILE PROGRAM, как описано в книге *Application Development Guide*.

## Использование в транзакции нескольких баз данных

При использовании в одной транзакции нескольких баз данных требования к настройке и управления средой будут различными в зависимости от числа изменяемых в транзакции баз данных.

### Изменение одной базы 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).

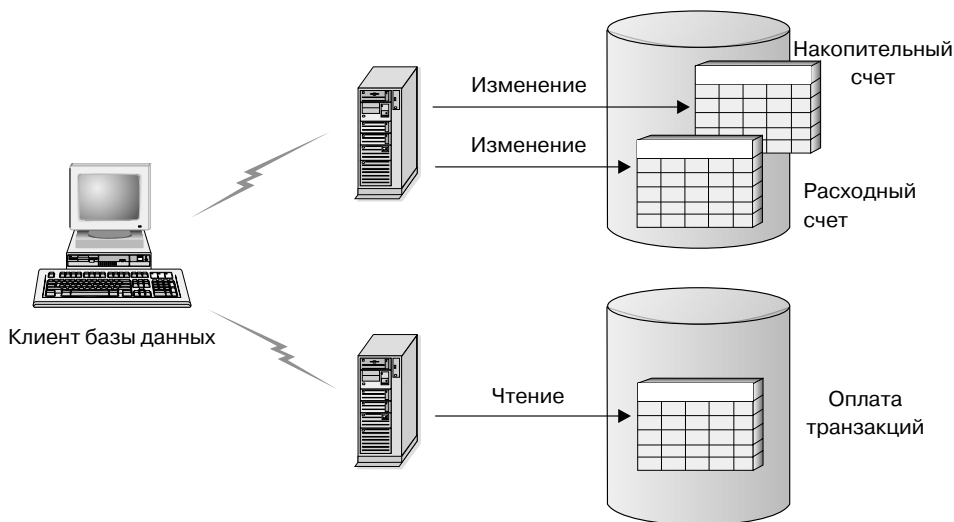


Рисунок 31. Использование в транзакции нескольких баз данных

На рис. 31 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. Этот пример подобен примеру на рис. 30 на стр. 138, но баз данных теперь несколько и таблицы размещены по-другому.

Чтобы сконфигурировать прикладную программу денежных переводов для этой среды, нужно:

1. Создать необходимые таблицы в соответствующих базах данных (смотрите раздел "Реализация вашего проекта" руководства *Administration Guide: Implementation*)

2. Для физически удаленной баз данных: задать для серверов баз данных использование подходящих протоколов связи, как описано в книге *Дополнение по установке и настройке*
3. Для физически удаленных баз данных: внести в каталог узлы и базы данных, чтобы определить эти базы для серверов баз данных, как описано в книгах *Быстрый старт*
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), как описано в книге *Application Development Guide*.

Если база данных находится на сервере баз данных хоста или AS/400, для связи с этим сервером требуется DB2 Connect. Информацию об установке DB2 Connect смотрите в одной из книг *Быстрый старт* по DB2 Connect. Информацию об использовании DB2 Connect смотрите в книге *DB2 Connect. Руководство пользователя*.

## **Изменение 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*.

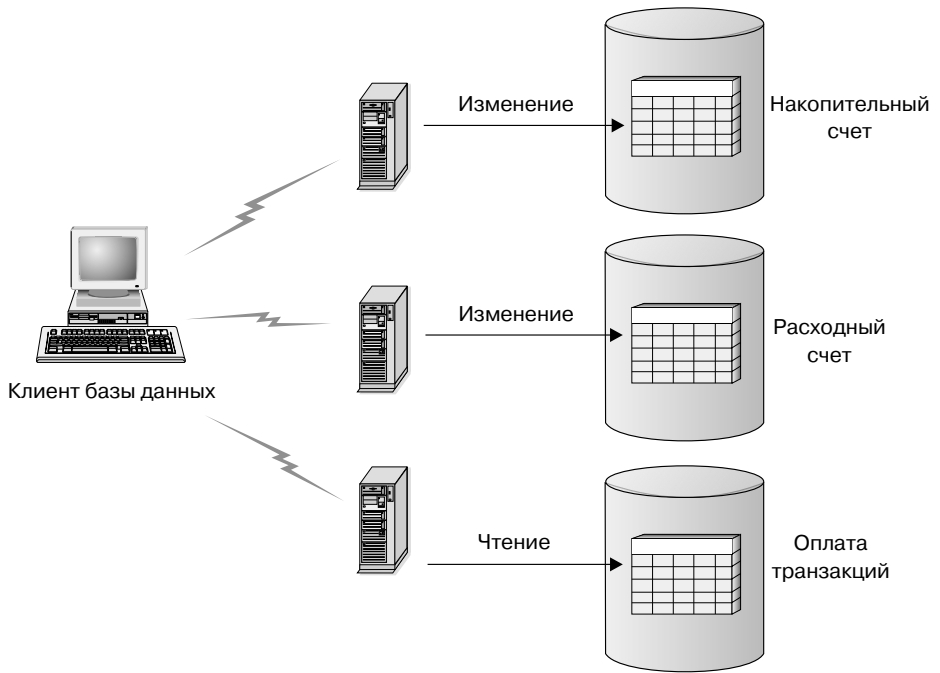


Рисунок 32. Изменение нескольких баз данных в одной транзакции

На рис. 32 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.

Чтобы сконфигурировать прикладную программу денежных переводов для этой среды, есть две возможности:

1. С менеджером транзакций (ТМ):
  - a. Создать необходимые таблицы в соответствующих базах данных (смотрите раздел "Реализация вашего проекта" книги *Administration Guide: Implementation*).
  - b. Для физически удаленной базы данных: задать для сервера баз данных использование подходящего протокола связи, как описано в книге *Дополнение по установке и настройке*.
  - c. Для физически удаленных баз данных: внести в каталог узлы и базы данных, чтобы определить эти базы для серверов баз данных, как описано в книгах *Быстрый старт*.
  - 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), как описано в книге *Application Development Guide*.

- e. Сконфигурировать менеджер транзакций DB2, как описано в разделе “Использование менеджера транзакций DB2”.

2. Без менеджера транзакций:

- a. Создать необходимые таблицы в соответствующих базах данных (смотрите раздел “Реализация вашего проекта” книги *Administration Guide: Implementation*).
- b. Для физически удаленной базы данных: задать для сервера баз данных использование подходящего протокола связи, как описано в книге *Дополнение по установке и настройке*.
- c. Для физически удаленных баз данных: внести в каталог узлы и базы данных, чтобы определить эти базы для серверов баз данных, как описано в книгах *Быстрый старт*.
- 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), как описано в книге *Application Development Guide*.

### **Использование менеджера транзакций DB2**

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.

Если используется менеджер транзакций, совместимый с XA (например, IBM TXSeries, BEA Tuxedo или Microsoft Transaction Server), инструкции по его интеграции в систему смотрите в разделе “Глава 10. Планирование использования менеджеров транзакций” на стр. 155.

Если для координации транзакций используется DB2 UDB для систем на основе UNIX, операционных систем Windows или OS/2, 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 OS/390 и DB2 for OS/390, конфигурирование будет несложным.

**DB2 UDB for OS/390 и DB2 for OS/390 при использовании связи 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 for OS/390) используется только TCP/IP.

- В транзакции используются только серверы баз данных DB2 UDB для систем на основе UNIX, операционных систем Windows, OS/2 или OS/390.
- 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 AS/400.
  - Двухфазное принятие координируется менеджером транзакций, совместимым с XA (например, IBM TXSeries CICS).  
Это верно как для связи SNA, так и для связи TCP/IP с серверами баз данных хоста или AS/400. Подробную информацию смотрите в разделе “Глава 10. Планирование использования менеджеров транзакций” на стр. 155. 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 перезапустите DB2.

База данных, которая будет использоваться в качестве базы данных менеджера транзакций, задается на клиенте базы данных параметром конфигурации *tm\_database*. Дополнительную информацию об этом параметре конфигурации смотрите в разделе “Конфигурирование DB2” руководства *Administration Guide: Performance*. При задании этого параметра конфигурации учтите следующие факторы:

- Менеджер транзакций может быть:
  - Базой данных DB2 UDB для систем на основе UNIX, операционных систем Windows или OS/2
  - Базой данных DB2 for OS/390 Версии 5 или более новой.  
В качестве базы данных менеджера транзакций рекомендуется использовать именно этот сервер баз данных. Системы OS/390 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.

- Для каталогизации и управления доступом к базам данных используются службы каталогов DCE.

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 MVS Версии 3 или Версии 4, DB2 for AS/400 или DB2 for VM&VSE
- Выполняются обращения к DB2 for OS/390 с использованием SNA
- Для доступа к серверам баз данных хоста или AS/400 используется менеджер точек синхронизации DB2 Connect

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*. Дополнительную информацию об этом параметре конфигурации смотрите в разделе "Конфигурирование DB2" руководства *Administration Guide: Performance*. При задании этого параметра конфигурации учтите следующие факторы:

- Базой данных менеджера транзакций может быть базой данных DB2 UDB для систем на основе UNIX, операционных систем Windows или OS/2.
- 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.
- Для каталогизации и управления доступом к базам данных используются службы каталогов DCE.

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 особенности конфигурирования

При конфигурировании среды надо рассмотреть следующие параметры конфигурации. Дополнительную информацию о задании этих параметров смотрите в руководстве *DB2 Connect. Руководство пользователя*.

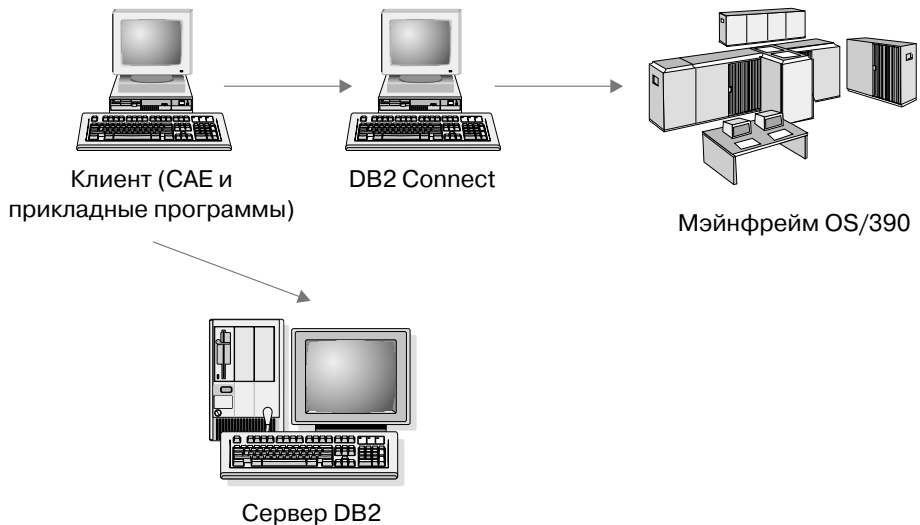


Рисунок 33. Особенности конфигурирования

### 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. Дополнительную информацию о неоднозначных транзакциях смотрите в разделе “Восстановление после ошибок двухфазного принятия” на стр. 151.

- *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 потребуются 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.

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

### **Прикладные программы хоста или AS/400, обращающиеся к серверу DB2 Universal Database в локальной сети при многоузловом изменении**

DB2 Universal Database не поддерживает многоузловое изменение с клиентов баз данных хоста или AS/400 с использованием связи TCP/IP. В этой ситуации поддерживается только связь SNA (Systems Network Architecture). Для многоузлового изменения требуется менеджер точек синхронизации DB2. DB2 Connect при этом не используется.



Сервер баз данных, к которому обращаются с клиента баз данных хоста или AS/400, не обязан быть локальным для рабочей станции, на которой установлен менеджер точек синхронизации DB2. Клиент баз данных хоста или AS/400 может соединиться 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 AS/400.

Шаги конфигурирования:

- На рабочей станции, к которой будут напрямую обращаться прикладные программы хоста или AS/400:
  1. Для поддержки многоузлового изменения с клиентов баз данных хоста или AS/400 установите DB2 Universal Database Enterprise Edition или Enterprise - Extended 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. Проверьте, что в значении реестра DB2COMM указано APPC.
  5. 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 *spt\_name*. Для других поддерживаемых стеков SNA требуется ручная настройка. Подробную информацию смотрите в руководстве *Дополнение по установке и настройке*.
  6. Определите значение, которое нужно задать для параметра конфигурации менеджера баз данных *spt\_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.
  7. При необходимости измените значение параметра *spt\_name* на сервере DB2 Universal Database, используя команду UPDATE DATABASE MANAGER CONFIGURATION.
  8. Сконфигурируйте связь, чтобы эта рабочая станция DB2 могла в случае необходимости соединиться с удаленными серверами DB2 UDB.
  9. Сконфигурируйте связь, чтобы удаленные серверы DB2 UDB могли соединиться с этим сервером DB2.
  10. Чтобы в первый раз запустить SPM, остановите и снова запустите менеджер баз данных на сервере DB2 Universal Database.

Необходимо, чтобы с этой рабочей станции DB2 UDB можно было соединиться с удаленными серверами DB2 UDB.

- На каждом удаленном сервере DB2 UDB, к которому будет обращаться клиент базы данных хоста или AS/400:
  1. Сконфигурируйте связь, чтобы рабочая станция удаленного сервера DB2 UDB с менеджером точек синхронизации DB2 могла соединиться с этим сервером DB2 UDB.
  2. Остановите и снова запустите менеджер баз данных.

---

## Процесс двухфазного принятия

На рис. 34 на стр. 149 показаны шаги многоузлового изменения. Понимание того, как выполняется управление транзакциями, поможет вам исправлять ошибки, возникшие в процессе двухфазного принятия.

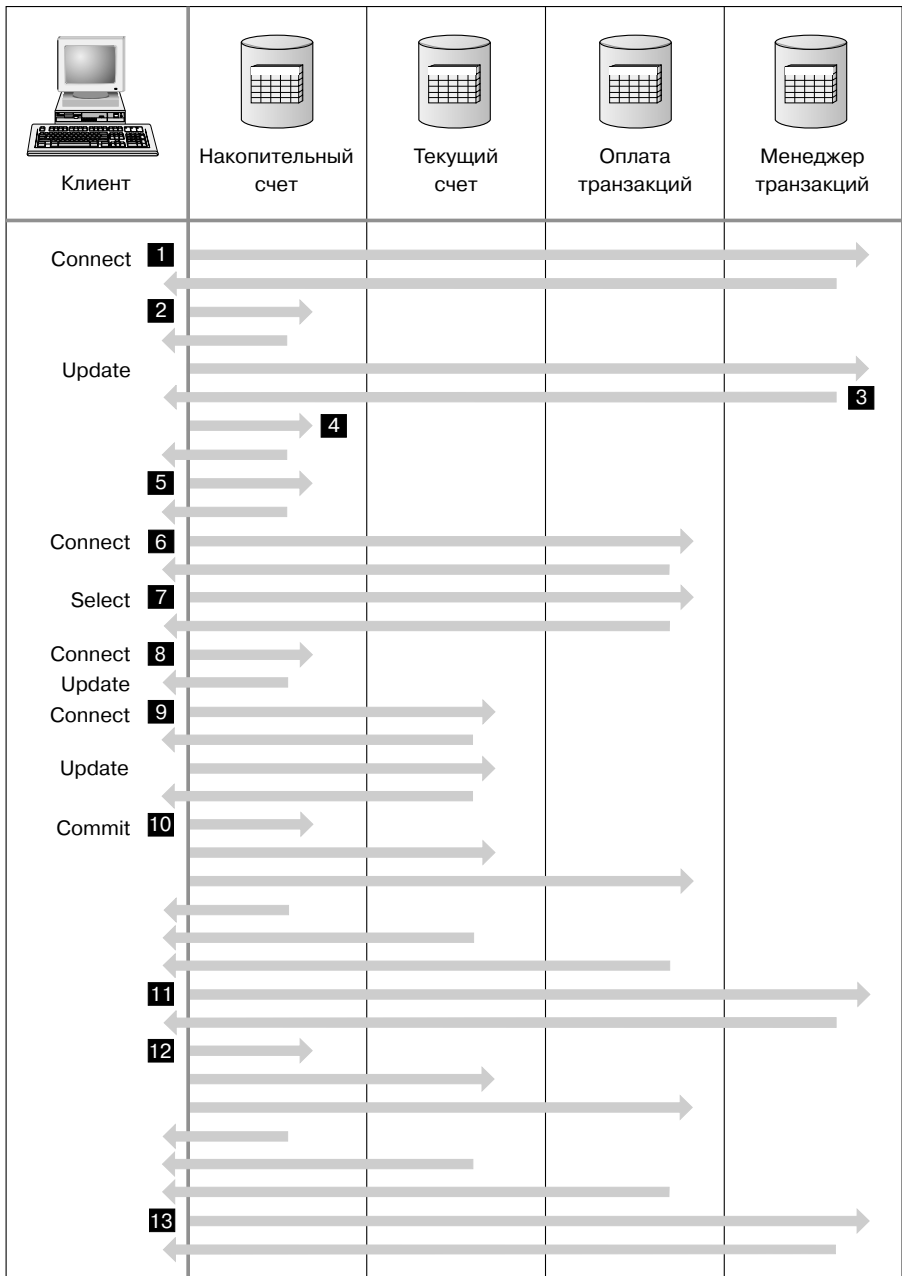


Рисунок 34. Изменение нескольких баз данных

- 0** Прикладная программа подготавливается для двухфазного принятия. Для этого могут использоваться опции прекомпиляции (подробную информацию смотрите в руководстве *Application Development Guide*).

Для этого также может использоваться конфигурация CLI (интерфейс уровня вызовов) DB2 (подробную информацию смотрите в руководстве *CLI Guide and Reference*).

- 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. (Описание SQLCA смотрите в руководстве *Application Development Guide* и справочнике *SQL Reference*.)
- 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 записывает 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 записывает 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 записывает 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 записывает запись "COMMITTED" в свой файл журнала, указывающую, что единица работы завершена, и отвечает клиенту, сообщая ему, что операция завершена.

---

## Восстановление после ошибок двухфазного принятия

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, описанный в разделе "Процесс двухфазного принятия" на стр. 148 (точки **10**, **11** и **12**). 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 записывает 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, к которой возникла ошибка. Однако прикладной программе в SQLCA будет передана информация о том, что принятие было успешным. DB2 обеспечит принятие непринятых транзакций в 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* (смотрите раздел "Конфигурирование DB2" руководства *Administration Guide: Performance*). 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". Дополнительную информацию о ручном восстановлении неоднозначных транзакций смотрите в разделе "Принятие эвристических решений" на стр. 167.

### **Ресинхронизация неоднозначных транзакций, если AUTORESTART=OFF**

Информацию об особенностях конфигурирования в среде DB2 Universal Database с двухфазным принятием смотрите в разделе "Другие особенности конфигурирования" на стр. 145.

В частности, 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.





---

## Глава 10. Планирование использования 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 нескольких баз данных” на стр. 140.

Следующие темы помогут вам освоить использование менеджера баз данных с таким XA-совместимыми менеджерами транзакций, как IBM TXSeries CICS, IBM TXSeries Encina, BEA Tuxedo и Microsoft Transaction Server:

- “Модель распределенной обработки транзакций X/Open” на стр. 156
- “Задание базы данных в качестве менеджера ресурсов” на стр. 160
- “Использование строк ха\_ореп и ха\_close” на стр. 160
- “Новый формат строки ха\_ореп для DB2 Версии 7” на стр. 160
- “Значения параметров TRM и TR\_MON\_NAME” на стр. 162
- “Формат строки ха\_ореп для прежних версий DB2” на стр. 166
- “Изменения на серверах баз данных хоста или AS/400” на стр. 166
- “Особенности соединения с базой данных” на стр. 166
- “Принятие эвристических решений” на стр. 167
- “Особенности защиты” на стр. 170
- “Особенности конфигурирования” на стр. 171
- “Поддерживаемая функция XA” на стр. 172
- “Диагностика ошибок интерфейса XA” на стр. 174
- “Конфигурирование менеджеров транзакций XA для использования DB2 UDB” на стр. 175
- “Конфигурирование сервера Microsoft Transaction Server” на стр. 180.

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/library/>

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”
- “Менеджер транзакций” на стр. 158
- “Менеджеры ресурсов” на стр. 159.

Эта модель и отношения между ее компонентами показаны на рис. 35.

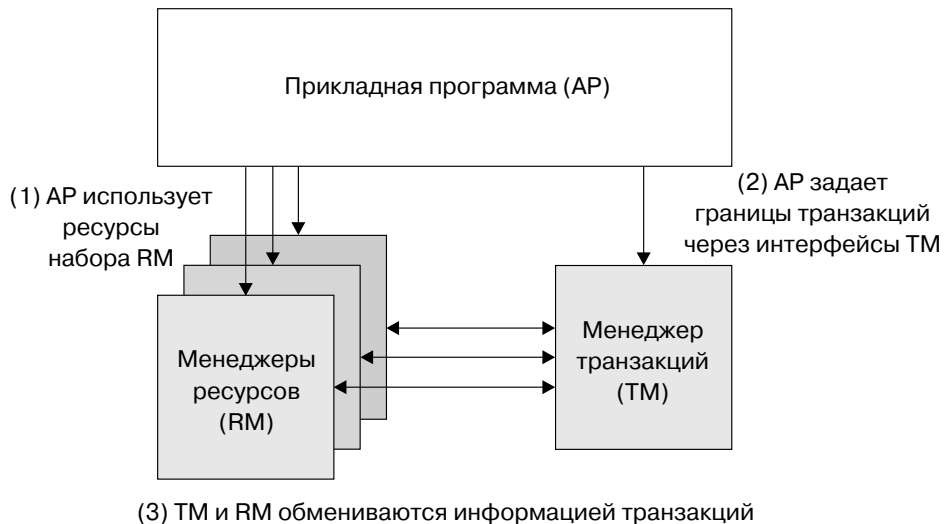


Рисунок 35. Модель распределенной обработки транзакций X/Open (DTP)

### Прикладная программа

Прикладной программы 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. Подробности приводятся в разделе "Особенности защиты" на стр. 170.

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 ТМ. 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 ТМ. 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 ТМ, 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. Смотрите раздел “Изменение нескольких баз данных” на стр. 140.)

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

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 менеджера 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 только чтение, менеджер транзакций вместо двухфазного принятия может использовать однофазное.

## Менеджеры ресурсов

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-совместимыми менеджерами 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. Дополнительную информацию о функциях XA, поддерживаемых менеджером баз данных, смотрите в разделе “Поддерживаемая функция XA” на стр. 172.

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. Информацию о вопросах программирования интерфейса X/Open XA смотрите в книге *Application Development Guide*.

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 конкретным менеджером транзакций смотрите в документации соответствующего менеджера транзакций. Информация об интеграции наиболее распространенных мониторов транзакций приводится в разделе “Конфигурирование менеджеров транзакций XA для использования DB2 UDB” на стр. 175.

---

## **Задание базы данных в качестве менеджера ресурсов**

Каждая база данных определяется для менеджера транзакций как отдельный менеджер ресурсов; база данных должна идентифицироваться строкой `xa_open`. Описание формата строки DB2 `xa_open` смотрите в разделе “Использование строк `xa_open` и `xa_close`”.

### **Использование строк `xa_open` и `xa_close`**

У строки менеджера баз данных `xa_open` есть два принятых формата. Один формат появился в DB2 Версии 7. Второй формат используется прежними версиями DB2 и оставлен в новой версии для совместимости. Новые реализации баз данных должны использовать новый формат, а более старые следует перенастроить на новый формат по возможности. Будущие версии DB2, возможно, не будут поддерживать старый формат строки `xa_open`. Информацию об исходном формате строки `xa_open` смотрите в разделе “Формат строки `xa_open` для прежних версий DB2” на стр. 166.

При задании базы данных в качестве менеджера ресурсов строка `xa_close` необязательна. Если ее задать, менеджер базы данных будет игнорировать ее.

### **Новый формат строки `xa_open` для DB2 Версии 7**

В DB2 Версии 7 используется следующий новый формат строки `xa_open` 7:

```
id1_параметра = <значение параметра>, id2_параметра = <значение параметра>, ...
```

Порядок этих параметров несуществен. Допустимые значения для *id\_параметра* описаны в следующей таблице.

Таблица 22. Допустимые значения для *parm\_id*

Имя параметра	Значение	Обязательность	Регистро-зависимость	Значение по умолчанию
DB	Алиас базы данных	Да	Нет	Нет
Алиас базы данных используется программой для обращения к базе данных.				
UID	ID пользователя	Нет	Да	Нет
ID пользователя, имеющего полномочия соединиться с этой базой данных. Обязательный, если задается пароль.				
PWD	Пароль	Нет	Да	Нет
Пароль, соответствующий данному ID пользователя. Обязателен, если задается ID пользователя.				
TRM	Имя монитора транзакций	Нет	Нет	Нет
Имя используемого монитора транзакций. Поддерживаемые значения смотрите в разделе “Значения параметров TRM и TP_MON_NAME” на стр. 162. Этот параметр можно задать, чтобы несколько мониторов транзакций могли использовать один экземпляр DB2. Задаваемое значение заменяет значение, заданное параметром конфигурации менеджера баз данных <i>tp_mon_name</i> .				
AXLIB	Библиотека, содержащая функции монитора транзакций <b>ax_reg</b> и <b>ax_unreg</b> .	Нет	Да	Нет
Это значение DB2 использует для получения адресов требуемых функций <b>ax_reg</b> и <b>ax_unreg</b> . Его можно использовать для переопределения значений, основанных на значении параметра TRM, его могут использовать также мониторы транзакций, которых нет в списке TRM.				
CHAIN_END	Флаг цепочки <i>xa_end</i> . Допустимые значения - T, F или нет значения.	Нет	Нет	F
Цепочка XA_END - это оптимизация, которую DB2 может использовать для сокращения сетевых потоков. Если среда монитора транзакций гарантирует, что <b>xa_prepare</b> будет вызван в том же потоке или процессе немедленно после вызова <b>xa_end</b> , и если CHAIN_END включен, флаг <i>xa_end</i> будет связан с вызовом <b>xa_prepare</b> , 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 TRM.				

Таблица 22. Допустимые значения для *parm\_id* (продолжение)

Имя параметра	Значение	Обязательность	Регистро-зависимость	Значение по умолчанию
SUSPEND_CURSOR	<p>Задает, должны ли храниться указатели, если поток транзакции управления приостанавливается. Допустимые значения - T, F или нет значения.</p> <p>Мониторы транзакций, которые приостановили выполнение ветви транзакции, могут повторно использовать приостановленный поток или процесс для других транзакций. В таких ситуациях указатели надо закрыть, чтобы их не унаследовала новая транзакция. Когда приостановленная транзакция возобновляется, программа должна получить эти указатели снова. Если SUSPEND_CURSOR включен, никакие открытые указатели не закрываются, но при этом поток или процесс не может повторно использоваться для других транзакций. Для приостановленной транзакции разрешается только возобновление. 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.</p>	Нет	Нет	F
HOLD_CURSOR	<p>Задает, удерживаются ли указатели после принятия транзакции. Допустимые значения - T, F или нет значения.</p> <p>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 HOLD_CURSOR включен, указатели удерживаются после принятия транзакций. 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.</p>	Нет	Нет	F

### Значения параметров TPM и TP\_MON\_NAME

Параметр TPM строки *ha\_orep* 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 23. 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

Таблица 23. Допустимые значения для параметров ТРМ и *tp\_mon\_name* (продолжение)

Значение ТРМ	Монитор транзакций	Внутренние параметры
MTS	Microsoft Transaction Server	Конфигурировать DB2 для MTS необязательно. MTS определяется автоматически драйвером DB2 ODBC.
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.

## 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 ХаOpenString 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 ха\_орен:

```
db=алиас_базы_данных,uid=id_пользователя,pwd=пароль
```

- Для каждой базы данных, определенной для XA TM, задать строку ха\_орен:

```
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 ХА ТМ, задать строку  
ха\_ореп:  
db=алиас\_базы\_данных,uid=id\_пользователя,pwd=пароль,axlib=myaxlib,chain\_end=T
- Для каждой базы данных, определенной для ХА ТМ, задать строку  
ха\_ореп:  
db=алиас\_базы\_данных,uid=id\_пользователя,pwd=пароль,axlib=myaxlib,chain\_end

## Формат строки ха\_ореп для прежних версий DB2

Ниже описывается формат строки ха\_ореп, который использовался в прежних версиях DB2. Этот формат по-прежнему поддерживается из соображений совместимости. По возможности программы следует перенастроить на новый формат (смотрите раздел “Новый формат строки ха\_ореп для DB2 Версии 7” на стр. 160).

Каждая база данных определяется для менеджера транзакций как отдельный менеджер ресурсов и должна идентифицироваться строкой ха\_ореп:

```
"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.

При задании базы данных в качестве менеджера ресурсов строка ха\_close необязательна. Если ее задать, менеджер базы данных будет игнорировать ее.

## Изменения на серверах баз данных хоста или AS/400

Возможность изменения для серверов баз данных хоста или AS/400 зависит от архитектуры менеджера транзакций ХА. Для поддержки последовательностей принятий от различных процессов должен быть включен концентратор DB2 Connect. Чтобы включить концентратор DB2 Connect EE, задайте для параметра конфигурации менеджера баз данных *max\_logicagents* значение больше значения параметра *maxagents*. Имейте в виду, что для поддержки последовательностей принятий ХА от различных процессов концентратору DB2 Connect EE требуется клиент DB2 Версии 7.1. Информацию об операторах SQL, допустимых в этой среде, смотрите в книге *Application Development Guide*. Информацию о концентраторе смотрите в книге *DB2 Connect. Руководство пользователя*.

Если вы будете вносить изменения на серверах баз данных хоста или AS/400, понадобится DB2 Connect с сконфигурированным менеджером точек синхронизации (SPM) DB2. Указания смотрите в одной из книг *Быстрый старт*.

## Особенности соединения с базой данных

В этом разделе обсуждаются следующие темы:

- “Оператор RELEASE” на стр. 167

- “Доступ к транзакциям в многораздельных базах данных”

### **Оператор RELEASE**

Если для освобождения соединения с базой данных используется оператор RELEASE, для восстановления соединения следует использовать оператор CONNECT, а не SET CONNECTION.

### **Доступ к транзакциям в многораздельных базах 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.

## **Принятие эвристических решений**

XA-совместимый менеджер транзакций (монитор обработки транзакций) использует процесс двухфазного принятия, сходный с процессом, описанным в разделе “Процесс двухфазного принятия” на стр. 148, который использует менеджером 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 обсуждались для менеджера транзакций DB2 (смотрите раздел “Восстановление после ошибок двухфазного принятия” на стр. 151). 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. Для получения информации о неоднозначных транзакциях из DB2 UDB в системах на основе UNIX, в системах Windows или OS/2 соединитесь с базой данных и введите команду LIST INDOUBT TRANSACTIONS WITH PROMPTING или вызовите эквивалентный API. Информацию об этой команде и соответствующих ей API управления смотрите в книгах *Command Reference* или *Administrative API Reference*.

Информацию о неоднозначности транзакций, относящуюся к серверам баз данных хоста или AS/400 можно получить двумя способами:

- Информацию о неоднозначности можно получить непосредственно с сервера хоста или AS/400.

Чтобы получить информацию о неоднозначности непосредственно от DB2 for OS/390, введите команду DISPLAY THREAD TYPE(INDOUBT). Для принятия эвристического решения используйте команду RECOVER. Чтобы получить информацию о неоднозначности непосредственно от DB2 for OS/400, введите команду **wrkcmdfn**.

- Информацию о неоднозначности можно получить от сервера DB2 Connect, использовавшегося для доступа к серверу баз данных хоста или AS/400.

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 *spt\_name*. Затем введите команду LIST DRDA INDOUBT TRANSACTIONS WITH PROMPTING для вывода неоднозначных транзакций и принятия эвристических решений.

Пользуйтесь этими командами (или соответствующими API) с *чрезвычайной предосторожностью* 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 течении 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.

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, недоступном 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.

Не выполняйте эвристическую функцию `forget` ("забыть"), если только эвристическое принятие или откат транзакции не приведет к переполнению журнала, которое указывается в выводе команды `INDOUBT TRANSACTIONS`.

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.

## Особенности защиты

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 `ha_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 в 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, зависящих от данной таблицы или производной таблицы.
2. Определите имена программ сервера (например, программ CICS), соответствующие этим пакетам по соглашению об именах, используемому 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.

## Особенности конфигурирования

При настройке среды монитора транзакций следует рассмотреть следующие параметры конфигурации:

- *tr\_mon\_name*

Этот параметр конфигурации идентифицирует имя используемого продукта монитора транзакций (например, "CICS" или "ENCINA").

- *trname*

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. Дополнительную информацию смотрите в руководствах *Быстрый старт*.

- *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 числа 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. Дополнительную информацию о неоднозначных транзакциях смотрите в разделе "Восстановление после ошибок двухфазного принятия" на стр. 151.

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 потребуются 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.

## Поддерживаемая функция XA

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. Дополнительную информацию об этих двух методах смотрите в разделе “Менеджеры ресурсов” на стр. 159.

- Передача транзакций

DB2 Universal Database не поддерживает передачу транзакций от одного управляющего потока другому.

Дополнительную информацию об использовании строк *xa\_open* и *xa\_close* смотрите в разделе “Использование строк *xa\_open* и *xa\_close*” на стр. 160.

## 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
<b>name</b>	Имя продукта менеджера баз данных. Например, DB2 for AIX.
<b>flags</b>	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.
<b>version</b>	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 менеджер 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 и OS/2:** 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 двух методов нужно связать свою программу с библиотекой *libdb2* (в системе на основе UNIX) или *db2api.lib* (в OS/2).

**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 ;
}
```

## Диагностика ошибок 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. Дополнительную информацию о журнале First Failure Service Log смотрите в руководстве *Troubleshooting Guide*.

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.

Вот пример записи в журнале ошибок для ошибки строки `xa_open` (строка `xa_open` пропущена), сгенерированной в AIX:

```
Tue Apr  4 15:59:08 1995
toop pid(83378) process (xatest) XA DTP Support      sqlxa_open Probe:101
DIA4701E Database "" could not be opened for distributed transaction
processing.
String Title : XA Interface SQLCA pid(83378)
SQLCODE = -998 REASON CODE: 4 SUBCODE: 1
Dump File : /u/toop/diagnostics/83378.dmp Data : SQLCA
```

---

## Конфигурирование менеджеров транзакций XA для использования DB2 UDB

Ниже описано, как конфигурировать отдельные продукты для использования DB2 в качестве менеджера ресурсов. Можно использовать любой из следующих продуктов:

- “Конфигурирование IBM TXSeries CICS”
- “Конфигурирование IBM TXSeries Encina” на стр. 176
- “Конфигурирование BEA Tuxedo” на стр. 178
- “Конфигурирование сервера Microsoft Transaction Server” на стр. 180.

### 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/](http://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/).

Серверы баз данных хоста и AS/400 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 или DB2 for MVS, DB2 for OS/390, DB2 for AS/400 или DB2 for VSE&VM при обращении через DB2 Connect. Документацию по TXSeries можно просмотреть на сайте [http://www.transarc.com/Library/documentation/websphere/WAS-EE/en\\_US/html/](http://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/).

Серверы баз данных хоста и AS/400 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 клиента (ССА) или процессора командной строки DB2 (CLP). Например:  

```
DB2 CATALOG DATABASE inventdb AS inventdb AT NODE host1 AUTH SERVER  
DB2 CATALOG TCP/IP 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 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 Eiconsole 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 менеджер ресурсов"). Для простоты его можно задать таким же, как у базы данных.

Строка `ha_orep` содержит информацию, которая требуется для установления соединения с базой данных. Содержание этой строки определяется конкретным менеджером ресурсов. Строка `ha_orep` для DB2 UDB содержит алиас открываемой базы данных и, необязательно, ID пользователя и пароля, 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. Информацию о строке `ha_orep` для DB2 смотрите в разделе "Задание базы данных в качестве менеджера ресурсов" на стр. 160.

Строка `ha_close` в DB2 не используется.

Соглашение о потоке управления определяет, может ли агент прикладной программы обрабатывать несколько транзакций за раз. DB2 UDB поддерживает значение по умолчанию для `TMXA_SERIALIZE_ALL_OPERATIONS`, то есть поток может вновь использоваться только после завершения транзакции.

При обращении к DB2 for OS/390, DB2 for MVS, DB2 for AS/400 или DB2 for VSE&VM следует использовать менеджер точек синхронизации DB2. Инструкции по конфигурированию смотрите в руководстве *DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings*.

### **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)
```

Для DB2 (серверы баз данных DB2 Universal Database, хоста или AS/400) надо использовать значение по умолчанию `MON_EXCLUSIVE`. Это гарантирует, что:

- На время существования транзакции процесс прикладной программы блокируется.
- Программа работает как однопоточная.

**Примечание:** При использовании ODBC или интерфейса уровня вызовов DB2 надо отключить поддержку многопоточности. Это можно сделать, задав значение параметра конфигурации интерфейса уровня вызовов DB2 `DISABLEMULTITHREAD = 1` (отключение

поддержки многопоточности). По умолчанию для DB2 Universal Database используется `DISABLEMULTITHREAD = 0` (включение поддержки многопоточности). Дополнительную информацию смотрите в руководстве *CLI Guide and Reference*.

2. При помощи API регистрации менеджера ресурсов Encina задайте переключатель XA и логическое имя менеджер ресурсов, используемые Encina при обращении к менеджер 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 OS/2, 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 менеджер ресурсов совпадает с именем базы данных.

Третий параметр возвращает внутренний идентификатор или хэндл, который используется TM для ссылок на это соединение.

**Примечание:** При использовании Encina для обработки транзакций с DB2 через интерфейс XA менеджера транзакций учтите, что вложенные транзакции в настоящее время интерфейсом XA DB2 не поддерживаются. Избегайте использовать такие транзакции, если это возможно. Если же избежать их нельзя, постарайтесь обеспечить, чтобы обработка SQL выполнялась только на одном члене семейства транзакций Encina.

## Конфигурирование BEA Tuxedo

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 соединиться 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_UBD
```

где {TUXDIR} - каталог установки Tuxedo.

- Для Windows NT:

```
%TUXDIR%\bin\buildtms -r UDB_XA -o %TUXDIR%\bin\TMS_UBD
```

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 OS/390, DB2 for OS/400 или DB2 for VM&VSE, требуется концентратор XA для DB2 Connect. Подробности конфигурирования и ограничения смотрите в книге *DB2 Connect. Руководство пользователя*.
9. Запустите Tuxedo:

```
tmbboot -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.

## Конфигурирование сервера Microsoft Transaction Server

DB2 UDB, начиная с Версии 5.2, может быть полностью интегрирована с сервером транзакций MTS (Microsoft Transaction Server) Версии 2.0. Программы, запускаемые под MTS в 32-битных операционных системах Windows, могут использовать MTS для координирования двухфазного принятия с несколькими серверами баз данных DB2 UDB, хоста и AS/400, а также с другими совместимыми с MTS менеджерами ресурсов.

**Примечание:** На Web-сайте IBM можно получить дополнительную техническую информацию, которая поможет вам установить и сконфигурировать поддержку DB2 MTS. Введите адрес <http://www.ibm.com/software/data/db2/library/>, и поищите DB2 Universal Database "Technote" с ключевым словом "MTS".

## Включение поддержки MTS в DB2

Поддержка Microsoft Transaction Server включается автоматически. Вы можете установить для параметра конфигурации *tp\_mon\_name* менеджера баз данных значение MTS, но это необязательно, и заданное значение будет проигнорировано.

## Предварительные требования для программного обеспечения MTS

Для поддержки MTS требуется DB2 Client Application Enabler (CAE) Версии 5.2 или более новой, а сервер транзакций MTS должен быть версии 2.0 с последними исправлениями (выпуск 0772 или новее).

В 32-битных операционных системах Windows при установке драйвера DB2 ODBC в реестр автоматически будет добавлено новое ключевое слово:

```
HKEY_LOCAL_MACHINE\software\ODBC\odbcinit.ini\IBM DB2 ODBC Driver:  
Keyword Value Name: CTimeout  
Data Type: REG_SZ  
Value: 60
```

## установка и конфигурирование

Ниже приводится сводная информация по установке и настройке конфигурации MTS. Для использования поддержки DB2 для MTS вы должны:

1. Установить MTS и клиент DB2 на том компьютере, где запускается прикладная программа MTS.
2. Если серверы баз данных хоста или AS/400 участвуют в многоузловом изменении:
  - a. Установить на локальном или удаленном компьютере DB2 Connect Enterprise Edition. DB2 Connect EE позволяет участвовать серверам баз данных хоста или AS/400 в транзакции многоузлового изменения.
  - b. Включить для вашего сервера DB2 Connect EE поддержку многоузлового изменения. Информацию о включении поддержки многоузловых изменений DB2 Connect смотрите в руководстве DB2 Connect Enterprise Edition Quick Beginnings для вашей платформы.

При запуске программ DB2 CLI/ODBC нельзя изменять значения по умолчанию, заданные в файле *db2cli.ini*, следующих ключевых слов конфигурации:

- Ключевое слово CONNECTTYPE (значение по умолчанию 1)
- Ключевое слово MULTICONNECT (значение по умолчанию 1)
- Ключевое слово DISABLEMULTITHREAD (значение по умолчанию 0)
- Ключевое слов CONNECTIONPOOLING (значение по умолчанию 0)
- Ключевое слов KEEPCONNECTION (значение по умолчанию 0)

Программы DB2 CLI, написанные для поддержки MTS, не должны изменять значения атрибутов, соответствующие этим ключевым словом. Кроме того, такие программы не должны изменять значения по умолчанию следующих атрибутов:

- SQL\_ATTR\_CONNECT\_TYPE (по умолчанию SQL\_CONCURRENT\_TRANS)
- SQL\_ATTR\_CONNECTON\_POOLING (по умолчанию SQL\_CP\_OFF)

### Проверка правильности установки

1. Сконфигурируйте для вашего клиента DB2 и DB2 Connect EE доступ к серверу баз данных DB2 UDB, хоста или AS/400.
2. Проверьте соединение из компьютера DB2 CAE к серверам баз данных DB2 UDB.
3. Проверьте соединение из компьютера DB2 Connect к серверу баз данных хоста или AS/400 с DB2 CLP и введите несколько запросов.
4. Проверьте соединение из компьютера DB2 CAE через шлюз DB2 Connect к серверу баз данных хоста или AS/400 и введите несколько запросов.

### Поддерживаемые серверы баз данных DB2

Многоузловое изменение с использованием транзакций, координируемых MTS, поддерживается следующими серверами:

- DB2 Universal Database Enterprise Edition Версии 6 и новее
- DB2 Universal Database Enterprise - Extended Edition Версии 6 и новее
- DB2 for OS/390
- DB2 for MVS
- DB2 for AS/400
- DB2 for VM & VSE

### Срок ожидания транзакции MTS и поведение соединения DB2

Значение срока ожидания транзакции можно установить через MTS Explorer. Дополнительную информацию смотрите в электронном руководстве *MTS Administrator Guide*.

Если транзакция продолжается дольше срока ожидания (значение по умолчанию - 60 секунд), MTS асинхронно выдаст команду прекращения работы всех вовлеченных менеджеров ресурсов, и вся транзакция будет прервана.

Для соединения с сервером DB2 прекращение транзакции преобразуется в требование отката DB2. Как и любое другое требование базы данных, это требование отката преобразуется при соединении в последовательную форму для гарантии целостности данных на сервере баз данных.

В результате:

- Если подключение незанято, откат будет выполнен немедленно.
- Если продолжается обработка оператора SQL, требование отката ждет завершения этого оператора.

## Поддержка пула соединений

Пулы соединений позволяют программам использовать соединение из пула, поэтому отпадает необходимость каждый раз переустанавливать соединение, когда оно используется. Как только соединение создано и помещено в пул, программа может использовать его, не выполняя полный процесс соединения. Когда программа отсоединяется от источника данных ODBC, соединение помещается в пул; оно будет предоставлено для нового соединения с совпадающими атрибутами.

Функция пулов соединений уже использовалась в менеджере драйверов ODBC 2.x. В самом последнем менеджере драйверов ODBC (версия 3.5), поставляемом с MTS, функция пулов соединений отличается некоторыми особенностями конфигурации и новым поведением соединений ODBC для объектов транзакций MTS COM (смотрите раздел “Множественное использование соединений ODBC между объектами COM, участвующих в одной транзакции” на стр. 184).

Менеджер драйверов ODBC версии 3.5 перед тем, как разрешить активацию пула соединений, требует, чтобы драйвер ODBC зарегистрировал в реестре новое ключевое слово. Это ключевое слово:

```
Key Name: SOFTWARE\ODBC\ODBCINST.INI\IBM DB2 ODBC DRIVER
Name: CPOutTimeout
Type: REG_SZ
Data: 60
```

Драйверы DB2 ODBC Версии 6 и более новых для 32-битных операционных систем Windows полностью поддерживают пулы соединений, и данное ключевое слово регистрируется. Клиенты Версии 5.2 должны установить пакет исправлений 3 (WR09024) или более новый.

Значение по умолчанию - 60 - означает, что соединение будет находиться в пуле в течение 60 секунд до его отключения.

В средах с высокой занятостью лучше увеличить значение CPOutTimeout срока ожидания для пула соединений (в определенных средах Microsoft иногда предлагает значение 10 минут), чтобы избежать роста числа физических соединений и рассоединений, так как при этом потребляется много системных ресурсов, включая системную память и ресурсы стеков связи.

Кроме того, чтобы то же самое соединение использовалось между объектами в той же транзакции на многопроцессорном компьютере, надо отключить поддержку нескольких пулов на один процессор. Для этого надо скопировать следующее значение реестра в файл под названием `odbcpool.reg`, сохранить его как простой текстовый файл и выполнить команду `odbcpool.reg`. Операционная система Windows импортирует это значение реестра.

REGEDIT4

```
[HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\ODBC Connection Pooling]
"NumberOfPools"="1"
```

Если не задать для этого ключевого слова значение 1, MTS может поместить соединения в разные пулы, что не даст возможности повторно использовать то же самое соединение.

### **Объединение в пул соединений MTS при помощи ADO 2.1 и более новых**

Если COM-объекты MTS используют для обращения к базе данных ADO, надо отключить объединение в пул ресурсов OLEDB, чтобы провайдер Microsoft OLEDB для ODBC (MSDASQL) не мешал объединять в пул соединения ODBC. Эта возможность при инициализации отключается в ADO 2.0, но включается в ADO 2.1. Чтобы отключить объединение в пул ресурсов OLEDB, надо скопировать следующее значение реестра в файл под названием `oledb.reg`, сохранить его как простой текстовый файл и выполнить команду **oledb.reg**. Операционная система Windows импортирует эти значения реестра.

REGEDIT4

```
[HKEY_CLASSES_ROOT\CLSID\{c8b522cb-5cf3-11ce-ade5-00aa0044773d}]
@="MSDASQL"
"OLEDB_SERVICES"=dword:ffffffff
```

### **Множественное использование соединений ODBC между объектами COM, участвующих в одной транзакции**

В соединениях ODBC в объектах MTS COM используется пул соединений, включаемый автоматически (независимо от участия объектов COM в транзакциях).

Если несколько объектов MTS COM участвуют в одной и той же транзакции, соединение между ними может использоваться многократно следующим образом.

Предположим, что есть два объекта COM - COM1 и COM2, которые соединяются с одним источником данных ODBC и участвуют в одной транзакции.

После того, как COM1 соединится и выполнит свою работу, он отсоединяется, и соединение помещается в пул. Однако это соединение будет зарезервировано для использования другими объектами COM той же транзакции. Оно станет доступно для других транзакций, только когда закончится текущая.

Когда в этой же транзакции вызывается COM2, предоставляется соединение из пула. MTS гарантирует, что данное соединение будет предоставлено только объектам COM, участвующим в той же транзакции.

С другой стороны, если объект COM1 не отключится явно, он займет это соединение до окончания транзакции. Когда в этой же транзакции будет вызван COM2, ему будет предоставлено отдельное соединение. С этого момента данная транзакция будет занимать два соединения вместо одного.

Такая функция многократного использования соединений для объектов COM, участвующих в одной транзакции, имеет следующие преимущества:

- На клиенте и на сервере используется меньше ресурсов. Необходимо только одно соединение.
- Устраняется возможность, что два соединения, участвующие в одной транзакции (обращающиеся к одному серверу баз данных и к одним и тем же данным), будут блокировать друг друга, так как серверы DB2 обрабатывают разные соединения из объектов MTS COM как отдельные транзакции.

### Настройка связи TCP/IP

Если в среде с высокой рабочей нагрузкой, где одновременно происходит слишком много физических соединений и рассоединений, используется маленькое значение CPTimeout, стек TCP/IP может столкнуться с ограничениями ресурсов.

Для преодоления этой проблемы используйте записи реестра TCP/IP. Они описаны в книге *Windows NT Resource Guide*, Том 1. Значения ключей реестра находятся в HKEY\_LOCAL\_MACHINE—> SYSTEM—> CurrentControlSet—> Services—> TCPIP—> Parameters.

Значения по умолчанию и предлагаемые параметры:

Имя	Значение по умолчанию	Предлагаемое значение
KeepAlive time	7200000 (2 часа)	То же
KeepAlive interval	1000 (1 секунда)	10000 (10 секунд)
TcpKeepCnt	120 (2 минуты)	240 (4 минуты)
TcpKeepTries	20 (20 попыток)	То же
TcpMaxConnectAttempts	3	6
TcpMaxConnectRetransmission	3	6
TcpMaxDataRetransmission	5	8
TcpMaxRetransmissionAttempts	7	10
Если значение реестра не задано, создайте его.		

### Тестирование DB2 программой примера MTS "BANK"

Программу примера "BANK", поставляемую с MTS, можно использовать для проверки настройки клиентских продуктов и MTS.

Выполните следующие шаги:

1. Измените файл `\Program Files\Common Files\ODBC\Data Sources\MTSSamples.dsn`, чтобы он выглядел так:

```
[ODBC]
DRIVER=IBM_DB2_ODBC_DRIVER
UID=ваш_id_пользователя
PWD=ваш_пароль
DSN=алиас_вашей_базы_данных
Description=MTS Samples
```

где:

- *ваш\_id\_пользователя* и *ваш\_пароль* - это ID пользователя и пароль, используемые для соединения с хостом.
  - *алиас\_вашей\_базы\_данных* - алиас базы данных, используемый для соединения с сервером баз данных.
2. Откройте Управление ODBC из Панели управления, выберите закладку **System DSN** и добавьте источник данных:
    - a. Выберите драйвер IBM ODBC и нажмите кнопку **Готово**.
    - b. Когда появится список алиасов базы данных, выберите алиас, заданный ранее.
    - c. Нажмите кнопку **ОК**.
  3. Воспользуйтесь DB2 CLP, чтобы соединиться с базой данных DB2 с указанным выше ID *ваш\_id\_пользователя*.
    - a. Свяжите файл `db2cli.lst`:

```
db2 bind @C:\sqllib\bnd\db2cli.lst blocking all grant public
```
    - b. Свяжите утилиты.  
Если сервер - сервер хоста DRDA, свяжите `ddcsmvs.lst`, `ddcs400.lst` или `ddcsvm.lst` в зависимости от хоста, с которым соединяетесь (OS/390, AS/400 или VSE&VM). Например:

```
db2 bind @C:\sqllib\bnd\ddcsmvs.lst blocking all grant public
```

  
Иначе свяжите файл `db2ubind.lst`:

```
db2 bind @C:\sqllib\bnd\db2ubind.lst blocking all grant public
```
    - c. Создайте для программы примера MTS таблицу и данные примера:

```
db2 create table account (accountno int, balance int)
db2 insert into account values(1, 1)
```
  4. Убедитесь, что на клиенте DB2 для параметра конфигурации менеджера баз данных `tr_top_name` задано значение MTS.
  5. Запустите программу "BANK". Нажмите кнопку **Account** и выберите опцию **Visual C++**, затем отправьте требование. Другие опции могут использовать SQL, специфичный для сервера SQL, и они могут не работать.



---

## Глава 11. Высокая доступность и восстановление при отказах. Введение

Успех электронной коммерции зависит от непрерывной доступности систем обработки транзакций, которые, в свою очередь, управляются системами управления базами данных, такими как DB2, которые должны быть доступны 24 часа в сутки и 7 дней в неделю (“24 x 7”).

---

### Высокая доступность

*Высокая доступность* (High availability, HA) - термин, применяемый для описания систем, работающих и доступных клиентам в любой момент времени. Для этого:

- Транзакции должны обрабатываться эффективно, то есть без значительного понижения производительности (или даже потери доступности) во время максимальной нагрузки. В среде многораздельных баз данных DB2 для эффективной обработки транзакций может использовать преимущества как внутрираздельного, так и межраздельного параллелизма. *Внутрираздельный параллелизм* можно использовать в среде SMP для одновременной обработки различных составляющих сложного оператора SQL. С другой стороны, *межраздельный параллелизм* в среде многораздельных баз данных, означает одновременную обработку запроса на всех участвующих узлах; каждый узел обрабатывает свой набор строк таблицы. Дополнительную информацию о параллелизме смотрите в разделе “Типы параллелизма” на стр. 37.
- Должна быть возможность быстро восстановить систему, если произошла аппаратная или программная ошибка или случилась авария. В DB2 используется непрерывная система контрольных точек и есть возможность параллельного восстановления, что обеспечивает предельно быстрое восстановления при отказах.

Способность системы быстро восстанавливаться зависит также от наличия надежной резервной копии и алгоритма ее восстановления. Дополнительную информацию об алгоритмах восстановления смотрите в книге *Справочное руководство по восстановлению данных и высокой доступности*.

- Программное обеспечение, обеспечивающее работу базы данных предприятия, должно быть пригодно для непрерывной работы и обработки транзакций. Для непрерывности работы менеджера баз данных надо, чтобы в случае отказа выполнение его функций взял на себя другой менеджер баз данных. Это называется восстановлением при отказах. Возможность *восстановления при отказах* позволяет при аппаратных отказах автоматически передавать рабочую нагрузку с одной системы на другую.

Для обеспечения восстановления при отказах можно хранить копию базы данных на другом компьютере, который будет постоянно повторять транзакции из файлов журналов. *Отправка журналов* - это процесс копирования всех файлов журналов на резервный компьютер с устройства архивации или через программу обработчика пользователя, выполняемую для исходной базы данных. При таком подходе исходная база данных восстанавливается на резервном компьютере либо с помощью утилиты восстановления DB2, либо с помощью функции отделения зеркальной копии. Для быстрой инициализации новой базы данных можно использовать новую возможность приостановки ввода-вывода (смотрите раздел “Высокая доступность с помощью оперативного отделения зеркальной копии и поддержки приостановленного ввода/вывода” на стр. 190). Вторичная база данных на резервном компьютере постоянно повторяет транзакции из файлов журналов. Если в первичной базе данных происходит отказ, все оставшиеся файлы журналов копируются на резервный компьютер. После повтора всех транзакций до конца журналов и прекращения операции все клиенты подсоединяются к вторичной базе данных на резервном компьютере.

Восстановление при отказах можно также обеспечить с помощью программ для конкретных платформ, которые можно добавить в систему. Например:

- HACMP/ES (High Availability Cluster Multi-Processing, Enhanced Scalability - мультипроцессорная кластерная обработка с высокой доступностью и улучшенной масштабируемостью) для AIX.

Подробную информацию о HACMP/ES смотрите в книге *Справочное руководство по восстановлению данных и высокой доступности*, или в документе под названием “IBM DB2 Universal Database Enterprise Edition для AIX и HACMP/ES”, который можно найти на сайте “DB2 UDB and DB2 Connect Online Support” (<http://www.ibm.com/software/data/pubs/papers/>).

- Microsoft Cluster Server для Windows NT или Windows 2000.

Подробную информацию о MSCS смотрите в книге *Справочное руководство по восстановлению данных и высокой доступности*.

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

Информацию о Sun Cluster 2.x смотрите в книге *Справочное руководство по восстановлению данных и высокой доступности*; информацию о Sun Cluster 3.0 смотрите в документе под названием “DB2 and High Availability on Sun Cluster 3.0” на сайте “DB2 UDB and DB2 Connect Online Support”

(<http://www.ibm.com/software/data/pubs/papers/>). Информацию о VERITAS Cluster Server смотрите в документе под названием “DB2 and High Availability on VERITAS Cluster Server”, который также можно найти на сайте “DB2 UDB and DB2 Connect Online Support”.

- Multi-Computer/ServiceGuard для Hewlett-Packard.

Подробную информацию о HP MC/ServiceGuard смотрите в документе под названием “IBM DB2 EE v.7.1 Implementation and Certification With

Hewlett-Packard's MC/ServiceGuard High Availability Software”, который можно найти на сайте (<http://www.ibm.com/software/data/pubs/papers/>).

Алгоритмы восстановления при отказах обычно основаны на использовании кластеров из систем. *Кластер* - это группа связанных систем, работающих совместно, как одна система. Каждый процессор рассматривается как узел в кластере. Кластеризация позволяет серверам подстраховывать друг друга: при отказе рабочую нагрузку сервера, на котором произошел отказ, принимает на себя другой сервер кластера.

Подмена IP-адреса - это возможность переносить IP-адрес сервера с одного компьютера на другой при отключении сервера; для пользователей эти два компьютера в разные моменты времени представляются одним сервером.

Программное обеспечение для восстановления при отказах для проверки связи между системами использует *мониторинг работоспособности* или *пакеты подтверждения активности*. Мониторинг работоспособности выполняется системными службами, которые обеспечивают постоянную связь между узлами кластера. Если пропадают сигналы работоспособности, начинается восстановление на резервную систему. Конечные пользователи обычно не замечают, что в работе системы был сбой.

Два наиболее часто используемых алгоритма восстановления при отказах - это *горячее резервирование* и *взаимная подмена*; в зависимости от производителя эти конфигурации могут называться по-другому.

#### **Горячее резервирование**

В этом режиме одна система используется для запуска экземпляра DB2, а вторая система “ожидает” или находится в режиме резервирования; она готова в случае сбоя операционной системы или аппаратного обеспечения, затрагивающего первую систему, принять на себя обслуживание экземпляра. Общая производительность системы не изменяется, так как пока в резервной системе нет потребности, она не несет нагрузки.

#### **Взаимная подмена**

В этом режиме каждая система является одновременно и резервной копией другой системы. Общая производительность системы может снизиться, так как для восстановления после отказа система должна выполнять дополнительную работу: собственную работу плюс работу, которая выполнялась отказавшей системой.

Алгоритмы восстановления при отказах могут применяться для восстановления экземпляра, раздела или нескольких логических узлов.

## Высокая доступность с помощью оперативного отделения зеркальной копии и поддержки приостановленного ввода/вывода

*Приостановленный ввод/вывод* обеспечивает постоянную доступность системы, полностью реализуя оперативное отделение зеркальной копии, то есть отделение зеркальной копии без закрытия базы данных. *Зеркальная копия* - это “мгновенная” копия базы данных, для ее получения можно поддерживать точную копию дисков с данными и отделить эту зеркальную копию, когда потребуется. *Зеркальное копирование дисков* - это процесс параллельной записи всех данных на два отдельных жестких диска, один из них будет зеркальной копией другого. *Отделение* зеркальной копии - это процесс снятия резервной копии с зеркальной.

Если вы не хотите снимать резервную копию с большой базы данных с помощью утилиты резервного копирования DB2, можно сделать копию с зеркальной копии с помощью приостановленного ввода/вывода и функции разделения зеркальной копии. Этот подход:

- Устраняет издержки от операции резервного копирования на основном компьютере
- Представляет собой быстрый способ клонирования систем
- Представляет собой быструю реализацию восстановления с помощью горячего резервирования. В этом случае нет предварительных операций для восстановления, и, в отличие от процедуры повтора транзакций, которая может оказаться слишком медленной или приводить к ошибкам, повторная инициализация происходит очень быстро.

Команда **db2inidb** инициализирует отделенную зеркальную копию, в результате ее можно использовать:

- Для создания клона базы данных  
Клон базы данных, предназначенный только для чтения, можно использовать, например, для создания отчетов.
- Как резервную базу данных
- Как резервную копию

Эту команду можно выполнить только для отделенной зеркальной копии, а для отделенной зеркальной копии надо выполнить **db2inidb**, прежде чем ее можно будет использовать (смотрите книгу *Справочное руководство по восстановлению данных и высокой доступности*).

В среде многораздельных баз данных надо выполнить команду **db2inidb** для всех разделов, прежде чем можно будет использовать отделенную копию какого-либо из разделов. Инструмент можно применять ко всем разделам одновременно.

## Создание клона базы данных

Клон базы данных можно использовать как автономную “резервную копию” основной (действующей) базы данных. Однако с клонированной базы данных нельзя сделать резервную копию, восстановить эту копию на исходную систему и повторить транзакции из файлов журналов, созданных на исходной системе.

Чтобы клонировать базу данных, выполните следующие действия:

1. Приостановите операции ввода/вывода в исходной базе данных:  
`db2 set write suspend for database`
2. Воспользуйтесь соответствующими командами уровня операционной системы, чтобы отделить зеркальную копию от исходной базы данных.
3. Возобновите операции ввода/вывода в исходной базе данных:  
`db2 set write resume for database`
4. Подключитесь к зеркальной копии базы данных с другого компьютера.
5. Запустите экземпляр базы данных:  
`db2start`
6. Инициализируйте зеркальную копию базы данных как клон первичной базы данных:  
`db2inidb алиас_базы_данных as snapshot`

**Примечание:** Эта команда произведет откат транзакций, которые выполнялись в момент отделения.

## Использование отделенной зеркальной копии в качестве резервной базы данных

Так как для зеркальной (резервной) базы данных непрерывно выполняется повтор транзакций с использованием журналов, новые журналы, которые создаются в исходной базе данных, постоянно считываются из исходной системы. Чтобы воспользоваться отделенной зеркальной копией как резервной базой данных, выполните следующие действия:

1. Приостановите операции ввода/вывода в исходной базе данных:  
`db2 set write suspend for database`
2. Воспользуйтесь соответствующими командами уровня операционной системы, чтобы отделить зеркальную копию от исходной базы данных.
3. Возобновите операции ввода/вывода в исходной базе данных:  
`db2 set write resume for database`
4. Присоедините зеркальную копию базы данных к другому экземпляру.
5. Переведите зеркальную копию базы данных в состояние отложенного повтора транзакций:  
`db2inidb алиас_базы_данных as standby`

Если у вас есть только табличные пространства, управляемые базой данных (DMS), можно сделать резервную копию всей базы данных, чтобы уменьшить издержки от снятия резервной копии с рабочей базы данных.

6. Настройте программу обработчика пользователя, чтобы она получала из исходной базы данных файлы журналов.
7. Повторите для базы данных транзакции до конца журналов.
8. Продолжайте получать файлы журналов и повторять транзакции до конца журналов, пока исходная база данных будет работать.

## **Использование отделенной зеркальной копии в качестве резервной копии**

Чтобы воспользоваться отделенной зеркальной копией как “резервной копией”, выполните следующие действия:

1. Приостановите операции ввода/вывода в исходной базе данных:  
`db2 set write suspend for database`
2. Воспользуйтесь соответствующими командами уровня операционной системы, чтобы отделить зеркальную копию от исходной базы данных.
3. Возобновите операции ввода/вывода в исходной базе данных:  
`db2 set write resume for database`
4. В исходной системе происходит отказ, и требуется восстановление из резервной копии.
5. Воспользуйтесь командами операционной системы, чтобы скопировать отделенные данные поверх исходной системы. Не копируйте отделенные журналы, поскольку для повтора транзакций надо иметь журналы исходной системы.
6. Запустите исходный экземпляр базы данных:  
`db2start`
7. Инициализируйте исходную базу данных:  
`db2inidb алиас_базы_данных as mirror`
8. Повторите для исходной базы данных транзакции до конца журналов.

---

## Часть 4. Приложения





---

## Приложение А. Правила именования

Перейдите к разделу, описывающему требуемые вам правила именования для:

- “Общие правила именования”
- “Правила именования объектов”
- “Сохранение регистрозависимых значений в системе объединения” на стр. 200

---

### Общие правила именования

Если не сказано иного, все имена могут содержать следующие символы:

- А - Z. В большинстве имен символы от А до Z преобразуются из строчных в прописные.
- От 0 до 9
- @, #, \$ и \_ (подчеркивание)

Имена не могут начинаться со знака номера или символа подчеркивания.

Не используйте зарезервированные слова SQL для имен таблиц, производных таблиц, столбцов, индексов или ID авторизации. Список зарезервированных слов языка SQL смотрите в справочнике *SQL Reference*.

Есть и другие специальные символы, которые могут рассматриваться по-разному в разных операционных системах, откуда вы работаете с DB2. Даже если они работают, нет гарантии, что они будут работать и дальше.

Использование этих дополнительных специальных символов в именах объектов базы данных не рекомендуется.

---

### Правила именования объектов

На все объекты распространяются общие правила именования. Кроме того, для некоторых объектов есть дополнительные ограничения, описанные ниже.

Таблица 24. Правила именования баз данных, алиасов баз данных и экземпляров

Объекты	Рекомендации
<ul style="list-style-type: none"> <li>• Базы данных</li> <li>• Алиасы баз данных</li> <li>• Экземпляры</li> </ul>	<ul style="list-style-type: none"> <li>• Имена баз данных должны быть уникальными в пределах того положения, в котором они занесены в каталог. В реализациях DB2 для систем на основе UNIX положением является путь, а в реализациях под Windows - буква диска.</li> <li>• Имена алиасов баз данных должны быть уникальными в пределах системного каталога баз данных. При создании новой базы данных ее алиасом по умолчанию считается имя базы данных. Это означает, что нельзя создать базу данных, имя которой совпадало бы с существующим алиасом, даже если базы данных с этим именем не существует.</li> <li>• Длина имен баз данных, алиасов баз данных и экземпляров не может быть больше 8 байт.</li> <li>• В системах Windows NT и Windows 2000 имя экземпляра не должно совпадать с именем какой-либо службы.</li> </ul> <p><b>Примечание:</b> Во избежание возможных проблем не используйте символы @, # и \$ в именах баз данных, к которыми предполагается обращаться по связи. Кроме того, поскольку эти символы имеются не на всех клавиатурах, не используйте их, если с базой данных будут работать на другом языке.</p>

Таблица 25. Правила именования объектов баз данных

Объекты	Рекомендации
<ul style="list-style-type: none"> <li>• Алиасы</li> <li>• Пулы буферов</li> <li>• Столбцы</li> <li>• Мониторы событий</li> <li>• Индексы</li> <li>• Методы</li> <li>• Группы узлов</li> <li>• Схемы</li> <li>• Хранимые процедуры</li> <li>• Таблицы</li> <li>• Табличные пространства</li> <li>• Триггеры</li> <li>• Пользовательские функции</li> <li>• Пользовательские типы</li> <li>• Производные таблицы</li> </ul>	<p>Могут содержать до 18 байтов, <i>кроме:</i></p> <ul style="list-style-type: none"> <li>• Имен таблиц (в том числе производных, таблиц сводок, алиасов и внутриоператорных имен), которые могут содержать до 128 байтов</li> <li>• Имен столбцов, которые могут содержать до 30 байт</li> <li>• Имен схем, которые могут содержать до 30 байт</li> <li>• Имена объектов могут также содержать: <ul style="list-style-type: none"> <li>– допустимые символы национальных алфавитов (например, ö)</li> <li>– многобайтные символы, за исключением многобайтных пробелов (в многобайтных средах)</li> </ul> </li> </ul>

## Дополнительная информация об именах схем

- Таблицы с именами схем, длина которых больше 18 байт, нельзя реплицировать.
- Имена схем пользовательских типов не могут быть длиннее 8 байтов.
- Следующие имена схем зарезервированы и использовать их нельзя: SYSCAT, SYSFUN, SYSIBM, SYSSTAT.
- Во избежание возможных проблем с перенастройкой не следует использовать имена схем, начинающиеся с SYS. менеджер баз данных не позволит вам создавать триггеры, пользовательские типы и пользовательские функции, имена схем которых начинаются с SYS.
- Не рекомендуется использовать имя схемы SESSION. К этой схеме должны относиться объявленные временные таблицы. Поэтому возможна ситуация, когда программ объявит временную таблицы с именем, идентичным имени одной из постоянных таблиц, что может привести к сложностям в работе. Не используйте схему SESSION, если вы не работаете с объявленными временными таблицами.

Таблица 26. Правила именования пользователей, ID пользователей и групп

Объекты	Рекомендации
<ul style="list-style-type: none"> <li>• Имена групп</li> <li>• Имена пользователей</li> <li>• ID пользователей</li> </ul>	<ul style="list-style-type: none"> <li>• Имена групп могут содержать до 8 байтов.</li> <li>• ID пользователей в системах на основе UNIX могут содержать до 8 символов.</li> <li>• Имена пользователей в Windows могут содержать до 30 символов. В настоящее время в Windows NT и в Windows 2000 имена не могут быть длиннее 20 символов.</li> <li>• При использовании аутентификации DCE имена пользователей не могут быть длиннее 8 символов.</li> <li>• Если не используется аутентификация DCE или аутентификация клиента, соединения клиентов из других систем (не 32-битных систем Windows) с Windows NT и Windows 2000 с именами пользователей длиннее 8 символов поддерживаются, если имя пользователя и пароль заданы в явном виде.</li> <li>• Имена и ID не должны:             <ul style="list-style-type: none"> <li>– Совпадать с именами USERS, ADMINS, GUESTS, PUBLIC, LOCAL, а также с зарезервированными словами SQL, перечисленными в справочнике <i>SQL Reference</i>.</li> <li>– Начинаться с IBM, SQL или SYS.</li> <li>– Содержать символы национальных алфавитов.</li> </ul> </li> </ul> <p>В системах на основе UNIX у групп и у пользователей могут быть одинаковые имена. В операторе GRANT надо указать, имеете ли вы в виду группу или же пользователя. В операторе REVOKE необходимость задания пользователя или группы зависит от того, есть ли в таблицах каталогов авторизации строки с одинаковыми GRANTEE, но разными значениями GRANTEETYPE.</p> <p>В системах Windows NT у локальных групп, глобальных групп и у пользователя не может быть одинаковых имен.</p> <p>В системах OS/2 у группы и у пользователя не может быть одинаковых имен.</p> <p><b>Примечания:</b></p> <ol style="list-style-type: none"> <li>1. В некоторых операционных системах ID пользователей и пароли регистрозависимы. Посмотрите в документации правила для вашей операционной системы.</li> <li>2. ID авторизации, возвращаемый успешно выполненным оператором CONNECT или ATTACH, усекается до 8 символов. К ID авторизации добавляется многоточие (...), а поля SQLWARN содержат предупреждения, указывающие на усечение. Дополнительную информацию смотрите в описании оператора CONNECT в справочнике <i>SQL Reference</i>.</li> </ol>

### Дополнительная информация о паролях

Вам может потребоваться менять пароли. Обычно такие действия выполняются на сервере, но многие пользователи слабо знакомы с работой в среде сервера, и для них подобная задача может представлять значительные трудности. В DB2

UDB есть способ изменить и проверить пароль с клиента. Например, DB2 for OS/390 Версии 5 поддерживает следующий метод изменения пароля пользователя. Если вы получили сообщение SQL1404N “Срок действия пароля истек”, измените пароль при помощи оператора CONNECT так:

```
CONNECT TO <база_данных> USER <id_пользователя> USING <пароль>
NEW <новый_пароль> CONFIRM <новый_пароль>
```

Пароль можно изменить также при помощи диалогового окна “Изменение пароля” Ассистента конфигурирования клиента DB2. Дополнительные сведения об этих методах изменения пароля смотрите в справочнике *SQL Reference* и в электронной справке Ассистента конфигурирования клиента.

Таблица 27. Правила именования объектов баз данных объединения

Объекты	Рекомендации
<ul style="list-style-type: none"> <li>• Отображения функций</li> <li>• Спецификации индексов</li> <li>• Псевдонимы</li> <li>• Серверы</li> <li>• Отображения типов</li> <li>• Отображения пользователей</li> <li>• Оболочки</li> </ul>	<ul style="list-style-type: none"> <li>• Псевдонимы, имена отображений, спецификаций индексов, серверов и оболочек не должны быть длиннее 128 байтов.</li> <li>• Опции сервера и псевдонима, а также значения опций не должны быть длиннее 255 байтов.</li> <li>• Имена объектов баз данных объединения могут также содержать: <ul style="list-style-type: none"> <li>– Допустимые буквы национальных алфавитов (например, ö)</li> <li>– В многобайтных средах - многобайтные символы, за исключением многобайтных пробелов.</li> </ul> </li> </ul>

## Использование идентификаторов с ограничителями в качестве имен объектов

Можно использовать ключевые слова. Если ключевое слово используется в контексте, где его можно принять за ключевое слово SQL, для него надо использовать ограничители.

Используя идентификаторы с ограничителями, можно создать объекты, имена которых нарушают приведенные правила; однако при последующем использовании такого объекта могут возникнуть ошибки. Например, если создать столбец, в имя которого входят знаки + или –, и впоследствии использовать этот столбец в индексе, при попытке реорганизации таблицы возникнут трудности. Подробности об ограничителях смотрите в разделе “Идентификаторы SQL” справочника *SQL Reference*.

---

## Сохранение регистрозависимых значений в системе объединения

В распределенных требованиях бывает необходимым задание идентификаторов и паролей, которые на источнике данных рассматриваются как регистрозависимые. Чтобы они правильно передавались на источник данных, выполняйте следующие правила:

- Задавайте идентификаторы и пароли в нужном регистре, заключая в двойные кавычки.
- При задании ID пользователя установите для опции сервера *fold\_id* значение "n" ("Нет, не менять регистр") для этого источника данных. При задании пароля установите для опции сервера *fold\_pw* значение "n" для этого источника данных.

Есть и другие варианты для ID пользователей и паролей. Если источник данных требует задавать ID пользователя в нижнем регистре, вы можете задать его в любом регистре и установить для опции сервера *fold\_id* значение "l" ("Посылать ID на источник данных в нижнем регистре"). Если источник данных требует задавать ID пользователя в верхнем регистре, вы можете задать его в любом регистре и установить для опции сервера *fold\_id* значение "u" ("Посылать ID на источник данных в верхнем регистре"). Аналогичным образом, если требуется задавать пароль в нижнем или в верхнем регистре, это можно сделать, задав для опции сервера *fold\_pw* значение "l" или "u".

Дополнительную информацию об опциях сервера смотрите в разделе "Использование опций сервера для определения характеристик источников данных и настройки процесса аутентификации" книги *Administration Guide: Implementation*.

- Если вы заключаете регистрозависимый идентификатор или пароль в двойные кавычки в командной строке операционной системы, необходимо убедиться в правильности анализа системой этих двойных кавычек. Для этого:
  - В операционной системе на основе UNIX - заключите весь оператор в одинарные кавычки.
  - В операционной системе Windows NT - перед каждой кавычкой ставьте обратную косую черту.

Например, многие идентификаторы с ограничителями на источниках данных DB2 регистрозависимы. Предположим, вы хотите создать псевдоним NICK1 для производной таблицы DB2 for CS "my\_schema"."wkly\_sal", которая находится на источнике данных под названием NORBASE.

В командной строке системы на основе UNIX введите:

```
db2 'create nickname nick1 for norbase."my_schema"."wkly_sal"'
```

В командной строке Windows NT введите:

```
db2 create nickname nick1 for norbase.\"my_schema\".\"wkly_sal\"
```

| Если вы вводите оператор в командной строке DB2 (интерактивный режим)  
| или задаете его в программе, одинарные кавычки или обратные косые черты  
| не нужны. Например, в командной строке DB2, независимо от операционной  
| системы, нужно ввести:

```
create nickname nick1 for norbase."my_schema"."wkly_sal"
```





---

## Приложение В. Планирование перенастройки баз данных

В этом разделе приводится обзор процесса перенастройки. Обратите внимание на то, что базы данных DB2 UDB Версия 6 не нужно перенастраивать в Версию 7. Подробную информацию о перенастройке баз данных DB2 UDB Версии 5.x можно найти в руководстве *Быстрый старт* для вашей операционной системы.

Когда вы перенастраиваете базу данных:

- Перенастраиваются следующие объекты базы данных:
  - Файл конфигурации базы данных
  - Таблицы системного каталога базы данных
  - Каталоги базы данных
  - Заголовок файла журнала базы данных
- Таблицы системного каталога изменяются следующим образом:
  - Добавляются новые столбцы.
  - Создаются новые таблицы.
  - Набор производных таблиц каталога перенастраивается, и в схеме SYSCAT создается набор новых производных таблиц.
  - В схеме SYSSTAT создается набор изменяемых производных таблиц каталога.
  - Набор скалярных функций общего назначения сохраняется, и в схеме SYSFUN создается новый набор скалярных функций общего назначения. При перенастройке базы данных только скалярная функция SYSFUN.DIFFERENCE удаляется и пересоздается.
- В каталоге базы данных создается файл хронологии базы данных и его тень. Этот файл содержит сводку информации резервного копирования, которую можно использовать при восстановлении базы данных; она обновляется при проведении определенных операций с базой данных. Сводка информации резервного копирования хранится также для операций резервного копирования и восстановления в табличном пространстве.

---

### Особенности перенастройки

Чтобы успешно перенастроить базу данных, созданную предыдущей версией менеджера базы данных, необходимо учесть следующее:

- “Ограничения перенастройки” на стр. 204
- “Защита и авторизация” на стр. 204
- “Требования к памяти” на стр. 204

- “Несовместимость выпусков” на стр. 205

## Ограничения перенастройки

Существуют определенные предварительные условия и ограничения, которые надо учитывать, прежде чем пытаться перенастроить базу данных в Версию 7:

- Перенастройка поддерживается только для Версий V5.x или V6. Перенастройка DB2 V1.2 Parallel Edition не поддерживается. Ранние версии DB2 должны быть перенастроены в Версии V5.x или V6, прежде чем перенастраивать их в V7.
- Клиент Версии V7 может передать команду перенастройки базы данных на сервер V7, но для клиентов более ранних версий DB2 эта возможность не поддерживается.
- Не поддерживается перенастройка с одной платформы на другую.
- Пользовательские объекты в базе данных не могут использовать зарезервированные имена схем V7 в качестве спецификаторов объектов. Зарезервированы следующие имена схем: SYSCAT, SYSSTAT и SYSFUN.
- Пользовательские особые типы с именами BIGINT, REAL, DATALINK или REFERENCE надо переименовать перед перенастройкой базы данных.
- Нельзя перенастраивать базу данных, которая находится в одном из следующих состояний:
  - Отложенное резервное копирование
  - Отложенный повтор
  - Одно или несколько табличных пространств в ненормальном состоянии
  - Несовместимая транзакция
- Восстановление ранних (V5.x или V6) резервных копий базы данных поддерживается, но повтор транзакций из журналов ранних версий не поддерживается.

## Защита и авторизация

Для перенастройки базы данных необходимы полномочия SYSADM.

## Требования к памяти

В процессе перенастройки требуется пространство как для старых, так и для новых каталогов. Размер требуемого пространства на диске зависит от сложности базы данных, количества и размера объектов базы данных. К объектам относятся все таблицы и производные таблицы. Необходимо освободить по крайней мере в два раза большее дисковое пространство, чем занимает каталог базы данных в текущий момент. Если дискового пространства не хватит, перенастройка завершится неудачно.

Если ваше табличное пространство SYSCAT - типа SMS, надо также учесть обновление параметров конфигурации базы данных, связанных с файлами журнала. Надо увеличить значения параметров *logfilesiz*, *logprimary* и *logsecond*, чтобы предотвратить ошибку из-за нехватки места для этих файлов (SQL1704N

с кодом причины 3). Если эта ошибка произошла, увеличьте параметры пространства для журналов и перезапустите команду MIGRATE DATABASE.

## Несовместимость выпусков

Планируя перенастройку базы данных, учтите возможность несовместимости двух версий продукта,

Чтобы воспользоваться преимуществами Версии 7, после перенастройки баз данных надо выполнить настройку базу данных и конфигурации менеджера баз данных. Советуем для этого записать и сравнить значения параметров конфигурации до и после перенастройки. (Описание команд GET DATABASE CONFIGURATION и GET DATABASE MANAGER CONFIGURATION смотрите в книге *Command Reference*.)

## Перенастройка базы данных

Ниже приведены шаги, необходимые для перенастройки базы данных. Перед перенастройкой надо запустить менеджер баз данных.

### ПЕРЕД ПЕРЕНАСТРОЙКОЙ:

**Примечание:** Подготовительные шаги необходимо выполнить в предыдущем выпуске (то есть в выпуске, установленном до перенастройки или установки нового выпуска).

1. Убедитесь, что все условия раздела “Ограничения перенастройки” на стр. 204 выполнены.
2. Отсоедините все прикладные программы и всех конечных пользователей от всех перенастраиваемых баз данных (используйте команду LIST APPLICATIONS и, если необходимо, FORCE APPLICATIONS).
3. Используйте утилиту подготовки к перенастройке DB2CKMIG, чтобы проверить, можно ли перенастроить базу данных (подробную информацию об использовании этой утилиты смотрите в книге *Быстрый старт* для вашей платформы). Обратите внимание на то, что в среде Windows NT или OS/2 вам предлагается запустить эту утилиту в процессе установки, а в системах на основе UNIX она автоматически активируется при перенастройке экземпляра.
4. Сделайте резервную копию базы данных.

Перенастройка - необратимый процесс. Если вы скопируете базу данных до изменения зарезервированных имен схем Версии 6, вы не сможете восстановить базу данных при помощи DB2 UDB Версии 7. Чтобы восстановить базу данных, вам придется воспользоваться предыдущей версией менеджера баз данных.

**Внимание!** Если у вас нет резервной копии базы данных, а перенастройка завершилась неудачно, вы не сможете восстановить базу данных ни с помощью DB2 UDB Версии 7, ни с помощью предыдущей версии менеджера баз данных.

Надо понимать также, что любые транзакции базы данных, выполненные после резервного копирования, но до завершения перехода на Версию 7, будут потеряны. Это означает, что если через некоторое время после завершения установки Версии 7 и перенастройки в нее базу данных понадобится восстановить (на уровне Версии 7), записи журнала, сделанные до установки Версии 7, нельзя будет использовать для повтора транзакций.

### ПЕРЕНАСТРОЙКА:

5. Перенастройте базу данных, используя одну из следующих команд:
  - MIGRATE DATABASE
  - RESTORE DATABASE (при восстановлении полной резервной копии базы данных).
  - sqlmgdb - API перенастройки баз данных.

**В OS/2:** Утилита перенастройки DB2CIDMG, работающая в среде Configuration/Installation/Distribution (CID), доступна только в DB2 for OS/2. Она позволяет проводить удаленную автоматическую установку и конфигурирование на сетевых рабочих станциях. Для перенастройки CID в сети должна работать NetView DM/2.

**В системах на основе UNIX:** В книге *Быстрый старт* для вашей платформы описано, что делать, если вы не хотите перенастраивать все базы данных для данного экземпляра.

### ПОСЛЕ ПЕРЕНАСТРОЙКИ:

6. (Необязательно) Воспользуйтесь утилитой DB2UDDL, чтобы облегчить управление поэтапной перенастройкой индексов уникальности в соответствии с вашим расписанием. (Для баз данных DB2 Версии 5, которые были созданы в Версии 5, эта утилита не требуется для отложенной проверки уникальности, потому что для всех индексов уникальности, созданных в Версии 5, это уже предполагается. Однако для баз данных, которые раньше были перенастроены в Версию 5, эта возможность не задана автоматически, если только вы не пользовались утилитой DB2UDDL для изменения индексов уникальности.) Эта утилита порождает операторы CREATE UNIQUE INDEX для индексов уникальности пользовательских таблиц и записывает их в файл. Запуская этот файл как командный файл процессора командной строки DB2, вы преобразуете индекс уникальности в Версию 7. Подробную информацию об использовании этой утилиты смотрите в книгах *Быстрый старт*.
7. (Необязательно) Запустите команду RUNSTATS для таблиц, которые особенно критичны для выполнения запросов SQL. Прежняя статистика остается в перенастроенной базе данных и не обновляется, пока вы не выполните команду RUNSTATS.

8. (Необязательно) Воспользуйтесь утилитой DB2RBIND для перепроверки достоверности всех пакетов, или же разрешите неявное выполнение этой операции при первом использовании пакета.
9. (Необязательно) Перенастройте таблицы объяснения, если вы собираетесь использовать их в Версии 7. Дополнительную информацию смотрите в разделе "Утилита объяснения SQL" руководства *Administration Guide: Performance*.
10. Настройте базу данных и параметры конфигурации менеджера баз данных, чтобы воспользоваться преимуществами Версии 7.



---

## Приложение С. Несовместимость 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 6 и 7 DB2 Universal Database. Они сгруппированы по следующим категориям:

- Производные таблицы системного каталога
- Прикладное программирование
- SQL
- Защита и настройка баз данных
- Утилиты и инструменты
- Возможности соединений и сосуществование
- 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

**Примечание:** Как и в DB2 Universal Database Версии 6, клиенты Версии 1.x и Версии 2.x, в том числе клиенты, поставляемые с серверами DB2 Parallel Edition Версии 1.2, более не поддерживаются.

---

## Планируемые несовместимости в 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.

### Производные таблицы только для чтения в будущей версии DB2 Universal Database

WIN	UNIX	OS/2
-----	------	------

#### Изменение

Производные таблицы системного каталога станут производными таблицами только для чтения. Производные таблицы SYSSTAT останутся изменяемыми.

#### Симптом

Операторы UPDATE, применяемые к столбцам производных таблиц SYSCAT, будут вызывать ошибки.

#### Объяснение

В коде инструмента или прикладной программы задано изменение значений в каталоге путем изменения столбца, определенного в производной таблице SYSCAT.

#### Решение

Измените инструмент или прикладную программу, чтобы менять каталог путем изменения столбца, определенного в производной таблице SYSSTAT.

### PK\_COLNAMES и FK\_COLNAMES в будущей версии DB2 Universal Database

WIN	UNIX	OS/2
-----	------	------

#### Изменение

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.



### Решение

Измените инструмент или прикладную программу, чтобы использовать вместо SYSCAT.REFERENCES производную таблицу SYSCAT.KEYCOLUSE.

## Столбец COLNAMES в будущей версии DB2 Universal Database

WIN	UNIX	OS/2
-----	------	------

### Изменение

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.

---

## Несовместимости DB2 Universal Database Версии 7

В этом разделе описаны несовместимости DB2 Universal Database Версии 7 с предыдущими версиями системы.

### 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.

Как описано в *Application Development Guide*, восьмой байт второй SQLVAR теперь может принимать значение X'12' (в дополнение к значениям X'00' и X'01'). Это значение может повлиять на поведение прикладных программы, где такая возможность не предусмотрена.

**Решение:** 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 sql/lib/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 `SQLLEJ.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 А заменяет файл jar и не обновляет классы.
2. Далее он вызывает из jar хранимую процедуру. Если этот вызов использует тот же процесс DARI, пользователь А получит устаревшую версию файла jar.
3. Пользователь В 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 удастся связать прикладную программу. При попытке связать 32-битные объекты с 64-битной библиотекой прикладных программ DB2 выдается сообщение об ошибке компоновщика операционной системы.

**Решение:** 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.

**Решение:** 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.

**Решение:** 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 временных таблиц прикладной программой:

```
select tabschema, tablename 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

### Менеджер файлов связей данных и фильтр файловых систем в Solaris

	UNIX	
--	------	--

**Изменение:** Менеджер файлов связей данных и фильтр файловых систем не поддерживаются в операционной системе Solaris OS 2.5.1.

### 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.

### Файловая система связей данных и Norton Utilities\*\*

WIN		
-----	--	--

**Изменение:** Файловая система связей данных Windows NT несовместима с Norton Utilities.

**Признак:** При удалении файла с дисководов, управляемого DLFS, происходит исключительная ситуация ядра: error 0x1E (Kernel Mode Exception Not Handled). Это исключительная ситуация 0xC0000005 (Нарушение доступа).

**Объяснение:** Это нарушение происходит из-за того, что драйвер Norton Utilities загружается после того, как загружен драйвер фильтра DLFS.

**Решение:** Временным обходным приемом для этого положения является загрузка драйвера DLFSД после драйвера Norton Utilities. Задайте загрузку драйвера DLFSД вручную: нажмите кнопку Пуск, выберите Настройка—> Панель управления—>Устройства—>DLFSД и установите для нее ручной режим.

Можно создать пакетный файл, который при запуске системы загружает драйвер DLFSД и службу DLFM. Этот пакетный файл должен выглядеть так:

```
net start dlfsd
net start "dlfm service"
```

Назовите этот пакетный файл start\_dlfs.bat и скопируйте его в каталог **WINNT\Profiles\Administrator\Start Menu\Programs\Startup**. Привилегия загружать драйвер фильтра DLFS и службу DLFM есть только у администратора.

## 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.

**Признак:** Если на клиенте и на сервере работает код Версии 7, возвращается сообщение SQL1434N; в противном случае попытка подключения или соединения завершается с SQLCODE -30081.

**Решение:** Используйте 64-битные клиенты.

---

## Несовместимости DB2 Universal Database Версии 6

В этом разделе описаны несовместимости DB2 Universal Database Версии 6 с системами ранних версий.

### Производные таблицы системных каталогов

#### Производные таблицы системных каталогов в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** В производных таблицах системных каталогов были введены новые коды: "U" для типизированных таблиц и "W" для типизированных производных таблиц.

**Признак:** Запросы поиска таблиц и производных таблиц в системных каталогах, использующие код типа "Т" для таблиц и "V" для производных таблиц, не находят более типизированные таблицы и типизированные производные таблицы.

**Объяснение:** В некоторых системных каталогах, в том числе в производных таблицах системных каталогов TABLES, PACKAGEDEP, TRIGDEP и VIEWDEP, есть столбец TYPE или BTYPE, содержащий однобуквенный код типа. В Версии 5.2 код типа "Т" использовался для всех таблиц, а "V" - для всех производных таблиц. В Версии 6 у нетипизированных таблиц код типа по-прежнему будет "Т", а для типизированных введен новый код типа "U". Подобно этому, у нетипизированных производных таблиц код типа по-прежнему будет "V", а у типизированных производных - "W". Кроме того, в таблицах системных каталогов кодом типа "Н" обозначен новый тип таблиц - иерархические таблицы; они не создаются пользователями непосредственно, а используются системой для реализации иерархий таблиц.

**Решение:** Измените инструмент или прикладную программу, чтобы распознавать коды типизированных таблиц и производных таблиц. Если для инструмента или прикладной программы важно логическое представление таблиц, используйте коды типов "Т", "U", "V" и "W". Если для инструмента или прикладной программы важно физическое представление таблиц, включая таблицы иерархии, используйте коды типов "Т" и "Н".

### Имена столбцов первичных и внешних ключей в Версии 6 DB2 Universal Database

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Тип данных у двух столбцов таблицы SYSCAT.REFERENCES - PK\_COLNAMES и FK\_COLNAMES - изменен с VARCHAR(320) на VARCHAR(640).

**Признак:** Имена столбцов первичных или внешних ключей усечены, неправильны или отсутствуют.

**Объяснение:** Когда в первичном или внешнем ключе используются имена столбцов длиной более 18 байт, невозможно соблюдать прежний формат списка имен столбцов. После столбца, длина имени которого (*n*) больше 18, 20-байтные имена столбцов, отделенные пробелами, будут сдвинуты на *n*-18 байт вправо. Если длина списка имен столбцов больше 640 байт, столбец будет содержать пустую строку.

**Решение:** Производная таблица SYSCAT.KEYCOLUSE содержит список столбцов, которые составляют первичный ключ, внешний ключ или ключ уникальности; эту таблицу надо использовать вместо столбцов из

SYSCAT.REFERENCES. Другой вариант - ограничить длину имен столбцов 18 байтами или ограничить суммарную длину списка столбцов 640 байтами.

### Столбец TEXT таблицы SYSCAT.VIEWS в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** При просмотре текста столбца TEXT таблицы SYSCAT.VIEWS он не будет разбиваться на строки. Тип данных изменен с VARCHAR(3600) на CLOB(64K).

**Признак:** Прикладная программа или инструмент не выводят полный текст столбца.

**Объяснение:** Инструменты или прикладные программы, которые ожидают значение столбца TEXT не длиннее 3600 (или 3900) байт, не могут обработать более длинные значения из этого поля. Механизм получения нескольких строк и воссоздания текста, использующий поле SEQNO, более не нужен. Значение SEQNO всегда будет равно 1.

**Решение:** Измените инструмент или прикладную программу, чтобы они могли обрабатывать значения столбца TEXT длиннее 3600 байт. Другой вариант - переписать TEXT, сократив его до 3600 байт.

### Столбец TEXT таблицы SYSCAT.STATEMENTS в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** При просмотре текста столбца TEXT в таблице SYSCAT.STATEMENTS он не будет разбиваться на строки. Тип данных изменен с VARCHAR(3600) на CLOB(64K).

**Признак:** Прикладная программа или инструмент не выводят полный текст оператора.

**Объяснение:** Инструменты или прикладные программы, которые ожидают значение столбца TEXT не длиннее 3600 (или 3900) байт, не могут обработать более длинные значения из этого поля. Механизм получения нескольких строк и воссоздания текста, использующий поле SEQNO, более не нужен. Значение SEQNO всегда будет равно 1.

**Решение:** Измените инструмент или прикладную программу, чтобы они могли обрабатывать значения столбца TEXT длиннее 3600 байт. Другой вариант - переписать TEXT, сократив его до 3600 байт.



## Столбец COLNAMES таблицы SYSCAT.INDEXES в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Тип данных столбца COLNAMES таблицы SYSCAT.INDEXES изменен с VARCHAR(320) на VARCHAR(640).

**Симптом:** Имена столбцов отсутствуют в индексе.

**Объяснение:** В коде инструмента или прикладной программы предусматривается получение данных из столбца с типом данных VARCHAR(320), поэтому они не могут обработать более длинные значения из этого поля.

**Решение:** Производная таблица SYSCAT.INDEXCOLUSE содержит список столбцов, образующих индекс; эту таблицу надо использовать вместо столбца COLNAMES. Другой вариант - удалить столбец из индекса или сократить имя столбца, чтобы длина списка имен столбцов (с ведущими + или -) не превышала 320 байт.

## Столбец TEXT таблицы SYSCAT.CHECKS в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Тип данных столбца TEXT таблицы CHECKS изменен с CLOB(32K) на CLOB(64K).

**Симптом:** Условие проверочного ограничения неполно.

**Объяснение:** В коде инструментов или прикладных программ предусматривается получение данных из столбца с типом данных CLOB(32K), поэтому они не могут обработать более длинные значения из этого поля.

**Решение:** Измените инструмент или прикладную программу, чтобы они могли обрабатывать значения столбца TEXT длиннее 32 Кбайт. Другой вариант - перепишите условие проверочного ограничения, уложившись в 32 Кбайта.

## Тип данных столбцов изменен на BIGINT в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** У нескольких столбцов производных таблиц системных каталогов тип данных был изменен с INTEGER на BIGINT.

**Симптом:** Значения (особенно статистических показателей) гораздо меньше (или больше), чем ожидалось.

**Объяснение:** В коде инструментов или прикладных программ предусматривается получение данных из столбца с типом данных INTEGER, поэтому они не могут обработать более длинные значения из этого поля.

**Решение:** Измените инструмент или прикладную программу, чтобы они могли обрабатывать значения больше максимального или меньше минимального значения поля INTEGER. Другой вариант - измените базовую структуру или код SQL, порождающий значение, которое нельзя представить как INTEGER.

### Несоответствие столбцов в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Новые столбцы не вставляются в конец производных таблиц в определении производных таблиц SYSCAT.

**Признак:** Не удастся выполнить повторную препроцессорную обработку прикладных программ из-за несоответствия столбцов или типов данных столбцов.

**Объяснение:** В производные таблицы системных каталогов введены новые столбцы; они размещены оптимальным образом для среды произвольных запросов, а именно, более короткие столбцы помещены перед длинными, а столбец REMARKS всегда идет последним.

**Решение:** Называйте столбцы в списке выбора явно вместо того, чтобы задавать их в коде "SELECT \*".

### SYSCAT.COLUMNS и SYSCAT.ATTRIBUTES в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Таблицы SYSCAT.COLUMNS и SYSCAT.ATTRIBUTES теперь содержат записи для наследуемых столбцов и атрибутов.

**Признак:** Запросы к таблице SYSCAT.COLUMNS для получения столбцов типизированной таблицы или производной таблицы, и запросы, применяемые к таблице SYSCAT.ATTRIBUTES для получения атрибутов структурного типа, могут выдать больше строк в Версии 6, чем в Версии 5.2, если предметом запроса является подтаблица, производная подтаблица, или подтип.

**Объяснение:** В Версии 5.2 для данной таблицы, производной таблицы, или структурного типа в каталогах COLUMNS и ATTRIBUTES содержались записи только для столбцов и атрибутов, введенных самой этой таблицей, производной таблицей или типом. Столбцы и атрибуты, наследуемые от надтаблиц или надтипов, в каталог не заносились. В Версии 6 COLUMNS и SYSCAT.ATTRIBUTES будут содержать записи для наследуемых столбцов и атрибутов.

**Решение:** Измените инструмент или прикладную программу, чтобы распознавать новые записи в каталогах COLUMNS и ATTRIBUTES.

### **Производные таблицы OBJCAT более не поддерживаются в DB2 Universal Database Версии 6**

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Рекурсивные производные таблицы каталога в схеме OBJCAT Версии 5.2 теперь не входят в поставляемый продукт DB2 Universal Database.

**Признак:** Запросы к производным таблицам каталогов OBJCAT завершаются неудачно.

**Решение:** Большая часть информации, хранимой раньше в производных таблицах OBJCAT, теперь включена в обычные производные таблицы каталогов SYSCAT. В большинстве случаев можно получить информацию из производных таблиц системных каталогов. Если вы выполняете перенастройку из Версии 5.2 и существуют производные таблицы каталогов OBJCAT, их надо удалить. Это можно сделать при помощи сценария CLP под именем objcatdp.db2, находящегося в подкаталоге misc каталога sql1lib.

Можно также создать собственный набор производных таблиц OBJCAT, эквивалентных производным таблицам каталогов из Версии 5.2.

В Версии 5.2 в "Приложение E" справочника *SQL Reference* пользователей предупреждали о том, что производные таблицы каталогов OBJCAT - временная мера, которая не будет поддерживаться в следующих выпусках.

### **Коды зависимостей изменены в DB2 Universal Database Версии 6**

WIN	UNIX	OS/2
-----	------	------

**Изменение:** В производных таблицах системных каталогов иерархические зависимости, ранее обозначавшиеся кодом "H", теперь обозначаются кодом "O".

**Признак:** Запросы, искавшие иерархические зависимости по коду "H" в производных таблицах каталогов, больше не будут правильно работать.

**Объяснение:** В нескольких системных каталогах, в том числе в производных таблицах системных каталогов PACKAGEDEP, TRIGDEP и VIEWDEP, есть столбец с именем BTYPE. В Версии 5.2 в производных таблицах OBJCAT иерархические зависимости обозначались кодом "H". В Версии 6 они обозначаются кодом "O".

**Решение:** Переделайте эти запросы так, чтобы они искали код "O".

### Таблицы базовых каталогов SYSIBM в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Ниже приводятся изменения в таблицах базовых каталогов SYSIBM, которые можете использовать вместо производных таблиц SYSCAT:

- Удалены поля (они остаются в производных таблицах SYSCAT):
  - SYSSTMT.SEQNO
  - SYSVIEWS.SEQNO
- Переименована таблица каталогов: SYSTRIGDEP в SYSDEPENDENCIES. Кроме того, столбцы BCREATOR и DCREATOR переименованы в BSCHEMA и DSCHEMA соответственно. Производная таблица SYSCAT.TRIGDEP не изменилась.
- Удалены поля (их не было в производных таблицах SYSCAT):
  - SYSATTRIBUTES.DEFAULT\_VALUE
  - SYSATTRIBUTES.NULLS
  - SYSCOLUMNS.SERVERTYPE
  - SYSDATATYPES.REFREP\_TYPENAME
  - SYSDATATYPES.REFREP\_TYPESCHEMA
  - SYSDATATYPES.REFREP\_LENGTH
  - SYSDATATYPES.REFREP\_SCALE
  - SYSDATATYPES.REFREP\_CODEPAGE
  - SYSINDEXES.TEXT  
(Было в производной таблице, но было зарезервировано только для будущего использования).
  - SYSPLANDEP.PUBLICPRIV
  - SYSSECTION.SEQNO
  - SYSTABAUTH.UPDATE\_BY\_COLS
  - SYSTABAUTH.REF\_BY\_COLS
  - SYSTABLES.MINPDLLENGTH
  - SYSTABLESPACES.READONLY
  - SYSTABLESPACES.REMOVABLEMEDIA

- Изменен тип данных:
  - SYSSECTION.SECTION с VARCHAR(3600) на CLOB(10M)
  - SYSPLANDEP.COLUSAGE с VARCHAR(3000) FOR BIT DATA на BLOB(5K)

## Прикладное программирование

### Тип данных VARCHAR в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Максимальный возможный размер типа данных VARCHAR (VARGRAPHIC) в Версии 6 увеличен с 4000 символов (2000 двухбайтных символов) до 32672 символов (16336 двухбайтных символов).

**Признак:** Прикладная программа, использующая буферы фиксированной длины (4000 байт) для типа данных VARCHAR (VARGRAPHIC), потенциально может перезаписать или усечь значение данных, если она читает поле VARCHAR, занимающее более 4000 байт, в слишком маленький буфер. Функция CLI - SQLGetTypeInfo() - теперь возвращает размер VARCHAR 32672 байт. Прикладные программы CLI, использующие это значение в таблице DDL, могут приводить к ошибкам, так как табличные пространства с достаточным размером страницы не доступны. Подробную информацию о размере страницы табличных пространств смотрите в разделе “Данные пользовательских таблиц” на стр. 95.

**Решение:** При написании прикладной программы рекомендуем сначала описать столбцы набора результатов (используя оператор DESCRIBE), а потом использовали буферы, размер которых определяется длиной, возвращаемой оператором DESCRIBE.

### Позиционированные UPDATE и DELETE в программировании на языке Java в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** При программировании на языке Java в Версии 6 надо учитывать, что позиционированные операторы UPDATE и DELETE по умолчанию используют идентификатор авторизации того, кто связывал пакет указателя. Это отличается от Версии 5.2, в которой использовался идентификатор авторизации того, кто запускал пакет.

**Признак:** Пакет, содержащий позиционированные операторы UPDATE и DELETE, может не быть запущен, потому что идентификатор авторизации того, кто связывает пакет, не имеет достаточных полномочий.

**Решение:** Идентификатору авторизации того, кто связывает пакет, надо дать достаточные полномочия для запуска в пакете позиционированных операторов UPDATE и DELETE. Предоставьте нужные привилегии и пересвяжите пакет.

### Изменение синтаксиса условия FOR UPDATE в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** В Версии 5.2 условие FOR UPDATE в операторе SELECT можно было использовать в программе SQLJ для идентификации столбцов, которые можно обновить в последующих операторах с указанием позиции UPDATE. В Версии 6 этот синтаксис изменен.

**Признак:** Если оператор SELECT содержит условие FOR UPDATE, вы получите сообщение об ошибке SQJ0204E.

**Решение:** Удалите условие FOR UPDATE из оператора SELECT. Задайте изменяемый итератор с помощью условия объявления итераторов. Например:

```
#sql public iterator DelByName implements sqlj.runtime.ForUpdate(String EmpNo)
with updateColumns = (salary);
```

Если вы хотите явным образом указать, какие столбцы являются изменяемыми, задайте их с помощью ключевого слова `updateColumns`, используемого совместно с условием WITH.

Дополнительную информацию об объявлении позиционированных итераторов смотрите в руководстве *Application Development Guide*.

### Размеры символьных имен в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** DB2 Universal Database Версии 6 поддерживает 128-байтные имена таблиц, производных таблиц и алиасы и 30-байтные имена столбцов. Ранее максимальная длина этих имен составляла 18 байт.

Специальные регистры USER и CURRENT SCHEMA были типа CHAR(8), а теперь - VARCHAR(128). Специальный регистр CURRENT EXPLAIN MODE был типа CHAR(8), а теперь - VARCHAR(254). Выходные параметры встроенных функций TYPE\_SCHEMA и TABLE\_SCHEMA были типа CHAR(8), а теперь - VARCHAR(128).

**Признак:** Если прикладные программы, разработанные до появления Версии 6, применяются к базе данных Версии 6, не использующей длинных имен, поведение прикладных программ не должно изменяться вообще. Однако

применение этих прикладных программ к базе данных Версии 6, где *действительно* используются более длинные имена, может привести к некоторым побочным эффектам в зависимости от того, как были написаны прикладные программы.

Ниже приведены несколько примеров:

- Рассмотрим существующую прикладную программу, которая берет имя таблицы или столбца (обычно из производной таблицы каталога) и помещает его в переменную хоста, определенную с длиной 18 байт. Так как ранее (до Версии 6) предельная длина имени таблицы или столбца составляла 18 байт, эта программа могла не проверять бит `sqlwarn1` в `SQLCA`. Она предполагает (ошибочно), что усечение никогда не происходит.
- Рассмотрим прикладную программу, которая берет имя таблицы или столбца (обычно из производной таблицы каталога) и помещает его в `SQLDA`, где размер поля `sqldata` был задан на основании поля `sqllen` из условия `DESCRIBE` в операторе `SELECT`. При этом программе будет возвращен правильный (неусеченный) результат, даже если используются длинные имена таблиц или столбцов. Если логика другой прикладной программы основана на предположении, что длина имен столбцов не превышает 18 байт, получив более длинное имя, программа может обработать его неожиданным образом; например, вывод более длинных имен столбцов может быть усечен до 18 байт.
- Поле элементов `SQLCA (sqlerrmc)` ограничено 70 байтами, и это может повлиять на существующие программы, пытающиеся вставить строку в таблицу. Реагируя на ошибку `SQL0204N`, такие программы берут имя таблицы из поля `SQLCA sqlerrmc`, а потом производят некоторые операции, основанные на этом имени объекта. В ранних версиях DB2 ограничение длины идентификатора таблицы или схемы гарантировало, что имя таблицы поместится в `SQLCA` полностью. В Версии 6 это не так.
- Прикладная программа, использующая устаревшие API, получит только первые 18 байт имени таблицы.
- Существующие прикладные программы CLI и ODBC, использующие функции схем (такие, как `SQLTables()`, `SQLColumns()` и другие), могут столкнуться с проблемами при соединении с сервером, который поддерживает длинные (больше 18 байт) имена. Хотя предупреждения об усечении выдаются, программа может не проверить их и продолжить работу с усеченным именем.

**Решение:** Лучший способ решения проблем этого типа - предусмотреть в коде прикладной программы обработку более длинных имен таблиц и столбцов. Иначе придется применять старые программы только к тем базам данных Версии 6, где длинные (более 18 байт) имена не используются.

### Изменения формата PC/IXF в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** DB2 Universal Database Версии 6 поддерживает 128-байтные имена таблиц, производных таблиц и алиасы и 30-байтные имена столбцов. Ранее максимальная длина этих имен составляла 18 байт.

**Признак:** Клиент DB2 Universal Database Версии 5 не может импортировать файл PC/IXF, экспортированный клиентом DB2 Universal Database Версии 6 (ошибка SQL3059N). Файл PC/IXF (экспортированный клиентом DB2 Universal Database Версии 6) не может быть загружен в базу данных DB2 Universal Database Версии 5 (ошибка SQL3059N).

**Решение:** Используйте для импорта и загрузки информации PC/IXF совместимые версии DB2 Universal Database.

### **SQLNAME в недублированной SQLVAR в DB2 Universal Database Версии 6**

WIN	UNIX	OS/2
-----	------	------

**Изменение:** DB2 Universal Database Версии 6 поддерживает 30-байтные имена столбцов. Предыдущая версия поддерживала 18-байтные имена. В Версии 5 в соответствии с документацией символ "0xFF" помещался в 30-й байт поля SQLNAME для недублированной SQLVAR; для имен, порожденных системой, и имен столбцов, заданных пользователем в условии "AS", в 30-й байт помещался символ "0x00".

В Версии 6 "0xFF" возвращается в 30-м байте только для порожденных системой имен.

**Признак:** Любые прикладные программы, проверяющие 30-й байт поля SQLNAME для определения, является ли имя порожденным системой или заданным пользователем, могут получить неожиданные результаты, если длина имени столбца, заданного пользователем - 30 символов. Такие случаи должны быть довольно редки.

**Решение:** Эти прикладные программы надо модифицировать, чтобы они проверяли 30-й байт поля SQLNAME на равенство "0xFF", только если длина этого поля меньше 30 символов. Если длина поля - 30 символов, это значит, что имя задано пользователем.

### **Устаревшие ключевые слова конфигурации DB2 CLI/ODBC в DB2 Universal Database Версии 6**

WIN		
-----	--	--



**Изменение:** При перенастройке в новую версию DB2 UDB вы можете изменить поведение драйвера DB2 CLI/ODBC путем задания набора необязательных ключевых слов в файле `db2cli.ini`.

Для Версии 6 ключевые слова TRANSLATEDLL и TRANSLATEOPTION устарели.

**Признак:** Если эти ключевые слова будут найдены, они будут проигнорированы. Вы можно заметить изменения поведения, вызванные игнорированием этих параметров.

**Решение:** Надо просмотреть новый список допустимых параметров и решить, какие ключевые слова и параметры подходят для вашей среды. Сведения об этих ключевых словах смотрите в книге *CLI Guide and Reference*.

### Формат выходного потока монитора событий в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** В выходные потоки монитора событий не включается номер версии. В результате добавление поддержки длинных (более 18 байтов) имен таблиц требует изменения формата выходного потока.

**Признак:** Прикладные программы, анализирующие выходные потоки монитора событий, не могут нормально работать.

**Решение:** Есть две возможности:

- Изменить прикладную программу, чтобы использовать новый потока данных.
- Задать переменную реестра

`DB20LDEVMON=evmonname1, evmonname2, ...`

где *evmonname* - имя монитора событий, для которого вы хотите задать запись в старом формате. Обратите внимание на то, чтобы все новые поля в мониторе событий не будут доступны в старом формате.

## SQL

### Столбцы DATALINK в DB2 Universal Database Версии 6

	UNIX	
--	------	--

**Изменение:** Для значений DATALINK, вставленных DB2 Universal Database Версии 6, в дескрипторе значений столбцов требуется 4 дополнительных байта.

**Признак:** Когда обновляются столбцы DATALINK, созданные в Версии 5.2, на странице набора данных требуются дополнительные 4 байта, чтобы сохранить новые значения столбцов. В результате на странице набора данных может не хватить места для завершения обновления, и ее, может быть, придется переместить на новую страницу. Это перемещение может вызвать нехватку пространства во время обновления.

**Решение:** Необходимо добавить пространства в вашу систему, чтобы изменения были возможны.

### Сигнатуры строчных функций SYSFUN в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Для многих строчных функций в схеме SYSFUN теперь есть усовершенствованные версии в схеме SYSIBM (встроенные функции). Это функции LCASE, LTRIM, RTRIM и UCASE.

**Признак:** При подготовке операторов или создании производных таблиц, возвращаемый тип данных любой из этих функций в Версии 6 может быть другим. Это происходит потому, что разрешение встроенных функций (из схемы SYSIBM) обычно происходит до разрешения функций из схемы SYSFUN.

**Решение:** Никаких действий не требуется. Встроенной функции обычно отдается предпочтение перед функцией из схемы SYSFUN. Поведение предыдущих версий можно воспроизвести, изменив путь SQL (чтобы SYSFUN шла перед SYSIBM), но это ухудшит производительность. Функцию предыдущей версии можно также вызвать, задав имя этой функции с именем схемы SYSFUN.

Перенастроенные пакеты, производные таблицы, сводные таблицы, триггеры и ограничения, ссылающиеся на эти функции, продолжают использовать версию из схемы SYSFUN, пока это не будет изменено явно, например, при явном связывании пакета или пересоздании производной таблицы, сводной таблицы, триггера или ограничения.

### SET INTEGRITY заменяет SET CONSTRAINTS

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Оператор SET CONSTRAINTS заменен оператором SET INTEGRITY. Для обратной совместимости DB2 Версии 6 и 7 воспринимают оба оператора.

## Изменение столбцов SYSTABLE с новым состоянием целостности в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Состояния "U" в столбце CONST\_CHECKED таблицы SYSCAT.TABLES изменяются иначе при выполнении оператора SET INTEGRITY ... OFF.

**Признак:** До Версии 6 любое состояние "U" в столбце CONST\_CHECKED при выполнении оператора SET INTEGRITY ... OFF заменялось на состояние "N". Теперь состояние "U" заменяется на состояние "W".

**Решение:** Никаких действий не требуется. Новое состояние "W" в столбце CONST\_CHECKED используется для того, чтобы указать, что тип ограничений ранее проверен пользователем и что некоторые данные таблицы, возможно, придется проверять на целостность.

Состояние "N" не уточняет, существуют ли старые данные, которые еще не проверены менеджером баз данных. Последующий оператор SET INTEGRITY ... IMMEDIATE CHECKED INCREMENTAL приведет к тому, что менеджер баз данных выдаст сообщение об ошибке, так как целостность данных не может быть гарантирована, если проверяются только новые изменения. С другой стороны, состояние "W" может быть обратно заменено на состояние "U" (если задана опция INCREMENTAL), чтобы указать, что пользователь по-прежнему сам отвечает за целостность данных в таблице. Если опция INCREMENTAL не задана, менеджер баз данных выберет полную обработку, заменит состояние "W" на состояние "Y", и возьмет на себя ответственность за целостность данных.

## Защита и настройка баз данных

### Создание клиентами баз данных в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Метод, используемый клиентами для создания базы данных.

**Признак:** Использование клиента устаревшей версии для создания базы данных приведет к ошибкам.

**Решение:** Если для создания базы данных используется клиент, удостоверьтесь, что на клиенте и на сервере работает код DB2 одной и той же версии.

### Привилегия SELECT, требуемая для иерархии в DB2 Universal Database Версии 6

32-битные системы Windows	UNIX	OS/2
---------------------------	------	------

**Изменение:** Задание ключевого слова ONLY (для таблицы) теперь требует, чтобы у пользователя была привилегия SELECT для всех подтаблиц заданной типизированной таблицы. Подобным образом, задание ключевого слова ONLY (для производной таблицы) теперь требует, чтобы у пользователя была привилегия SELECT для всех производных подтаблиц заданной типизированной таблицы. В предыдущих версиях DB2 привилегия SELECT требовалась только для заданной таблицы или производной таблицы.

**Признак:** Есть два возможных признака:

- Ошибка авторизации (SQLCODE -551, SQLSTATE 42501) при повторном связывании пакета, содержащего оператор SQL, в котором задано ключевое слово ONLY в условии FROM, если у ID авторизации, под которым связывался пакет, нет привилегии SELECT для подтаблиц заданной типизированной таблицы (или производной таблицы).
- Если определение производной таблицы или триггера содержит ключевое слово ONLY в условии FROM, производная таблица или триггер продолжают работать нормально. Однако определение производной таблицы или триггера нельзя использовать для создания новой производной таблицы или триггера, пока разработчик не получит привилегии SELECT для всех подтаблиц заданной таблицы (или производной таблицы).

**Решение:** Надо дать ID авторизации, под которым происходит повторное связывание пакета или создание новой производной таблицы или триггера, привилегию SELECT для всех подтаблиц (и производных подтаблиц) таблицы (или производной таблицы), заданной с ключевым словом ONLY.

### Устаревшие переменные реестра профиля и среды в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Следующие переменные реестра профиля или среды устарели:

- DB2\_VECTOR

**Решение:** Эти переменные больше не нужны.

## Утилиты и инструменты

### CURRENT EXPLAIN MODE в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Тип специального регистра "CURRENT EXPLAIN MODE" изменен с CHAR(8) на VARCHAR(254).

**Признак:** Если прикладная программа полагает, что тип этого регистра - CHAR(8), значение может быть усечено с 254 до 8 байт.

**Решение:** Переопределите тип всех переменных хоста, куда читается этот специальный регистр, с CHAR(8) на VARCHAR(254).

Это изменение требуется для двух новых значений специального регистра "CURRENT EXPLAIN MODE". Это значения "EVALUATE INDEXES" и "RECOMMEND INDEXES".

### Параметры COLUMNS и SORT BUFFER в DB2 Universal Database Версии 6

WIN	UNIX	OS/2
-----	------	------

**Изменение:** В Версии 6 более не поддерживаются параметры USING и SORT BUFFER команды LOAD. Эти параметры игнорируются.

**Признак:** Выдается предупреждение о том, что параметры USING и SORT BUFFER более не поддерживаются и будут игнорироваться утилитой загрузки.

**Решение:** Предупреждение можно игнорировать. Дополнительную информацию смотрите в руководстве *Data Movement Utilities Guide and Reference*.

## Возможности соединений и сосуществование

### Замена RUMBA на PCOMM в DB2 Universal Database Версии 6

WIN		
-----	--	--

**Изменение:** В Версии 6 RUMBA заменена на PCOMM в Windows NT, Windows 98 и Windows 95 (но не в Windows 3.1).

**Признак:** Отсутствует.

**Решение:** Отсутствует.

## Параметры конфигурации

### Устаревшие параметры конфигурации базы данных

WIN	UNIX	OS/2
-----	------	------

**Изменение:** Следующие параметры конфигурации баз данных устарели:

- DL\_NUM\_BACKUP (заменен на параметр конфигурации базы данных NUM\_DB\_BACKUP)

**Решение:** Удалите все ссылки на эти параметры из ваших программ.

---

## Приложение D. Поддержка национальных языков (NLS)

В этом разделе содержится информация о поддержке в DB2 национальных языков (national language support - NLS), включая информацию о поддерживаемых странах/регионах, языках и кодовых страницах (кодовых наборах), а также о конфигурировании и использовании функций NLS DB2 в базах данных и прикладных программах.

---

### Национальные версии

DB2 Версии 7 доступна на английском, французском, немецком, итальянском, испанском, бразильском португальском, японском, корейском, упрощенном китайском, традиционном китайском, датском, финском, норвежском, шведском, чешском, голландском, венгерском, польском, турецком, русском, болгарском и словенском языках.

---

### Поддержка кода страны/региона и кодовой страницы:

В Табл. 28 на стр. 234 показаны языки и кодовые наборы, поддерживаемые серверами баз данных, а также отображение этих значений на значения кода страны/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") или двухбайтной ("D"). 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.
- **Кодовый набор** показывает кодовый набор, связанный с поддерживаемым языком. Этот кодовый набор отображается на кодовую страницу DB2.
- **Терр** содержит идентификатор территории.
- **Код страны/региона** содержит код страны/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.
- **Название страны/региона** содержит названия стран/регионов.

Таблица 28. Поддерживаемые языки и кодовые наборы

Кодовая стр.	Группа	Кодовый набор	Код		Локаль	ОС	Название страны/региона
			Терр	страны/ региона			
----	-----	-----	--	----	-----	----	-----
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	sq_AL	AIX	Албания
850	S-1	IBM-850	AL	355	Sq_AL	AIX	Албания
819	S-1	iso88591	AL	355	-	HP	Албания
1051	S-1	roman8	AL	355	-	HP	Албания
819	S-1	ISO8859-1	AL	355	-	Sun	Албания
1252	S-1	1252	AL	355	-	WIN	Албания
1275	S-1	1275	AL	355	-	Mac	Албания
37	S-1	IBM-37	AL	355	-	HOST	Албания
1140	S-1	IBM-1140	AL	355	-	HOST	Албания
864	S-6	IBM-864	AA	785	-	OS2	арабские страны
1046	S-6	IBM-1046	AA	785	Ar_AA	AIX	арабские страны
1089	S-6	ISO8859-6	AA	785	ar_AA	AIX	арабские страны
1089	S-6	iso88596	AA	785	ar_SA.iso88596	HP	арабские страны
1256	S-6	1256	AA	785	-	WIN	арабские страны
420	S-6	IBM-420	AA	785	-	HOST	арабские страны
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	AIX	Австралия
850	S-1	IBM-850	AU	61	En_AU	AIX	Австралия
819	S-1	iso88591	AU	61	-	HP	Австралия
1051	S-1	roman8	AU	61	-	HP	Австралия
819	S-1	ISO8859-1	AU	61	en_AU	Sun	Австралия
819	S-1	ISO8859-1	AU	61	en_AU	SCO	Австралия
1252	S-1	1252	AU	61	-	WIN	Австралия
1275	S-1	1275	AU	61	-	Mac	Австралия
37	S-1	IBM-37	AU	61	-	HOST	Австралия
1140	S-1	IBM-1140	AU	61	-	HOST	Австралия
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	ge_AT	AIX	Австрия
850	S-1	IBM-850	AT	43	Ge_AT	AIX	Австрия
819	S-1	iso88591	AT	43	-	HP	Австрия
1051	S-1	roman8	AT	43	-	HP	Австрия
819	S-1	ISO8859-1	AT	43	de_AT	SCO	Австрия
819	S-1	ISO-8859-1	AT	43	de_AT	Linux	Австрия
819	S-1	ISO8859-1	AT	43	de_AT	Sun	Австрия
1252	S-1	1252	AT	43	-	WIN	Австрия
1275	S-1	1275	AT	43	-	Mac	Австрия
37	S-1	IBM-37	AT	43	-	HOST	Австрия
1140	S-1	IBM-1140	AT	43	-	HOST	Австрия



Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Группа	Кодовый набор	Код		ОС	Название страны/региона
			Терр	страны/ региона Локаль		
----	-----	-----	--	-----	-----	-----
915	S-5	IS08859-5	BY	375	-	OS2 Белоруссия
915	S-5	IS08859-5	BY	375	be_BY	AIX Белоруссия
1131	S-5	IBM-1131	BY	375	-	OS2 Белоруссия
1251	S-5	1251	BY	375	-	WIN Белоруссия
1283	S-5	1283	BY	375	-	Mac Белоруссия
1025	S-5	IBM-1025	BY	375	-	HOST Белоруссия
274	S-1	IBM-274	BE	32	-	HOST Бельгия
437	S-1	IBM-437	BE	32	-	OS2 Бельгия
850	S-1	IBM-850	BE	32	-	OS2 Бельгия
819	S-1	IS08859-1	BE	32	n1_BE	AIX Бельгия
850	S-1	IBM-850	BE	32	N1_BE	AIX Бельгия
819	S-1	iso88591	BE	32	-	HP Бельгия
819	S-1	IS08859-1	BE	32	fr_BE	SCO Бельгия
819	S-1	IS08859-1	BE	32	n1_BE	SCO Бельгия
819	S-1	ISO-8859-1	BE	32	n1_BE	Linux Бельгия
819	S-1	IS08859-1	BE	32	n1_BE	Sun Бельгия
1252	S-1	1252	BE	32	-	WIN Бельгия
1275	S-1	1275	BE	32	-	Mac Бельгия
500	S-1	IBM-500	BE	32	-	HOST Бельгия
1148	S-1	IBM-1148	BE	32	-	HOST Бельгия
855	S-5	IBM-855	BG	359	-	OS2 Болгария
915	S-5	IS08859-5	BG	359	-	OS2 Болгария
915	S-5	IS08859-5	BG	359	bg_BG	AIX Болгария
915	S-5	iso88595	BG	359	bg_BG.iso88595	HP Болгария
1251	S-5	1251	BG	359	-	WIN Болгария
1283	S-5	1283	BG	359	-	Mac Болгария
1025	S-5	IBM-1025	BG	359	-	HOST Болгария
850	S-1	IBM-850	BR	55	-	OS2 Бразилия
850	S-1	IBM-850	BR	55	-	AIX Бразилия
819	S-1	IS08859-1	BR	55	pt_BR	AIX Бразилия
819	S-1	IS08859-1	BR	55	-	HP Бразилия
819	S-1	IS08859-1	BR	55	pt_BR	SCO Бразилия
819	S-1	IS08859-1	BR	55	pt_BR	Sun Бразилия
819	S-1	ISO-8859-1	BR	55	pt_BR	Linux Бразилия
1252	S-1	1252	BR	55	-	WIN Бразилия
37	S-1	IBM-37	BR	55	-	HOST Бразилия
1140	S-1	IBM-1140	BR	55	-	HOST Бразилия

Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Группа	Кодовый набор	Код		Локаль	ОС	Название страны/региона
			Терр	страны/ региона			
----	-----	-----	--	----	-----	----	-----
850	S-1	IBM-850	CA	1	-	OS2	Канада
850	S-1	IBM-850	CA	1	En_CA	AIX	Канада
819	S-1	ISO8859-1	CA	1	en_CA	AIX	Канада
819	S-1	iso88591	CA	1	fr_CA.iso88591	HP	Канада
1051	S-1	roman8	CA	1	fr_CA.roman8	HP	Канада
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	Sun	Канада
819	S-1	ISO8859-1	CA	1	en_CA	Sun	Канада
819	S-1	ISO-8859-1	CA	1	en_CA	Linux	Канада
1252	S-1	1252	CA	1	-	WIN	Канада
1275	S-1	1275	CA	1	-	Mac	Канада
37	S-1	IBM-37	CA	1	-	HOST	Канада
1140	S-1	IBM-1140	CA	1	-	HOST	Канада
863	S-1	IBM-863	CA	2	-	OS2	Канада (французский)
<hr/>							
1381	D-4	IBM-1381	CN	86	-	OS2	Китай (КНР)
1386	D-4	GBK	CN	86	-	OS2	Китай (КНР)
1383	D-4	IBM-eucCN	CN	86	zh_CN	AIX	Китай (КНР)
1386	D-4	GBK	CN	86	Zh_CN.GBK	AIX	Китай (КНР)
1383	D-4	hp15CN	CN	86	zh_CN.hp15CN	HP	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	Sun	Китай (КНР)
1381	D-4	IBM-1381	CN	86	-	WIN	Китай (КНР)
1386	D-4	GBK	CN	86	-	WIN	Китай (КНР)
935	D-4	IBM-935	CN	86	-	HOST	Китай (КНР)
1388	D-4	IBM-1388	CN	86	-	HOST	Китай (КНР)
5488**	D-4		CN	86	-		Китай (КНР)
<p>** Кодовую страницу 5488 можно использовать только с утилитами LOAD и IMPORT для перемещения данных в кодовой странице 5488 в базу данных DB2 Unicode, или с утилитой EXPORT из базы данных DB2 Unicode в кодовую страницу 5488. Дополнительную информацию смотрите в разделе о Data Movement Utilities Guide and Reference Замечаний по выпуску Версии 7.2 FixPak 4.</p>							
<hr/>							
852	S-2	IBM-852	HR	385	-	OS2	Хорватия
912	S-2	ISO8859-2	HR	385	hr_HR	AIX	Хорватия
912	S-2	iso88592	HR	385	hr_HR.iso88592	HP	Хорватия
912	S-2	ISO8859-2	HR	385	hr_HR.ISO8859-2	SCO	Хорватия
912	S-2	ISO-8859-2	HR	385	hr_HR	Linux	Хорватия
1250	S-2	1250	HR	385	-	WIN	Хорватия
1282	S-2	1282	HR	385	-	Mac	Хорватия
870	S-2	IBM-870	HR	385	-	HOST	Хорватия

Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Группа	Кодовый набор	Терр	Код		ОС	Название страны/региона
				страны/ региона	Локаль		
----	-----	-----	--	---	-----	----	-----
852	S-2	IBM-852		CZ	421	-	OS2 Чехия
912	S-2	ISO8859-2		CZ	421	cs_CZ	AIX Чехия
912	S-2	iso88592		CZ	421	cs_CZ.iso88592	HP Чехия
912	S-2	ISO8859-2		CZ	421	cs_CZ.ISO8859-2	SCO Чехия
912	S-2	ISO-8859-2		CZ	421	cs_CZ	Linux Чехия
1250	S-2	1250		CZ	421	-	WIN Чехия
1282	S-2	1282		CZ	421	-	Mac Чехия
870	S-2	IBM-870		CZ	421	-	HOST Чехия
850	S-1	IBM-850		DK	45	-	OS2 Дания
819	S-1	ISO8859-1		DK	45	da_DK	AIX Дания
850	S-1	IBM-850		DK	45	Da_DK	AIX Дания
819	S-1	iso88591		DK	45	da_DK.iso88591	HP Дания
1051	S-1	roman8		DK	45	da_DK.roman8	HP 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	Sun Дания
819	S-1	ISO8859-1		DK	45	da	Sun Дания
819	S-1	ISO-8859-1		DK	45	da_DK	Linux Дания
1252	S-1	1252		DK	45	-	WIN Дания
1275	S-1	1275		DK	45	-	Mac Дания
277	S-1	IBM-277		DK	45	-	HOST Дания
1142	S-1	IBM-1142		DK	45	-	HOST Дания
922	S-10	IBM-922		EE	372	-	OS2 Эстония
922	S-10	IBM-922		EE	372	Et_EE	AIX Эстония
1257	S-10	1257		EE	372	-	WIN Эстония
1122	S-10	IBM-1122		EE	372	-	HOST Эстония
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_FI	AIX Финляндия
850	S-1	IBM-850		FI	358	Fi_FI	AIX Финляндия
819	S-1	iso88591		FI	358	fi_FI.iso88591	HP 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	-	Sun Финляндия
819	S-1	ISO-8859-1		FI	358	fi_FI	Linux Финляндия
1051	S-1	roman8		FI	358	-	HP Финляндия
1252	S-1	1252		FI	358	-	WIN Финляндия
1275	S-1	1275		FI	358	-	Mac Финляндия
278	S-1	IBM-278		FI	358	-	HOST Финляндия
1143	S-1	IBM-1143		FI	358	-	HOST Финляндия

Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Группа	Кодовый набор	Код страны/ региона	Локаль	ОС	Название страны/региона	
----	-----	-----	--	-----	----	-----	
855	S-5	IBM-855	MK	389	-	OS2	Македония
915	S-5	ISO8859-5	MK	389	-	OS2	Македония
915	S-5	ISO8859-5	MK	389	mk_MK	AIX	Македония
915	S-5	iso88595	MK	389	-	HP	Македония
1251	S-5	1251	MK	389	-	WIN	Македония
1283	S-5	1283	MK	389	-	Mac	Македония
1025	S-5	IBM-1025	MK	389	-	HOST	Македония
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_FR	AIX	Франция
850	S-1	IBM-850	FR	33	Fr_FR	AIX	Франция
819	S-1	iso88591	FR	33	fr_FR.iso88591	HP	Франция
1051	S-1	roman8	FR	33	fr_FR.roman8	HP	Франция
819	S-1	ISO8859-1	FR	33	fr	Sun	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	ISO-8859-1	FR	33	fr_FR	Linux	Франция
1252	S-1	1252	FR	33	-	WIN	Франция
1275	S-1	1275	FR	33	-	Mac	Франция
297	S-1	IBM-297	FR	33	-	HOST	Франция
1147	S-1	IBM-1147	FR	33	-	HOST	Франция
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_DE	AIX	Германия
850	S-1	IBM-850	DE	49	De_DE	AIX	Германия
819	S-1	iso88591	DE	49	de_DE.iso88591	HP	Германия
1051	S-1	roman8	DE	49	de_DE.roman8	HP	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	Sun	Германия
819	S-1	ISO-8859-1	DE	49	de_DE	Linux	Германия
1252	S-1	1252	DE	49	-	WIN	Германия
1275	S-1	1275	DE	49	-	Mac	Германия
273	S-1	IBM-273	DE	49	-	HOST	Германия
1141	S-1	IBM-1141	DE	49	-	HOST	Германия
819	S-1	ISO8859-1	DE	49	De_DE.88591	SINIX	Германия
819	S-1	ISO8859-1	DE	49	De_DE.6937	SINIX	Германия

Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Группа	Кодовый набор	Код		ОС	Название страны/региона
			Терр	страны/ региона Локаль		
----	-----	-----	--	-----	-----	-----
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	e1_GR AIX	Греция
813	S-7	iso88597	GR	30	e1_GR.iso88597 HP	Греция
813	S-7	ISO8859-7	GR	30	e1_GR.ISO8859-7 SCO	Греция
813	S-7	ISO-8859-7	GR	30	gr_GR Linux	Греция
737	S-7	737	GR	30	- WIN	Греция
1253	S-7	1253	GR	30	- WIN	Греция
1280	S-7	1280	GR	30	- Mac	Греция
423	S-7	IBM-423	GR	30	- HOST	Греция
875	S-7	IBM-875	GR	30	- HOST	Греция
852	S-2	IBM-852	HU	36	- OS2	Венгрия
912	S-2	ISO8859-2	HU	36	hu_HU AIX	Венгрия
912	S-2	iso88592	HU	36	hu_HU.iso88592 HP	Венгрия
912	S-2	ISO8859-2	HU	36	hu_HU.ISO8859-2 SCO	Венгрия
912	S-2	ISO-8859-2	HU	36	hu_HU Linux	Венгрия
1250	S-2	1250	HU	36	- WIN	Венгрия
1282	S-2	1282	HU	36	- Mac	Венгрия
870	S-2	IBM-870	HU	36	- HOST	Венгрия
850	S-1	IBM-850	IS	354	- OS2	Исландия
819	S-1	ISO8859-1	IS	354	is_IS AIX	Исландия
850	S-1	IBM-850	IS	354	Is_IS AIX	Исландия
819	S-1	iso88591	IS	354	is_IS.iso88591 HP	Исландия
1051	S-1	roman8	IS	354	is_IS.roman8 HP	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	- Sun	Исландия
819	S-1	ISO-8859-1	IS	354	is_IS Linux	Исландия
1252	S-1	1252	IS	354	- WIN	Исландия
1275	S-1	1275	IS	354	- Mac	Исландия
871	S-1	IBM-871	IS	354	- HOST	Исландия
1149	S-1	IBM-1149	IS	354	- HOST	Исландия

Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Группа	Кодовый набор	Код		Локаль	ОС	Название страны/региона
			Терр	страны/ региона			
----	-----	-----	--	---	-----	----	-----
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	AIX	Ирландия
850	S-1	IBM-850	IE	353	En_IE	AIX	Ирландия
819	S-1	iso88591	IE	353	-	HP	Ирландия
1051	S-1	roman8	IE	353	-	HP	Ирландия
819	S-1	ISO8859-1	IE	353	en_IE	Sun	Ирландия
819	S-1	ISO8859-1	IE	353	en_IE.ISO8859-1	SCO	Ирландия
819	S-1	ISO-8859-1	IE	353	en_IE	Linux	Ирландия
1252	S-1	1252	IE	353	-	WIN	Ирландия
1275	S-1	1275	IE	353	-	Mac	Ирландия
285	S-1	IBM-285	IE	353	-	HOST	Ирландия
1146	S-1	IBM-1146	IE	353	-	HOST	Ирландия
806	S-12	IBM-806	IN	91	hi_IN	-	Индия
1137	S-12	IBM-1137	IN	91	-	HOST	Индия
862	S-8	IBM-862	IL	972	-	OS2	Израиль
916	S-8	ISO8859-8	IL	972	iw_IL	AIX	Израиль
856	S-8	IBM-856	IL	972	Iw_IL	AIX	Израиль
916	S-8	ISO-8859-8	IL	972	iw_IL	Linux	Израиль
1255	S-8	1255	IL	972	-	WIN	Израиль
424	S-8	IBM-424	IL	972	-	HOST	Израиль
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_IT	AIX	Италия
850	S-1	IBM-850	IT	39	It_IT	AIX	Италия
819	S-1	iso88591	IT	39	it_IT.iso88591	HP	Италия
1051	S-1	roman8	IT	39	it_IT.roman8	HP	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	Sun	Италия
819	S-1	ISO-8859-1	IT	39	it_IT	Linux	Италия
1252	S-1	1252	IT	39	-	WIN	Италия
1275	S-1	1275	IT	39	-	Mac	Италия
280	S-1	IBM-280	IT	39	-	HOST	Италия
1144	S-1	IBM-1144	IT	39	-	HOST	Италия

Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Группа	Кодовый набор	Код		ОС	Название страны/региона
			Терр	страны/ региона Локаль		
---	-----	-----	--	-----	----	-----
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	IBM-eucJP		JP 81	ja_JP AIX	Япония
943*	D-1	IBM-943		JP 81	Ja_JP AIX	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	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
954	D-1	eucJP		JP 81	ja Sun	Япония
943	D-1	IBM-943		JP 81	ja_JP.PCK Sun	Япония
954	D-1	EUC-JP		JP 81	ja_JP Linux	Япония
943	D-1	IBM-943		JP 81	- WIN	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
1394**	D-1			JP 81	-	Япония
* В AIX 4.3 или новее используется кодовая страница 943. В AIX 4.2 или более ранних версиях используется кодовая страница 932.						
** Кодовую страницу 1394 можно использовать только с утилитами LOAD и IMPORT для перемещения данных в кодовой странице 1394 в базу данных DB2 Unicode, или с утилитой EXPORT из базы данных DB2 Unicode в кодовую страницу 1394. Дополнительную информацию смотрите в разделе о Data Movement Utilities Guide and Reference Замечаний по выпуску Версии 7.2 FixPak 4.						
949	D-3	IBM-949		KR 82	- OS2	Южная Корея
970	D-3	IBM-eucKR		KR 82	ko_KR AIX	Южная Корея
970	D-3	eucKR		KR 82	ko_KR.eucKR HP	Южная Корея
970	D-3	eucKR		KR 82	ko_KR.eucKR SGI	Южная Корея
970	D-3	5601		KR 82	ko Sun	Южная Корея
1363	D-3	1363		KR 82	- WIN	Южная Корея
933	D-3	IBM-933		KR 82	- HOST	Южная Корея
1364	D-3	IBM-1364		KR 82	- HOST	Южная Корея

Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Группа	Кодовый набор	Код		ОС	Название страны/региона
			Терр	страны/ региона Локаль		
----	-----	-----	--	-----	----	-----
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	- AIX	Латинская Америка
850	S-1	IBM-850		Lat 3	- AIX	Латинская Америка
819	S-1	iso88591		Lat 3	- HP	Латинская Америка
819	S-1	ISO8859-1		Lat 3	- Sun	Латинская Америка
819	S-1	ISO-8859-1		Lat 3	- Linux	Латинская Америка
1051	S-1	roman8		Lat 3	- HP	Латинская Америка
1252	S-1	1252		Lat 3	- WIN	Латинская Америка
1275	S-1	1275		Lat 3	- Mac	Латинская Америка
284	S-1	IBM-284		Lat 3	- HOST	Латинская Америка
1145	S-1	IBM-1145		Lat 3	- HOST	Латинская Америка
921	S-10	IBM-921	LV	371	- OS2	Латвия
921	S-10	IBM-921	LV	371	Lv_LV AIX	Латвия
1257	S-10	1257	LV	371	- WIN	Латвия
1112	S-10	IBM-1112	LV	371	- HOST	Латвия
921	S-10	IBM-921	LT	370	- OS2	Литва
921	S-10	IBM-921	LT	370	Lt_LT AIX	Литва
1257	S-10	1257	LT	370	- WIN	Литва
1112	S-10	IBM-1112	LT	370	- HOST	Литва
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	n1_NL AIX	Нидерланды
850	S-1	IBM-850	NL	31	N1_NL AIX	Нидерланды
819	S-1	iso88591	NL	31	n1_NL.iso88591 HP	Нидерланды
1051	S-1	roman8	NL	31	n1_NL.roman8 HP	Нидерланды
819	S-1	ISO8859-1	NL	31	n1 SCO	Нидерланды
819	S-1	ISO8859-1	NL	31	n1_NL SCO	Нидерланды
819	S-1	ISO8859-1	NL	31	n1 Sun	Нидерланды
819	S-1	ISO-8859-1	NL	31	n1_NL Linux	Нидерланды
1252	S-1	1252	NL	31	- WIN	Нидерланды
1275	S-1	1275	NL	31	- Mac	Нидерланды
37	S-1	IBM-37	NL	31	- HOST	Нидерланды
1140	S-1	IBM-1140	NL	31	- HOST	Нидерланды
850	S-1	IBM-850	NZ	64	- OS2	Новая Зеландия
850	S-1	IBM-850	NZ	64	En_NZ AIX	Новая Зеландия
819	S-1	ISO8859-1	NZ	64	en_NZ AIX	Новая Зеландия
819	S-1	ISO8859-1	NZ	64	- HP	Новая Зеландия
819	S-1	ISO8859-1	NZ	64	en_NZ SCO	Новая Зеландия
819	S-1	ISO8859-1	NZ	64	en_NZ Sun	Новая Зеландия
1252	S-1	1252	NZ	64	- WIN	Новая Зеландия
37	S-1	IBM-37	NZ	64	- HOST	Новая Зеландия
1140	S-1	IBM-1140	NZ	64	- HOST	Новая Зеландия



Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Группа	Кодовый набор	Код		ОС	Название страны/региона
			Терр	страны/ региона Локаль		
---	-----	-----	--	-----	----	-----
850	S-1	IBM-850	NO	47	-	OS2 Норвегия
819	S-1	ISO8859-1	NO	47	no_NO	AIX Норвегия
850	S-1	IBM-850	NO	47	No_NO	AIX Норвегия
819	S-1	iso88591	NO	47	no_NO.iso88591	HP Норвегия
1051	S-1	roman8	NO	47	no_NO.roman8	HP 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	Sun Норвегия
819	S-1	ISO-8859-1	NO	47	no_NO	Linux Норвегия
1252	S-1	1252	NO	47	-	WIN Норвегия
1275	S-1	1275	NO	47	-	Mac Норвегия
277	S-1	IBM-277	NO	47	-	HOST Норвегия
1142	S-1	IBM-1142	NO	47	-	HOST Норвегия
852	S-2	IBM-852	PL	48	-	OS2 Польша
912	S-2	ISO8859-2	PL	48	pl_PL	AIX Польша
912	S-2	iso88592	PL	48	pl_PL.iso88592	HP Польша
912	S-2	ISO8859-2	PL	48	pl_PL.ISO8859-2	SCO Польша
912	S-2	ISO-8859-2	PL	48	pl_PL	Linux Польша
1250	S-2	1250	PL	48	-	WIN Польша
1282	S-2	1282	PL	48	-	Mac Польша
870	S-2	IBM-870	PL	48	-	HOST Польша
860	S-1	IBM-860	PT	351	-	OS2 Португалия
850	S-1	IBM-850	PT	351	-	OS2 Португалия
819	S-1	ISO8859-1	PT	351	pt_PT	AIX Португалия
850	S-1	IBM-850	PT	351	Pt_PT	AIX Португалия
819	S-1	iso88591	PT	351	pt_PT.iso88591	HP Португалия
1051	S-1	roman8	PT	351	pt_PT.roman8	HP 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	Sun Португалия
819	S-1	ISO-8859-1	PT	351	pt_PT	Linux Португалия
1252	S-1	1252	PT	351	-	WIN Португалия
1275	S-1	1275	PT	351	-	Mac Португалия
37	S-1	IBM-37	PT	351	-	HOST Португалия
1140	S-1	IBM-1140	PT	351	-	HOST Португалия
852	S-2	IBM-852	RO	40	-	OS2 Румыния
912	S-2	ISO8859-2	RO	40	ro_RO	AIX Румыния
912	S-2	iso88592	RO	40	ro_RO.iso88592	HP Румыния
912	S-2	ISO8859-2	RO	40	ro_RO.ISO8859-2	SCO Румыния
912	S-2	ISO-8859-2	RO	40	ro_RO	Linux Румыния
1250	S-2	1250	RO	40	-	WIN Румыния
1282	S-2	1282	RO	40	-	Mac Румыния
870	S-2	IBM-870	RO	40	-	HOST Румыния

Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Группа	Кодовый набор	Код		ОС	Название страны/региона
			Терр	страны/ региона Локаль		
----	-----	-----	--	-----	-----	-----
866	S-5	IBM-866	RU	7	- OS2	Россия
915	S-5	IS08859-5	RU	7	- OS2	Россия
915	S-5	IS08859-5	RU	7	ru_RU AIX	Россия
915	S-5	iso88595	RU	7	ru_RU.iso88595 HP	Россия
915	S-5	IS08859-5	RU	7	ru_RU.IS08859-5 SCO	Россия
915	S-5	IS0-8859-5	RU	7	ru_RU Linux	Россия
1251	S-5	1251	RU	7	- WIN	Россия
1283	S-5	1283	RU	7	- Mac	Россия
1025	S-5	IBM-1025	RU	7	- HOST	Россия
855	S-5	IBM-855	SP	381	- OS2	Югославия
915	S-5	IS08859-5	SP	381	- OS2	Югославия
915	S-5	IS08859-5	SP	381	sr_SP AIX	Югославия
915	S-5	iso88595	SP	381	- HP	Югославия
1251	S-5	1251	SP	381	- WIN	Югославия
1283	S-5	1283	SP	381	- Mac	Югославия
1025	S-5	IBM-1025	SP	381	- HOST	Югославия
852	S-2	IBM-852	SK	422	- OS2	Словакия
912	S-2	IS08859-2	SK	422	sk_SK AIX	Словакия
912	S-2	iso88592	SK	422	sk_SK.iso88592 HP	Словакия
912	S-2	IS08859-2	SK	422	sk_SK.IS08859-2 SCO	Словакия
1250	S-2	1250	SK	422	- WIN	Словакия
1282	S-2	1282	SK	422	- Mac	Словакия
870	S-2	IBM-870	SK	422	- HOST	Словакия
852	S-2	IBM-852	SI	386	- OS2	Словения
912	S-2	IS08859-2	SI	386	s1_SI AIX	Словения
912	S-2	iso88592	SI	386	s1_SI.iso88592 HP	Словения
912	S-2	IS08859-2	SI	386	s1_SI.IS08859-2 SCO	Словения
912	S-2	IS0-8859-2	SI	386	s1_SI Linux	Словения
1250	S-2	1250	SI	386	- WIN	Словения
1282	S-2	1282	SI	386	- Mac	Словения
870	S-2	IBM-870	SI	386	- HOST	Словения
437	S-1	IBM-437	ZA	27	- OS2	ЮАР
850	S-1	IBM-850	ZA	27	- OS2	ЮАР
819	S-1	IS08859-1	ZA	27	en_ZA AIX	ЮАР
850	S-1	IBM-850	ZA	27	En_ZA AIX	ЮАР
819	S-1	iso88591	ZA	27	- HP	ЮАР
1051	S-1	roman8	ZA	27	- HP	ЮАР
819	S-1	IS08859-1	ZA	27	- Sun	ЮАР
819	S-1	IS08859-1	ZA	27	en_ZA.IS08859-1 SCO	ЮАР
1252	S-1	1252	ZA	27	- WIN	ЮАР
1275	S-1	1275	ZA	27	- Mac	ЮАР
285	S-1	IBM-285	ZA	27	- HOST	ЮАР
1146	S-1	IBM-1146	ZA	27	- HOST	ЮАР

Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Группа	Кодовый набор	Код		ОС	Название страны/региона	
			Терр	страны/ региона Локаль			
----	-----	-----	--	-----	-----	-----	
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_ES	AIX	Испания
850	S-1	IBM-850	ES	34	Es_ES	AIX	Испания
819	S-1	iso88591	ES	34	es_ES.iso88591	HP	Испания
1051	S-1	roman8	ES	34	es_ES.roman8	HP	Испания
819	S-1	ISO8859-1	ES	34	es	Sun	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	ISO-8859-1	ES	34	es_ES	Linux	Испания
1252	S-1	1252	ES	34	-	WIN	Испания
1275	S-1	1275	ES	34	-	Mac	Испания
284	S-1	IBM-284	ES	34	-	HOST	Испания
1145	S-1	IBM-1145	ES	34	-	HOST	Испания
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_SE	AIX	Швеция
850	S-1	IBM-850	SE	46	Sv_SE	AIX	Швеция
819	S-1	iso88591	SE	46	sv_SE.iso88591	HP	Швеция
1051	S-1	roman8	SE	46	sv_SE.roman8	HP	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	Sun	Швеция
819	S-1	ISO-8859-1	SE	46	sv_SE	Linux	Швеция
1252	S-1	1252	SE	46	-	WIN	Швеция
1275	S-1	1275	SE	46	-	Mac	Швеция
278	S-1	IBM-278	SE	46	-	HOST	Швеция
1143	S-1	IBM-1143	SE	46	-	HOST	Швеция
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	AIX	Швейцария
850	S-1	IBM-850	CH	41	De_CH	AIX	Швейцария
819	S-1	iso88591	CH	41	-	HP	Швейцария
1051	S-1	roman8	CH	41	-	HP	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	Sun	Швейцария
819	S-1	ISO-8859-1	CH	41	de_CH	Linux	Швейцария
1252	S-1	1252	CH	41	-	WIN	Швейцария
1275	S-1	1275	CH	41	-	Mac	Швейцария
500	S-1	IBM-500	CH	41	-	HOST	Швейцария
1148	S-1	IBM-1148	CH	41	-	HOST	Швейцария

Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Группа	Кодовый набор	Терр	Код		ОС	Название страны/региона
				страны/ региона	Локаль		
----	-----	-----	--	---	-----	----	-----
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	AIX	Тайвань
964	D-2	IBM-eucTW	TW	88	zh_TW	AIX	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	big5	Sun	Тайвань
964	D-2	cns11643	TW	88	zh_TW	Sun	Тайвань
950	D-2	big5	TW	88	-	WIN	Тайвань
937	D-2	IBM-937	TW	88	-	HOST	Тайвань
874	S-20	TIS620-1	TH	66	-	OS2	Таиланд
874	S-20	TIS620-1	TH	66	Th_TH	AIX	Таиланд
874	S-20	tis620	TH	66	th_TH.tis620	HP	Таиланд
874	S-20	TIS620-1	TH	66	-	WIN	Таиланд
838	S-20	IBM-838	TH	66	-	HOST	Таиланд
857	S-9	IBM-857	TR	90	-	OS2	Турция
920	S-9	ISO8859-9	TR	90	tr_TR	AIX	Турция
920	S-9	iso88599	TR	90	tr_TR.iso88599	HP	Турция
920	S-9	ISO8859-9	TR	90	tr_TR.ISO8859-9	SCO	Турция
920	S-9	ISO-8859-9	TR	90	tr_TR	Linux	Турция
1254	S-9	1254	TR	90	-	WIN	Турция
1281	S-9	1281	TR	90	-	Mac	Турция
1026	S-9	IBM-1026	TR	90	-	HOST	Турция
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	AIX	Великобритания
850	S-1	IBM-850	GB	44	En_GB	AIX	Великобритания
819	S-1	iso88591	GB	44	en_GB.iso88591	HP	Великобритания
1051	S-1	roman8	GB	44	en_GB.roman8	HP	Великобритания
819	S-1	ISO8859-1	GB	44	en_UK	Sun	Великобритания
819	S-1	ISO8859-1	GB	44	en_GB	SCO	Великобритания
819	S-1	ISO8859-1	GB	44	en	SCO	Великобритания
819	S-1	ISO-8859-1	GB	44	en_GB	Linux	Великобритания
1252	S-1	1252	GB	44	-	WIN	Великобритания
1275	S-1	1275	GB	44	-	Mac	Великобритания
285	S-1	IBM-285	GB	44	-	HOST	Великобритания
1146	S-1	IBM-1146	GB	44	-	HOST	Великобритания
819	S-1	88591	GB	44	En_GB.88591	SINIX	Великобритания
819	S-1	ISO8859-1	GB	44	En_GB.6937	SINIX	Великобритания

Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Группа	Кодовый набор	Код		ОС	Название страны/региона
			Терр	страны/ региона Локаль		
----	-----	-----	--	-----	-----	-----
1125	S-5	IBM-1125	UA	380	-	OS2 Украина
1124	S-5	IBM-1124	UA	380	uk_UA	AIX Украина
1251	S-5	1251	UA	380	-	WIN Украина
1123	S-5	IBM-1123	UA	380	-	HOST Украина
437	S-1	IBM-437	US	1	-	OS2 США
850	S-1	IBM-850	US	1	-	OS2 США
819	S-1	ISO8859-1	US	1	en_US	AIX США
850	S-1	IBM-850	US	1	En_US	AIX США
819	S-1	iso88591	US	1	en_US.iso88591	HP США
1051	S-1	roman8	US	1	en_US.roman8	HP США
819	S-1	ISO8859-1	US	1	en_US	Sun США
819	S-1	ISO8859-1	US	1	en_US	SGI США
819	S-1	ISO8859-1	US	1	en_US	SCO США
819	S-1	ISO-8859-1	US	1	en_US	Linux США
1252	S-1	1252	US	1	-	WIN США
1275	S-1	1275	US	1	-	Mac США
37	S-1	IBM-37	US	1	-	HOST США
1140	S-1	IBM-1140	US	1	-	HOST США
1163	S-11	IBM-1163	VN	84	-	OS2 Вьетнам
1163	S-11	IBM-1163	VN	84	vi_VN	AIX Вьетнам
1258	S-11	1258	VN	84	-	WIN Вьетнам
1164	S-11	IBM-1164	VN	84	-	HOST Вьетнам

Арабские страны (AA):

-----

/\* Арабский язык (Саудовская Аравия) \*/  
/\* Арабский язык (Ирак) \*/  
/\* Арабский язык (Египет) \*/  
/\* Арабский язык (Ливия) \*/  
/\* Арабский язык (Алжир) \*/  
/\* Арабский язык (Марокко) \*/  
/\* Арабский язык (Тунис) \*/  
/\* Арабский язык (Оман) \*/  
/\* Арабский язык (Йемен) \*/  
/\* Арабский язык (Сирия) \*/  
/\* Арабский язык (Иордания) \*/  
/\* Арабский язык (Ливан) \*/  
/\* Арабский язык (Кувейт) \*/  
/\* Арабский язык (Объединенные Арабские Эмираты) \*/  
/\* Арабский язык (Бахрейн) \*/  
/\* Арабский язык (Катар) \*/

Таблица 28. Поддерживаемые языки и кодовые наборы (продолжение)

Кодовая стр.	Кодовый Группа	Код набор	Код страны/ Терр региона	Локаль	ОС	Название страны/региона
Английский язык (US):						
-----						
/* Английский язык (Ямайка) */						
/* Английский язык (Страны Карибского бассейна) */						
Латинская Америка (Lat):						
-----						
/* Испанский язык (Мексика) */						
/* Испанский язык (Гватемала) */						
/* Испанский язык (Коста-Рика) */						
/* Испанский язык (Панама) */						
/* Испанский язык (Доминиканская Республика) */						
/* Испанский язык (Венесуэла) */						
/* Испанский язык (Колумбия) */						
/* Испанский язык (Перу) */						
/* Испанский язык (Аргентина) */						
/* Испанский язык (Эквадор) */						
/* Испанский язык (Чили) */						
/* Испанский язык (Уругвай) */						
/* Испанский язык (Парагвай) */						
/* Испанский язык (Боливия) */						

**Примечания:**

1. Кодовая страница 950 Solaris не поддерживает следующие символы из IBM 950:

Диапазон кодов	Описание	Sun Big-5	IBM Big-5
C6A1-C8FE	Символы	Зарезервированная область	Символы
F9D6-F9FE	Расширение ETen	Зарезервированная область	Расширение ETen
F286-F9A0	Выбранные IBM символы	Зарезервированная область	Выбранные IBM

2. В этой версии DB2 UDB поддерживает символ евро. Кодовые страницы Microsoft Windows ANSI изменены в соответствии с последним определением Microsoft и включают символ евро в позиции 0x80. Раньше этот код был не определен. Кроме того, определение кодовой страницы 850 было изменено так, что символ "i без точки" (код 0xD5) был заменен на символ евро. DB2 UDB по умолчанию использует новые определения этих кодовых страниц для поддержки символа евро. Такая реализация является подходящей установкой по умолчанию как для тех пользователей DB2 UDB, которым

требуется поддержка символа евро, так и для тех, кому она не нужна. Если вы, тем не менее, хотите продолжить использовать старые определения этих кодовых страниц, скопируйте следующие файлы:

- 12520850.cnv
- 08501252.cnv
- IBM00850.ucs
- IBM01252.ucs

из каталога

sqllib/conv/alt/

в каталог

sqllib/conv/

после завершения установки. Перед заменой файлов IBM01252.ucs и IBM00850.ucs на версии без символа евро можно сделать их резервные копии. После копирования этих файлов поддержки символа евро в DB2 UDB не будет.

---

## 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.

---

## Поддержка переведенных национальных версий на платформах на основе UNIX

Для платформ на основе UNIX можно установить сообщения продуктов DB2 и библиотеку для нескольких языков. Утилита установки DB2 помещает наборы файлов каталога сообщений в общепринятые каталоги национальных версий для данной платформы, как показано в следующих таблицах. В Табл. 29 на стр. 250

стр. 250 дается информация для AIX, HP-UX и Solaris. В Табл. 30 на стр. 251 дается информация для Linux, Linux/390, SGI и Dynix.

Если ваша система использует те же коды страниц, но другие имена национальных версий по сравнению с указанными в Табл. 29 и Табл. 30 на стр. 251, переведенные сообщения можно будет видеть, если создать связь с соответствующим каталогом сообщений.

Например, если на вашем компьютере AIX по умолчанию используется национальная версия `ja_JP.IBM-eucJP`, а кодовая страница для `ja_JP.IBM-eucJP` - 954, можно создать связь из `/usr/lpp/db2_07_01/msg/ja_JP.IBM-eucJP` в `/usr/lpp/db2_07_01/msg/ja_JP` при помощи команды:

```
ln -s /usr/lpp/db2_07_01/msg/ja_JP /usr/lpp/db2_07_01/msg/ja_JP.IBM-eucJP
```

После выполнения этой команды все сообщения DB2 выводятся на японском языке.

Таблица 29. Каталоги национальных версий для AIX, HP-UX, Solaris

Язык	AIX		HP-UX		Solaris	
	Национальная версия	Кодовая страница	Национальная версия	Кодовая страница	Национальная версия	Кодовая страница
Французский	fr_FR	819	fr_FR.iso88591	819	fr	819
	Fr_FR	850	fr_FR.roman8	1051		
Немецкий	de_DE	819	de_DE.iso88591	819	de	819
	De_DE	850	de_DE.roman8	1051		
Итальянский	it_IT	819	it_IT.iso88591	819	it	819
	It_IT	850	it_IT.roman8	1051		
Испанский	es_ES	819	es_ES.iso88591	819	es	819
	Es_ES	850	es_ES.roman8	1051		
Бразильский португальский	pt_BR	819			pt_BR	819
Японский	ja_JP	954	ja_JP.eucJP	954	ja	954
	Ja_JP	932				
Корейский	ko_KR	970	ko_KR.eucKR	970	ko	970
Упрощенный китайский	zh_CN	1383	zh_CN.hp15CN	1383	zh	1383
	Zh_CN.GBK	1386				



Таблица 29. Каталоги национальных версий для AIX, HP-UX, Solaris (продолжение)

Язык	AIX		HP-UX		Solaris	
	Национальная версия	Кодовая страница	Национальная версия	Кодовая страница	Национальная версия	Кодовая страница
Традиционный китайский	zh_TW	964	zh_TW.eucTW	964	zh_TW	964
	Zh_TW	950	zh_TW.big5	950	zh_TW.BIG5	950
Датский	da_DK	819	da_DK.iso88591	819	da	819
	Da_DK	850	da_DK.roman8	1051		
Финский	fi_FI	819	fi_FI.iso88591	819	fi	819
	Fi_FI	850	fi_FI.roman8	1051		
Норвежский	no_NO	819	no_NO.iso88591	819	нет	819
	No_NO	850	no_NO.roman8	1051		
Шведский	sv_SE	819	sv_SE.iso88591	819	sv	819
	Sv_SE	850	sv_SE.roman8	1051		
Чешский	cs_CZ	912				
Венгерский	hu_HU	912				
Польский	pl_PL	912				
Голландский	nl_NL	819				
	Nl_NL	850				
Турецкий	tr_TR	920				
Русский	ru_RU	915				
Болгарский	bg_BG	915	bg_BG.iso88595	915		
Словенский	sl_SI	912	sl_SI.iso88592	912	sl_SI	912

Таблица 30. Каталоги национальных версий для Linux, Linux/390, SGI, Dynix

Язык	Linux		Linux/390		SGI		Dynix	
	Национальная версия	Кодовая страница	Национальная версия	Кодовая страница	Национальная версия	Кодовая страница	Национальная версия	Кодовая страница
Французский	fr	819	fr	819			fr	819
Немецкий	de	819	de	819			de	819
Итальянский							es	819

Таблица 30. Каталоги национальных версий для Linux, Linux/390, SGI, Dynix (продолжение)

Язык	Linux		Linux/390		SGI		Dynix	
	Национальная версия	Кодовая страница	Национальная версия	Кодовая страница	Национальная версия	Кодовая страница	Национальная версия	Кодовая страница
Испанский								
Бразильский португальск.								
Японский	ja_JP.ujis	954	ja_JP.ujis	954			ja_JP.EUC	954
Корейский	ko	970	ko	970	ko_KO.euc	970		
Упрощенный китайский	zh_zh_CN.GBK	1386	zh_zh_CN.GBK	1386				
Традиционн. китайский	zh_TW.Big5	950	zh_TW.Big5	950				
Датский								
Финский								
Норвежский								
Шведский								
Чешский								
Венгерский								
Польский								
Голландский							nl	819
Турецкий								
Русский								
Болгарский								
Словенский								

### Наборы файлов Центра управления и документации

Наборы файлов Центра управления, справки Центра управления и документации помещаются на рабочей станции назначения в следующие каталоги:

- DB2 for AIX:
  - /usr/lpp/db2\_07\_01/cc/%L
  - /usr/lpp/db2\_07\_01/java/%L
  - /usr/lpp/db2\_07\_01/doc/%L
  - /usr/lpp/db2\_07\_01/qp/%L
  - /usr/lpp/db2\_07\_01/spb/%L

- DB2 for HP-UX:
  - /opt/IBMdb2/V7.1/cc/%L
  - /opt/IBMdb2/V7.1/java/%L
  - /opt/IBMdb2/V7.1/doc/%L
- DB2 for Linux:
  - /usr/IBMdb2/V7.1/cc/%L
  - /usr/IBMdb2/V7.1/java/%L
  - /usr/IBMdb2/V7.1/doc/%L
- DB2 for Solaris:
  - /opt/IBMdb2/V7.1/cc/%L
  - /usr/IBMdb2/V7.1/java/%L
  - /opt/IBMdb2/V7.1/doc/%L

Наборы файлов Центра управления используют кодовую страницу Unicode. Документация и справка Центра управления используют кодовую страницу, которую может распознать браузер. Если в вашей системе используется имя национальной версии, отличное от указанного, для запуска переведенной версии Центра управления и получения переведенного варианта справки надо создать связи с соответствующими языковыми каталогами.

Например, если на вашем компьютере AIX по умолчанию используется национальная версия ja\_JP.IBM-eucJP, можно создать связь из /usr/lpp/db2\_07\_01/cc/ja\_JP.IBM-eucJP в /usr/lpp/db2\_07\_01/cc/ja\_JP и из /usr/lpp/db2\_07\_01/doc/ja\_JP.IBM-eucJP в /usr/lpp/db2\_07\_01/doc/ja\_JP при помощи команд:

- **ln -s /usr/lpp/db2\_07\_01/cc/ja\_JP /usr/lpp/db2\_07\_01/cc/ja\_JP.IBM-eucJP**
- **ln -s /usr/lpp/db2\_07\_01/doc/ja\_JP /usr/lpp/db2\_07\_01/doc/ja\_JP.IBM-eucJP**

После выполнения этих команд текст Центра управления и справки выводятся на японском языке.

**Примечание:** Управляющий центр Web не поддерживается на самих серверах Linux/390 и NUMA-Q. Для управления базами данных на этих платформах его можно запустить с клиентской рабочей станции.

---

## Определение значений кодовых страниц

*Кодовая страница прикладной программы* определяется из активной среды при выполнении соединения с базой данных. Если задана переменная реестра DB2CODEPAGE, ее значение берется в качестве кодовой страницы прикладной программы. Как правило, устанавливать переменную реестра DB2CODEPAGE нет необходимости, поскольку DB2 определит необходимое значение кодовой

страницы из операционной системы. Задание для переменной реестра DB2CODEPAGE неправильного значения может привести к непредсказуемым результатам.

Если в коде программы задано использование интерфейса CLI, слой кода CLI будет в некоторых случаях использовать национальные настройки, даже если пользователь задал переменную реестра DB2CODEPAGE.

*Кодовая страница базы данных* определяется значением, указанным при создании этой базы данных (явно или по умолчанию). Ниже приводятся примеры того, как *активная среда* определяется в разных операционных системах:

<b>UNIX</b>	В операционных системах на основе UNIX активная среда определяется из значения локали (национальной версии), включающего в себя информацию о языке, территории и кодовом наборе.
<b>OS/2</b>	В OS/2 первичная и вторичная кодовые страницы указываются в файле CONFIG.SYS. Во время текущего сеанса для вывода и динамической смены кодовых страниц можно использовать команду <b>chcp</b> .
<b>Macintosh</b>	Для операционной системы Macintosh, если не задана переменная реестра DB2CODEPAGE, кодовая страница Macintosh определяется кодом региональной версии из установленного сценария.
<b>Windows</b>	Во всех 32-битных операционных системах Windows, если не установлена переменная реестра DB2CODEPAGE, кодовая страница определяется по значению кодовой страницы ANSI в реестре.

Полный список кодовых страниц для разных сред смотрите в Табл. 28 на стр. 234.

---

## Наборы символов

Обычно менеджер баз данных не ограничивает набор символов, доступный для прикладной программы. Подробное объяснение многобайтных наборов символов (multibyte character sets - MBCS), поддерживаемых DB2, смотрите в книге *Application Development Guide*.

## Набор символов для идентификаторов

Основной набор символов, который можно использовать в именах баз данных, состоит из однобайтных прописных и строчных латинских букв (A...Z, a...z), арабских цифр (0...9) и символа подчеркивания (\_). К этому списку для обеспечения совместимости с программными продуктами баз данных хоста добавляется три специальных символа (#, @ и \$). Используйте специальные символы #, @ и \$ в среде NLS с осторожностью, поскольку они не включены в инвариантный набор символов хоста NLS (EBCDIC). В зависимости от применяемой кодовой страницы можно использовать и символы из расширенного набора символов. Если вы используете базу данных в среде с несколькими кодовыми страницами, необходимо убедиться, что все кодовые страницы поддерживают все элементы из того расширенного набора символов, который вы планируете использовать.

При именовании объектов баз данных (таких, как таблицы и производные таблицы), меток программ, переменных хоста и указателей могут быть также использованы элементы из расширенного набора символов (например, русские буквы). Какие конкретно символы доступны, зависит от используемой кодовой страницы. Если вы используете базу данных в среде с несколькими кодовыми страницами, необходимо убедиться, что все кодовые страницы поддерживают все элементы из того расширенного набора символов, который вы планируете использовать. Информацию об идентификаторах с ограничителями, содержащих символы за пределами расширенного набора символов, которые можно использовать в операторах SQL, смотрите в справочнике *SQL Reference*.

### Определение расширенного набора символов для идентификаторов DBCS

В среде DBCS расширенный набор символов состоит из всех символов основного набора плюс:

- Все двухбайтные символы всех кодовых страниц DBCS, кроме двухбайтного пробела, являются допустимыми буквами.
- Двухбайтный пробел является специальным символом.
- Однобайтные символы, доступные на каждой из страниц со смешанной кодировкой, распадаются на несколько категорий:

Категория	Действительные значения кодов на каждой из страниц со смешанной кодировкой
Цифры	x30-39
Буквы	x23-24, x40-5A, x61-7A, xA6-DF (A6-DF только для кодовых страниц 932 и 942)
Специальные символы	Все другие действительные значения кодов однобайтных символов

## Кодирование операторов SQL

Кодирование операторов SQL не зависит от языка. Ключевые слова SQL можно вводить как в верхнем регистре, так и в нижнем регистре, а также в смешанном регистре. Имена объектов баз данных и переменных хоста, а также ярлыки программ в операторе SQL не могут содержать символы, выходящие за пределы расширенного набора символов, как описано в “Определение расширенного набора символов для идентификаторов DBCS” на стр. 255.

## Поддержка CCSID с двумя направлениями письма

Для правильной обработки данных с двумя направлениями письма на разных платформах требуются следующие атрибуты направления письма:

- Тип текста (LOGICAL или VISUAL)
- Форма (SHAPED или UNSHAPED)
- Ориентация (RIGHT-TO-LEFT или LEFT-TO-RIGHT)
- Форма цифр (ARABIC или HINDI)
- Симметричное обращение (YES или NO)

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 OS/390 обычно использует данные SHAPED VISUAL. Поэтому без поддержки для двунаправленных атрибутов данные, посланные из DB2 Universal Database for OS/390 в DB2 UDB 32-битной системы Windows, могут отображаться неправильно.

## CCSID с двумя направлениями письма

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.

CCSID (дес)	CCSID (шестн)	Кодовая страница	Тип строки
00420	x'01A4'	420	4
00424	x'01A8'	424	4
08612	x'21A4'	420	5
08616	x'21A8'	424	10
00856	x'0358'	856	5
00862	x'035E'	862	4
00864	x'0360'	864	5
00916	x'0394'	916	5
01046	x'0416'	1046	5
01089	x'0441'	1089	5
01255	x'04E7'	1255	5
01256	x'04E8'	1256	5
62208	x'F300'	856	4
62209	x'F301'	862	10
62210	x'F302'	916	4

62211	x'F303'	424	5
62213	x'F305'	862	5
62215	x'F307'	1255	4
62218	x'F30A'	864	4
62220	x'F30C'	856	6
62221	x'F30D'	862	6
62222	x'F30E'	916	6
62223	x'F30F'	1255	6
62224	x'F310'	420	6
62225	x'F311'	864	6
62226	x'F312'	1046	6
62227	x'F313'	1089	6
62228	x'F314'	1256	6
62229	x'F315'	424	8
62230	x'F316'	856	8
62231	x'F317'	862	8
62232	x'F318'	916	8
62233	x'F319'	420	8
62234	x'F31A'	420	9
62235	x'F31B'	424	6
62236	x'F31C'	856	10
62237	x'F31D'	1255	8
62238	x'F31E'	916	10
62239	x'F31F'	1255	10
62240	x'F320'	424	11
62241	x'F321'	856	11
62242	x'F322'	862	11
62243	x'F323'	916	11
62244	x'F324'	1255	11
62245	x'F325'	424	10
62246	x'F326'	1046	8
62247	x'F327'	1046	9
62248	x'F328'	1046	4
62249	x'F329'	1046	12
62250	x'F32A'	420	12

где типы строк CDRA определены так:

Тип строки	Тип текста	Форма цифр	Ориентация	Форма	Симметричное обращение
4	Visual	Passthru	LTR	Shaped	OFF
5	Implicit	Arabic	LTR	Unshaped	ON
6	Implicit	Arabic	RTL	Unshaped	ON
7(*)	Visual	Arabic	Contextual(*)	Unshaped-Lig	OFF
8	Visual	Arabic	RTL	Shaped	OFF
9	Visual	Passthru	RTL	Shaped	ON
10	Implicit	Passthru	Contextual-L	Unshaped	ON
11	Implicit	Passthru	Contextual-R	Unshaped	ON
12	Implicit	Arabic	RTL	Shaped	ON

**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.

### **Поддержка двух направлений письма в DB2 Universal Database**

Размещение текста с двумя направлениями письма реализуется в DB2 Universal Database при помощи новых определений 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.

Чтобы указать конкретный CCSID с двумя направлениями письма в среде (кроме DRDA), выберите в приведенной выше таблице CCSID, соответствующий характеристикам вашего клиента, и задайте это значение для 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. 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.

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. Информацию о переопределении CCSID смотрите в книге *DB2 Connect. Руководство пользователя*.

Переопределение CCSID не поддерживается для тех случаев, когда платформа хоста с кодом EBCDIC является клиентом, а DB2 UDB - сервером.



## Поддержка двух направлений письма в 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, 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.

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`.

Список поддерживаемых `CCSID` с двумя направлениями письма, а также их типы строк можно найти в разделе “`CCSID` с двумя направлениями письма” на стр. 256.

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, на которой запущен `CCSID 00424` (тип строки 4). Однако вы знаете, что для данных, содержащихся в базе данных хоста DB2, используется `CCSID 08616` (тип строки 6).

Здесь мы сталкиваемся с двумя проблемами: Во-первых, база данных хоста DB2 не знает, чем различаются типы двунаправленных строк с `CCSID 00424` и `08616`.

Во-вторых, база данных хоста DB2 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, 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. Надо использовать следующую команду catalog для базы данных хоста DB2:

```
db2 catalog dcs database nydb1 as telaviv parms ",,,,,,,BIDI=08616"
```

Эта команда заставляет DB2 Connect использовать CCSID базы данных хоста DB2 08616 вместо 00424. Такая замена включает в себя следующие действия:

1. DB2 Connect связывается с базой данных хоста DB2 с использованием CCSID 00862.
2. DB2 Connect выполняет преобразование размещения для данных, которые она собирается *послать* базе данных хоста DB2. 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. 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 удастся устранить посредством приведенных рекомендаций, установите NO для DB2BIDI.

## Последовательность сортировки

Менеджер баз данных сравнивает символьные данные при помощи *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. Например, последовательность сортировки может быть использована для того, чтобы указать, что прописные и строчные версии конкретного символа должны сортироваться одинаково.

Последовательность сортировки указывается во время создания базы данных и не может быть изменена после этого.

Менеджер баз данных позволяет создавать базы данных с произвольной последовательностью сортировки, используя интерфейс прикладного программирования (API). Информацию о создании пользовательской таблицы последовательности сортировки смотрите в книге *Application Development Guide*.

**Примечание:** Данные строк символов, определенные с атрибутом FOR BIT DATA, и данные двоичных больших объектов сортируются при помощи двоичной последовательности сортировки.

### **Общие соображения**

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. Дополнительную информацию об использовании последовательности сортировки базы данных смотрите в разделе "String Comparisons" в справочнике *SQL Reference*.

Ошибки могут происходить 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.

### **Особенности баз данных объединения**

Ваш выбор последовательности сортировки базы данных может влиять на производительность системы объединения. Если источник данных использует ту же самую последовательность сортировки, что и база данных объединения DB2, то DB2 может передать на источник данных обработку, включающую символьные данные. Если последовательность сортировки источника данных не совпадает с последовательностью сортировки DB2, данные считываются DB2, и

вся обработка с изменением уровня в отношении символьных данных выполняется локально (это может снизить производительность).

При определении, совпадают ли последовательности сортировки источника данных и DB2, учитывайте следующее:

- Поддержку национальных языков.  
Последовательность сортировки зависит от языка, поддерживаемого на сервере. Сравните информацию о NLS DB2 и NLS источника данных.
- Характеристики источника данных.  
Некоторые источники данных созданы с использованием регистронезависимых последовательностей сортировки, что в операциях, зависящих от порядка, может приводить к результатам, отличным от результатов DB2.
- Пользовательские настройки.  
Некоторые источники данных предоставляют для последовательностей сортировки различные опции или допускают пользовательскую настройку последовательности сортировки.

Выбирайте последовательность сортировки для базы данных объединения DB2 на основе совокупности источников данных, к которым с этой базы данных будет производиться доступ. Например:

- Если база данных DB2 будет обращаться в основном к базам данных Oracle с той же самой кодовой страницей (NLS), что и DB2, при создании базы данных укажите идентичную последовательность (базы данных Oracle используют эквивалентную последовательность сортировки).
- Если база данных DB2 будет обращаться только к базам данных DB2 UDB, задайте совпадающие значения последовательностей сортировки.

Информацию о задании последовательности сортировки MVS смотрите в руководстве *Administrative API Reference* (примеры после описания **sqlcrea** - Create Database API). В этих примерах содержатся таблицы последовательностей сортировки для кодовых страниц EBCDIC 500, 37 и 5026/5035.

После установки последовательности сортировки для базы данных DB2 проверьте, что опция сервера *collating\_sequence* установлена для всех серверов источников данных. Эта опция указывает, совпадает ли последовательность сортировки данного сервера источника данных с последовательностью сортировки базы данных DB2.

Задайте "Y" для опции *collating\_sequence*, если последовательности сортировки совпадают. Такая установка позволит оптимизатору DB2 рассматривать вариант обработки с зависимостью от порядка на источнике данных, что может повысить производительность. Однако если последовательность сортировки источника данных не совпадает с последовательностью сортировки базы данных

DB2, вы можете получить неверные результаты. Например, если ваш план использует объединение слияния, оптимизатор DB2 передаст операций упорядочения в максимально возможном объеме на источники данных. Если у источника данных другая последовательность сортировки, результаты объединения могут быть неверными.

Если последовательности сортировки не совпадают, задайте "N" для опции `collating_sequence`. Используйте это значение, когда последовательности сортировки источников данных и DB2 разные или когда операции сортировки источников данных могут быть регистронезависимыми. Например, на регистронезависимом источнике данных с кодовой страницей английского языка TOLLESON, ToLLeSoN и tolleson будут рассматриваться как равные. Если вы не уверены, что последовательность сортировки источника данных идентична последовательности сортировки DB2, задайте "N" для опции `collating_sequence`.

### **Сортировка символов тайского языка**

Если вы хотите правильно учитывать при сортировке специальные гласные ("ведущие гласные"), пометки тона и прочие специальные тайские символы, надо при создании базы данных задать условие `COLLATE USING NLSCHAR` в команде `CREATE DATABASE`. Синтаксис смотрите в руководстве *Command Reference*.

## **Значения даты и времени**

Ниже описываются типы данных даты и времени. Несмотря на то, что значения даты и времени могут быть использованы в определенных арифметических и строчных операциях и совместимы с определенными строками, они не являются ни строками, ни числами.

### **Дата**

*Дата* представляет собой значение из трех частей (год, месяц и день). Диапазон части года от 0001 до 9999. Диапазон части месяца от 1 до 12. Диапазон части дня от 1 до *x*, где *x* зависит от месяца.

Внутреннее представление даты - символьная строка длиной 4 байта. Каждый байт состоит из 2 упакованных десятичных цифр. Первые 2 байта представляют год, третий байт - месяц, а последний - день.

Длина столбца `DATE`, как описано в `SQLDA` - 10 байт, что позволяет представить это значение символьной строкой.

### **Время**

*Время* - значение из трех частей (часы, минуты и секунды), обозначающее время суток в 24-часовом формате. Диапазон части часов от 0 до 24. Диапазон других частей - от 0 до 59. Если значение часов 24, для минут и секунд устанавливается 0.

Внутреннее представление времени - символьная строка длиной 3 байта. Каждый байт состоит из 2 упакованных десятичных цифр. Первый байт представляет часы, второй байт - минуты, а последний - секунды.

Длина столбца TIME, как описано в SQLDA - 8 байт, что позволяет представить это значение символьной строкой.

### **Отметка времени**

*Отметка времени* - значение из семи частей (год, месяц, день, часы, минуты, секунды и микросекунды), обозначающее дату и время суток, как описано выше, за тем исключением, что во время включаются микросекунды.

Внутреннее представление отметки времени - символьная строка длиной 10 байт. Каждый байт состоит из 2 упакованных десятичных цифр. Первые 4 байта представляют дату, следующие 3 байта - время и последние 3 байта - микросекунды.

Длина столбца TIMESTAMP, как описано в SQLDA - 26 байт, что позволяет представить это значение символьной строкой.

### **Строчные представления значений даты и времени**

Значения с типом данных DATE, TIME или TIMESTAMP представляются во внутренней форме, прозрачной для пользователя SQL. Даты, времена и отметки времени, тем не менее, можно также представить символьными строками, и такие представления непосредственно затрагивают пользователя SQL, поскольку они не являются ни константами, ни переменными с типами данных DATE, TIME или TIMESTAMP. Поэтому для использования значение даты и времени должно быть назначено переменной символьной строки. Представление символьной строки обычно является используемым по умолчанию форматом значений даты и времени, ассоциированным с кодом страны/региона клиента, если только переопределяется заданием опции формата "F" при прекомпиляции программы или ее связывании с базой данных. Список форматов символьных строк для кодов различных стран/регионов смотрите в Табл. 33 на стр. 267.

Когда действительное строчное представление значения даты и времени используется в операции с внутренним значением даты и времени, это строчное представление перед выполнением такой операции преобразуется во внутреннюю форму даты, времени или отметки времени. Действительные строчные представления значений даты и времени определены в следующих разделах.

### **Строки даты**

Строчное представление даты представляет собой символьную строку, начинающуюся с цифры, длиной не менее 8 символов. Могут быть включены хвостовые пробелы; в частях месяца и дня даты могут отсутствовать начальные нули.

Действительные строчные форматы для дат перечислены в Табл. 31. Каждый формат идентифицируется названием и включает связанное с ним сокращение, а также пример использования.

*Таблица 31. Форматы для строчных представлений даты*

Название формата	Сокращение	Формат даты	Пример
ISO	ISO	гггг-мм-дд	1991-10-27
Стандарт IBM для США	USA	мм/дд/гггг	10/27/1991
Стандарт IBM для Европы	EUR	дд.мм.гггг	27.10.1991
Японский промышленный стандарт, христианское летоисчисление	JIS	гггг-мм-дд	1991-10-27
Локальный	LOC	Зависит от кода страны/региона базы данных	—

### **Строки времени**

Строчное представление времени представляет собой символьную строку, начинающуюся с цифры, длиной не менее 4 символов. Могут быть включены хвостовые пробелы; в части часов может отсутствовать начальный нуль, а секунды могут отсутствовать полностью. Если секунды не указаны, неявно подразумевается 0 секунд. Это значит, что 13.30 эквивалентно 13.30.00.

Действительные строчные форматы для времени перечислены в Табл. 32. Каждый формат идентифицируется названием и включает связанное с ним сокращение, а также пример использования.

*Таблица 32. Форматы для строчных представлений времени*

Название формата	Сокращение	Формат времени	Пример
ISO	ISO	чч.мм.сс	13.30.05
Стандарт IBM для США	USA	чч:мм AM или PM	1:30 PM
Стандарт IBM для Европы	EUR	чч.мм.сс	13.30.05
Японский промышленный стандарт, христианское летоисчисление	JIS	чч:мм:сс	13:30:05
Локальный	LOC	Зависит от кода страны/региона прикладной программы	—

### Примечания:

1. В форматах строки времени ISO, EUR и JIS .сс (или :сс) - обязательно.
2. В формате строки времени USA минуты могут отсутствовать, что неявно подразумевает 00 минут. Это значит, что 1 PM эквивалентно 1:00 PM.
3. В формате строки времени USA спецификация часов не может быть больше 12 и не может равняться 0, за исключением специального случая 00:00 AM. При использовании 24-часового формата ISO соответствие с форматом USA следующее:
  - 12:01 AM - 12:59 AM соответствует 00.01.00 - 00.59.00.
  - 01:00 AM - 11:59 AM соответствует 01.00.00 - 11.59.00.
  - 12:00 PM (полдень) - 11:59 PM соответствует 12.00.00 - 23.59.00.
  - 12:00 AM (полночь) соответствует 24.00.00, и 00:00 AM (полночь) соответствует 00.00.00.

### Строки отметки времени

Строчное представление отметки времени представляет собой символьную строку, начинающуюся с цифры, длиной не менее 16 символов. Полное представление строки отметки времени имеет вид *гггг-мм-дд-чч.мм.сс.нннннн*. Могут быть включены хвостовые пробелы; в частях месяца, дня и часов отметки времени могут отсутствовать начальные нули, а микросекунды могут быть сокращены или отсутствовать полностью. Если вы выбираете отсутствие любой цифры в части микросекунд, неявно подразумевается 0. Это значит, что 1991-3-2-8.30.00 эквивалентно 1991-03-02-08.30.00.000000.

### Особенности наборов символов

Строки отметки даты и времени должны содержать только однобайтные символы и цифры.

### Форматы даты и времени

Формат символьной строки, представляющей дату и время - это формат по умолчанию для значений даты и времени, связанный с кодом страны/региона для программы. Этот формат по умолчанию может быть переопределен, если при прекомпиляции программы или ее связывании с базой данных задана опция формата "F" .

Ниже приводится описание входных и выходных форматов для даты и времени:

- Входной формат времени
  - Для времени не существует входного формата по умолчанию
  - Любой формат допустимы на вводе для любого кода страны/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. 33 для Великобритании.)
- Выходной формат даты
  - Выходные форматы дат по умолчанию показаны в Табл. 33.

**Примечание:** В Табл. 33 приводится также список форматов строк для различных кодов стран/регионов.

Таблица 33. Форматы даты и времени для кодов стран/регионов

Код страны/региона	Местный формат даты	Местный формат времени	Выходной формат даты по умолчанию	Входные формат 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 33. Форматы даты и времени для кодов стран/регионов (продолжение)

Код страны/региона	Местный формат даты	Местный формат времени	Выходной формат даты по умолчанию	Входные формат 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 33. Форматы даты и времени для кодов стран/регионов (продолжение)

Код страны/региона	Местный формат даты	Местный формат времени	Выходной формат даты по умолчанию	Входные формат 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
<p><b>Примечания:</b></p> <ol style="list-style-type: none"> <li>1. Для стран/регионов, использующих национальную версию C по умолчанию, используется код страны/региона 001.</li> <li>2. гггг в буддийском летоисчислении эквивалентно григорианскому + 543 года (только Таиланд).</li> </ol>				

## Поддержка Unicode в DB2 UDB

Ниже описан уровень поддержки Unicode в DB2 UDB.

### Введение

Стандарт кодировки символов Unicode - схема кодировки символов с фиксированной длиной, включающая символы практически изо всех живых языков мира.

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.

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 (четырёхбайтной) версией (UCS-4). UCS-2 идентичен 16-битному варианту Unicode без замещающих символов. ISO/IEC 10646 определяет также технику расширения для кодирования некоторых символов UCS-4 двумя символами UCS-2. Это расширение, называемое 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.

Дополнительную информацию о Unicode можно найти в последнем издании книги The Unicode Standard и на Web-сайте The Unicode Consortium ([www.unicode.org](http://www.unicode.org)).

### UTF-8

Шестнадцатитбитные 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. Число байтов для каждого символа UTF-16 в формате UTF-8 можно определить из Табл. 34.

Таблица 34. 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	00000000	0xxxxxxx			
0xxxxxxx	0xxxxxxx				
00000ууу	00000ууу	110ууууу	10xxxxxx		
ууxxxxxx	ууxxxxxx				

Таблица 34. Положение битов UTF-8 (продолжение)

Код (двоичный)	UTF-16 (двоичный)	1-й байт (двоичный)	2-й байт (двоичный)	3-й байт (двоичный)	4-й байт (двоичный)
zzzzyuyu уухххххх	zzzzyuyu уухххххх	1110zzzz	10уууууу	10хххххх	
uuuu zzzzyuyu уухххххх	110110ww wwzzzyu 110111уу уухххххх	1110uuu (где uuuu = www+1)	10uuzzzz	10уууууу	10хххххх

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).

## Реализация Unicode в DB2 UDB

DB2 UDB поддерживает UCS-2, то есть Unicode без замещающих символов.

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 (смотрите предпоследнюю строку в Табл. 34 на стр. 270). 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 (смотрите последнюю строку в Табл. 34 на стр. 270).

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).

## 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. 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.

### Создание базы данных Unicode

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.

Например, чтобы создать базу 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 `sqlcrea`, необходимо соответствующим образом установить значения в `sqledbcountryinfo`. Например,

установите для SQLDBCODESET UTF-8, а для SQLDBLOCALE - любой действительный код территории (например, US).

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 большинства символов Unicode их порядок слияния при кодировке UTF-8 и UCS-2 совпадает. Однако если вы используете символ расширения, 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. Но в UCS-2 тот же символ расширения кодируется как два замещающих символа UCS-2, и DB2 UDB будет рассматривать их при сортировке как отдельные.

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. Число байтов для каждого символа UCS-2 в формате UTF-8 можно определить из таблицы в разделе “UTF-8” на стр. 270.

## Типы данных

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 скалярные 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). Описание поведения этих функций смотрите в справочнике *SQL Reference*.

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 sql dbchar. Обратите внимание на то, что для базы данных UCS-2 данные sql dbchar 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.

### Идентификаторы

В базе данных UCS-2 все идентификаторы находятся в многобайтном UTF-8. Поэтому можно использовать любые символы UCS-2 в идентификаторах, в которых использование символа из расширенного набора символов (например, русской буквы или многобайтного символа) разрешено DB2 UDB. Подробнее о том, какие идентификаторы позволяют использовать расширенные символы, смотрите в разделе “Приложение А. Правила именования” на стр. 195.

Клиенты могут вводить любые символы, поддерживаемые их средой; все эти символы в идентификаторах будут преобразованы менеджером баз данных в UTF-8. При задании символов национальных языков в идентификаторах для базы данных UCS-2 следует учитывать два обстоятельства:

- Для каждого символа, кроме символов ASCII, требуется от двух до четырех байт. Поэтому  $n$ -байтный идентификатор может содержать только от  $n/4$  до  $n$  символов в зависимости доли символов ASCII в нем. Если у вас только один-два символа не относятся к ASCII (например, в слове есть символ с диакритическим значком), значение ближе к  $n$  символам, а идентификатор, ни один символ которого не входит в ASCII (например, на японском языке), может содержать от  $n/4$  до  $n/3$  символов.
- Если идентификаторы будут вводиться из разных сред клиентов, они должны быть определены с использованием общего подмножества символов, доступных этим клиентам. Например, если обращение к базе данных UCS-2 производится из сред с латиницей-1, арабским и японским языками, все идентификаторы реально должны ограничиваться ASCII.

### Литералы Unicode

Литералы Unicode могут быть заданы двумя способами:

- Как константа графической строки с использованием формата G'...' или N'....', описанного в разделе "Graphic String Constants" главы "Language Elements" справочника *SQL Reference*. Любой заданный таким образом литерал будет преобразован менеджером баз данных из кодовой страницы прикладной программы в 16-битный Unicode.
- Как шестнадцатеричная строка Unicode с использованием 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).

### **Поиск по шаблону в базе данных UCS-2**

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. Обратите внимание на то, что для представления заменяющей пары нужно два знака подчеркивания.

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).

### **Особенности импорта, экспорта и загрузки**

Для базы данных UCS-2 поддерживаются форматы файлов DEL, ASC и PC/IXF, как описано в этом разделе. Формат WSF не поддерживается.

При экспорте из базы данных UCS-2 в файл ASCII с ограничителями (DEL) все символьные данные преобразуются в кодовую страницу прикладной программы. И данные символьных строк, и данные графических строк преобразуются в одну и ту же кодовую страницу SBCS или MBCS клиента. Такое поведение ожидается для экспорта любой базы данных и не может быть изменено, поскольку у всего файла ASCII с ограничителями может быть только одна кодовая страница. Поэтому при экспорте файла ASCII с ограничителями будут сохранены только те символы UCS-2, которые существуют в кодовой странице вашей прикладной программы. Другие символы заменяются на используемый по умолчанию символ замены для кодовой страницы прикладной программы. Для клиентов UTF-8 (кодовая страница 1208) потери данных не происходит, поскольку все символы UCS-2 поддерживаются клиентами UTF-8.

При импорте из файла ASCII (DEL или ASC) в базу данных UCS-2 данные символьных строк преобразуются из кодовой страницы прикладной программы в UTF-8, а данные графических строк преобразуются из кодовой страницы прикладной программы в UCS-2. Потери данных не происходит. Если вы хотите импортировать данные ASCII, которые были сохранены в другой кодовой странице, перед использованием команды IMPORT необходимо изменить кодовую страницу этого файла данных. Одним из способов сделать это - задать DB2CODEPAGE для кодовой страницы файла данных ASCII.

Диапазон действительных ограничений ASCII для клиентов SBCS и MBCS идентичен тому, что в данное время поддерживается для этих клиентов DB2 UDB. Диапазон действительных ограничителей для клиентов UTF-8 - от X'01' до X'7F' с обычными ограничениями. Полный список этих ограничений смотрите в приложении "Export/Import/Load Utility File Formats" к книге *Data Movement Utilities Guide and Reference*.

При экспорте из базы данных UCS-2 в файл PC/IXF данные символьных строк преобразуются в кодовую страницу SBCS/MBCS клиента. Данные графических строк не преобразуются и сохраняются в UCS-2 (кодовая страница 1200). Потери данных не происходит.

При импорте из файла PC/IXF в базу данных UCS-2 данные символьных строк считаются данными в кодовой странице SBCS/MBCS, хранящейся в заголовке PC/IXF, а данные графических строк считаются данными в кодовой странице DBCS, хранящейся в заголовке PC/IXF. Данные символьных строк преобразуются утилитой импорта из кодовой страницы, указанной в заголовке PC/IXF в кодовую страницу клиента, а затем из кодовой страницы клиента в UTF-8 (при помощи оператора INSERT). Данные графических строк преобразуются утилитой импорта из кодовой страницы DBCS, указанной в заголовке PC/IXF, непосредственно в UCS-2 (кодовая страница 1200).

Утилита загрузки помещает данные непосредственно в базу данных и по умолчанию считает, что данные в файлах ASC или DEL будут находиться в кодовой странице базы данных. Поэтому по умолчанию для файлов ASCII преобразования кодовых страниц не происходит. Когда для файла данных явно указана кодовая страница (с использованием модификатора `codepage`), утилита загрузки перед загрузкой данных использует эту информацию для преобразования из указанной кодовой страницы в кодовую страницу базы данных. Для файлов PC/IXF утилита загрузки всегда выполняет преобразование из кодовых страниц, указанных в заголовке IXF, в кодовую страницу базы данных (1208 для CHAR и 1200 для GRAPHIC).

Кодовая страница файлов DBCLOB для UCS-2 - всегда 1200. Кодовая страница для файлов CLOB - та же самая, что и кодовая страница для импортируемых, загружаемых или экспортируемых файлов данных. Например, при загрузке или импорте дата с использованием формата PC/IXF считается, что файл большого символьного объекта использует кодовую страницу, указанную в заголовке PC/IXF. Если файл двухбайтного символьного большого объекта находится в формате ASC или DEL, утилита загрузки считает, что для данных символьного большого объекта используется кодовая страница базы данных (если явным образом при помощи модификатора `codepage` не указано иное), а утилита импорта считает, что для них используется кодовая страница прикладной программы клиента.

Для базы данных UCS-2 модификатор `nochecklengths` указывается всегда, поскольку:

- Любой SBCS может быть связан с базой данных, для которой нет кодовой страницы DBCS
- У символьных строк в формате UTF-8 длина обычно отличается от их длины в кодовых страницах клиентов.

Дополнительную информацию об утилитах загрузки, импорта и экспорта смотрите в руководстве *Data Movement Utilities Guide and Reference*.

### **Несовместимости**

В прикладных программах, связанных с базой данных UCS-2, для данных графических строк всегда используется UCS-2 (кодировка 1200). В прикладных программах, связанных с другими базами данных, для данных графических строк используется кодировка DBCS прикладной программы, если же кодировка прикладной программы - кодировка SBCS, такие данные не разрешены. Например, если клиент 932 связан с базой данных (не UCS-2) с японским языком, для данных графических строк используется кодировка 301. В прикладных программах клиента 932, связанных с базой данных UCS-2, для данных графических строк используется UCS-2.



---

## Приложение Е. Использование библиотеки DB2

Библиотека DB2 Universal Database состоит из электронной справки, книг (в формате PDF и HTML) и примеров программ в формате HTML. В этом разделе объясняется, какая информация содержится в ней и как ее получить.

Для оперативного доступа к этой информации можно использовать Информационный центр. Дополнительную информацию смотрите в разделе “Доступ к информации через Информационный центр” на стр. 296. Вы можете просматривать сведения о задачах, книги DB2, информацию по устранению неисправностей, программы примеров и информацию по DB2 в Web.

---

### Файлы PDF и печатные книги DB2

#### Информация DB2

В следующей таблице книги DB2 разделены на 4 категории:

##### **Руководства и справочники по DB2**

В этих книгах содержится информация по DB2, общая для всех платформ.

##### **Информация по установке и конфигурированию DB2**

Эти книги применимы к DB2 для конкретной платформы. Например, есть отдельные книги *Быстрый старт* для DB2 на OS/2, Windows и на платформах на основе UNIX.

##### **Кроссплатформенные программы примеров в формате HTML**

Эти примеры - HTML-версии программ примеров, которые устанавливаются с клиентом разработки программ. Примеры используются для справок и не заменяют самих программ.

##### **Замечания по выпуску**

Эти файлы содержат самую свежую информацию, которую не успели включить в книги по DB2.

Руководства по установке, замечания по выпуску и обучающие книги в формате HTML можно просматривать прямо на компакт-диске. Большинство книг доступны в формате HTML на компакт-диске данного продукта (для просмотра) и в формате Adobe Acrobat (PDF) на компакт-диске публикаций DB2 (для просмотра и печати). Можно также заказать печатные копии в IBM; смотрите раздел “Заказ печатных копий” на стр. 292. Ниже в таблице перечислены книги, которые можно заказать.

На платформах OS/2 и Windows файлы в формате HTML можно установить в каталог `sqllib\doc\html`. Информация о DB2 переведена на различные языки, однако не на каждом языке доступна вся информация. Если информация на конкретном языке недоступна, приводится информация на английском языке.

На платформах UNIX вы можете установить версии файлов в формате HTML на нескольких языках в подкаталоги `doc/%L/html`, где `%L` - обозначение вашей национальной версии. Дополнительную информацию смотрите в соответствующей книге *Quick Beginnings* (Быстрый старт).

Вызвать книги DB2 и обратиться к информации в них можно разными способами:

- “Просмотр информации на экране” на стр. 295
- “Поиск электронной информации” на стр. 300
- “Заказ печатных копий” на стр. 292
- “Печать книг PDF” на стр. 291

Таблица 35. Информация DB2

Название	Описание	Номер формы	Каталог HTML
		Имя файла PDF	
<b>Руководства и справочники по DB2</b>			
<i>Administration Guide</i>	<i>Administration Guide: Planning</i> содержит обзор понятий баз данных, информацию по вопросам разработки (в частности, по логическому и физическому проектированию баз данных) и обсуждение доступности баз данных.	SH43-0146 db2d1x70	db2d0
	<i>Administration Guide: Implementation</i> содержит информацию о реализации ваших проектов, доступе к базам данных, аудите, резервном копировании и восстановлении.	SH43-0144 db2d2x70	
	<i>Administration Guide: Performance</i> содержит информацию о среде баз данных, оценке и настройке производительности программ.	SH43-0145 db2d3x70	
Эти три тома <i>Administration Guide</i> можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8934.			



Таблица 35. Информация DB2 (продолжение)

Название	Описание	Номер формы	Каталог HTML
		Имя файла PDF	
<i>Administrative API Reference</i>	Описывает интерфейсы прикладного программирования (API) DB2 и структуры данных, которые можно использовать при работе с вашими базами данных. Эта книга также объясняет, как вызывать API из ваших программ.	SC09-2947	db2b0
		db2b0x70	
<i>Application Building Guide</i>	Содержит информацию о настройке среды и пошаговые инструкции для компиляции, компоновки и запуска программ DB2 в системах Windows, OS/2 и на платформах на базе UNIX.	SC09-2948	db2ax
		db2axx70	
<i>APPC, C/PI-C, and SNA Sense Codes</i>	Содержит общие сведения о смысловых кодах APPC, C/PI-C и SNA, которые могут встретиться вам при работе с продуктами DB2 Universal Database.	Номера формы нет	db2ap
	Существует только в формате HTML.	db2apx70	
<i>Application Development Guide</i>	Объясняет, как разрабатывать программы, обращающиеся к базам данных DB2 с использованием встроенного SQL или Java (JDBC и SQLJ). Эта книга содержит обсуждение программирования хранимых процедур, пользовательских функций, создания пользовательских типов, использования триггеров и разработки прикладных программ для работы в многораздельной среде и в системах объединения.	SC09-2949	db2a0
		db2a0x70	
<i>CLI Guide and Reference</i>	Объясняет, как разрабатывать программы, обращающиеся к базам данных DB2 при помощи интерфейса уровня вызовов (CLI) DB2 - интерфейса SQL, совместимого со спецификациями Microsoft ODBC.	SC09-2950	db210
		db210x70	
<i>Command Reference</i>	Объясняет, как использовать процессор командной строки, и описывает команды DB2, которые можно использовать для управления вашей базой данных.	SC09-2951	db2n0
		db2n0x70	

Таблица 35. Информация DB2 (продолжение)

Название	Описание	Номер формы	Каталог HTML
		Имя файла PDF	
<i>Дополнение по возможностям соединений</i>	Содержит установочную и справочную информацию по использованию DB2 for AS/400, DB2 for OS/390, DB2 for MVS, или DB2 for VM как реквестеров прикладных программ DRDA с серверами DB2 Universal Database. В этой книге описано также использование серверов прикладных программ DRDA с реквестерами прикладных программ DB2 Connect.  Эта книга доступна только в форматах HTML и PDF.	Номера формы нет db2h1x70	db2h1
<i>Data Movement Utilities Guide and Reference</i>	Объясняет, как использовать утилиты DB2, в частности, import, export, load, AutoLoader и DPROF, которые упрощают перемещение данных.	SC09-2955 db2dmx70	db2dm
<i>Data Warehouse Center Administration Guide</i>	Содержит сведения о том, как построить и обслуживать хранилище данных при помощи Центра хранилища данных.	SC26-9993 db2ddx70	db2dd
<i>Data Warehouse Center Application Integration Guide</i>	Содержит информацию, которая поможет программистам интегрировать прикладные программы с Центром хранилища данных и Менеджером каталога данных.	SC26-9994 db2adx70	db2ad
<i>DB2 Connect. Руководство пользователя</i>	Содержит информацию по основным понятиям, программированию и общим вопросам использования продуктов DB2 Connect.	SH43-0130 db2c0x70	db2c0
<i>DB2 Query Patroller Administration Guide</i>	Содержит обзор системы DB2 Query Patroller, информацию по использованию и управлению, а также сведения по выполнению заданий при помощи утилит управления с графическим интерфейсом.	SC09-2958 db2dwx70	db2dw
<i>DB2 Query Patroller User's Guide</i>	Объясняет, как использовать средства и функции DB2 Query Patroller.	SC09-2960 db2wwx70	db2ww
<i>Глоссарий</i>	Содержит определения терминов, используемых в DB2 и его компонентах.  Доступен в формате HTML, а также в книге <i>SQL Reference</i> .	Номера формы нет db2t0x70	db2t0

Таблица 35. Информация DB2 (продолжение)

Название	Описание	Номер формы	Каталог HTML
		Имя файла PDF	
<i>Image, Audio, and Video Extenders Administration and Programming</i>	Содержит общую информацию о модулях расширения DB2, о конфигурировании модулей расширения для работы с изображениями, звуком и видео (IAV), об управлении ими и о программировании с использованием модулей расширения IAV. Включает в себя справочную информацию, диагностическую информацию (с сообщениями) и примеры.	SC26-9929 dmbu7x70	dmbu7
<i>Information Catalog Manager Administration Guide</i>	Руководство по управлению каталогами данных.	SC26-9995 db2dix70	db2di
<i>Information Catalog Manager Programming Guide and Reference</i>	Содержит определения для проектирования интерфейсов менеджера каталогов данных.	SC26-9997 db2bix70	db2bi
<i>Information Catalog Manager User's Guide</i>	Содержит информацию об использовании пользовательского интерфейса менеджера каталога данных.	SC26-9996 db2aix70	db2ai
<i>Дополнение по установке и настройке</i>	Помогает планировать, устанавливать и конфигурировать клиенты DB2 для конкретных платформ. Это дополнение содержит также информацию по связыванию, конфигурированию связей клиента и сервера, инструментам DB2 с графическим интерфейсом, DRDA AS, распределенной установке, конфигурации распределенных запросов и доступу к неоднородным источникам данных.	GP43-0126 db2iyx70	db2iy
<i>Справочник по сообщениям</i>	Содержит список сообщений и кодов, выдаваемых DB2, Information Catalog Manager, и Data Warehouse Center, и описывает для них рекомендуемые действия.  Оба тома Справочник по сообщениям можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8932.	Том 1 GH43-0128  db2m1x70 Том 2 GH43-0129  db2m2x70	db2m0
<i>OLAP Integration Server Administration Guide</i>	Объясняет, как использовать менеджер управления сервером OLAP Integration Server.	SC27-0782 db2dpx70	нет

Таблица 35. Информация DB2 (продолжение)

Название	Описание	Номер формы	Каталог HTML
		Имя файла PDF	
<i>OLAP Integration Server Metaoutline User's Guide</i>	Объясняет, как создавать и заполнять метамакеты OLAP при помощи стандартного интерфейса метамакетов OLAP (а не при помощи Metaoutline Assistant).	SC27-0784 db2urpx70	нет
<i>OLAP Integration Server Model User's Guide</i>	Объясняет, как создавать и заполнять метамакеты OLAP при помощи стандартного интерфейса моделей OLAP (а не при помощи Model Assistant).	SC27-0783 db2lpx70	нет
<i>Руководство по установке и использованию OLAP</i>	Содержит информацию о конфигурировании и установке для Начального комплекта OLAP.	SH43-0137 db2ipx70	db2ip
<i>Руководство пользователя надстройки электронных таблиц для Excel</i>	Описывает, как использовать программу электронных таблиц Excel для анализа данных OLAP.	SH43-0141 db2epx70	db2ep
<i>Руководство пользователя надстройки электронных таблиц для Lotus 1-2-3</i>	Описывает, как использовать программу электронных таблиц Lotus 1-2-3 для анализа данных OLAP.	SH43-0140 db2tpx70	db2tp
<i>Replication Guide and Reference</i>	Содержит информацию по планированию, конфигурированию, управлению и использованию инструментов IBM Replication, поставляемых с DB2.	SC26-9920 db2e0x70	db2e0
<i>Spatial Extender User's Guide and Reference</i>	Содержит информацию по установке, конфигурированию, управлению, программированию и устранению неисправностей для DB2 Spatial Extender. Кроме того, содержит содержательное описание понятий пространственных данных и справочную информацию (сообщения и SQL) по модулю Spatial Extender.	SC27-0701 db2sbx70	db2sb
<i>SQL Getting Started</i>	Введение в основные понятия SQL и примеры для многих конструкций и задач.	SC09-2973 db2y0x70	db2y0

Таблица 35. Информация DB2 (продолжение)

Название	Описание	Номер формы	Каталог HTML
		Имя файла PDF	
<i>SQL Reference, Том 1 и Том 2</i>	<p>Описывает синтаксис SQL, его семантику и правила языка. Эта книга включает также информацию о совместимости версий, ограничения продукта и обзор каталогов.</p> <p>Оба тома <i>SQL Reference</i> можно заказать на английском языке в Северной Америке, их номер формы - SBOF-8933.</p>	<p>Том 1 SC09-2974</p> <p>db2s1x70</p> <p>Том 2 SC09-2975</p> <p>db2s2x70</p>	db2s0
<i>System Monitor Guide and Reference</i>	Описывает сбор различной информации о базах данных и менеджере баз данных. Эта книга объясняет, как использовать информацию, чтобы понять работу с базой данных, улучшить производительность и найти причины ошибок.	<p>SC09-2956</p> <p>db2f0x70</p>	db2f0
<i>Text Extender Administration and Programming</i>	Содержит общую информацию о модулях расширения DB2, о конфигурировании модуля расширения для работы с текстом, об управлении им и о программировании с использованием модулей расширения для работы с текстом. Включает в себя справочную информацию, диагностическую информацию (с сообщениями) и примеры.	<p>SC26-9930</p> <p>desu9x70</p>	desu9
<i>Troubleshooting Guide</i>	Помогает определить причины ошибок, выполнить восстановительные операции, и использовать средства диагностики, консультируясь со Службой заказчиков DB2.	<p>GC09-2850</p> <p>db2p0x70</p>	db2p0
<i>Что нового</i>	Описывает новые возможности, функции и усовершенствования в DB2 Universal Database Версии 7.	<p>SH43-0131</p> <p>db2q0x70</p>	db2q0
<b>Информация по установке и конфигурированию DB2</b>			
<i>DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings</i>	Содержит информацию по планированию, установке и конфигурированию DB2 Connect Enterprise Edition в OS/2 и 32-битных системах Windows. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	<p>GC09-2953</p> <p>db2c6x70</p>	db2c6

Таблица 35. Информация DB2 (продолжение)

Название	Описание	Номер формы	Каталог HTML
		Имя файла PDF	
<i>DB2 Connect Enterprise Edition for UNIX Quick Beginnings</i>	Содержит информацию по планированию, установке, конфигурированию и выполнению заданий для DB2 Connect Enterprise Edition на платформах на основе UNIX. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2952 db2сух70	db2су
<i>DB2 Connect Personal Edition Быстрый старт</i>	Содержит информацию по планированию, установке, конфигурированию и выполнению заданий для DB2 Connect Personal Edition в OS/2 и 32-битных средах Windows. Эта книга содержит также информацию по установке и настройке для всех поддерживаемых клиентов.	GH43-0127 db2с1х70	db2с1
<i>DB2 Connect Personal Edition Quick Beginnings for Linux</i>	Содержит информацию по планированию, установке, перенастройке и конфигурированию DB2 Connect Personal Edition во всех поддерживаемых версиях Linux.	GC09-2962 db2с4х70	db2с4
<i>DB2 Data Links Manager Quick Beginnings</i>	Содержит информацию по планированию, установке и конфигурированию DB2 Data Links Manager в AIX и 32-битных операционных системах Windows.	GC09-2966 db2z6х70	db2z6
<i>DB2 Enterprise - Extended Edition for UNIX Quick Beginnings</i>	Содержит информацию по планированию, установке и конфигурированию DB2 Enterprise - Extended Edition на платформах на основе UNIX. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2964 db2v3х70	db2v3
<i>DB2 Enterprise - Extended Edition for Windows Quick Beginnings</i>	Содержит информацию по планированию, установке и конфигурированию DB2 Enterprise - Extended Edition в 32-битных системах Windows. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2963 db2v6х70	db2v6

Таблица 35. Информация DB2 (продолжение)

Название	Описание	Номер формы	Каталог HTML
		Имя файла PDF	
<i>DB2 for OS/2 Быстрый старт</i>	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database Personal Edition в операционной системе OS/2. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2968	db2i2
		db2i2x70	
<i>DB2 for UNIX Быстрый старт</i>	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database на платформах на основе UNIX. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2970	db2ix
		db2ixx70	
<i>DB2 for Windows Быстрый старт</i>	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database в 32-битных системах Windows. Эта книга содержит также информацию по установке и настройке для многих поддерживаемых клиентов.	GC09-2971	db2i6
		db2i6x70	
<i>DB2 Personal Edition Quick Beginnings</i>	Содержит информацию по планированию, установке, конфигурированию и использованию для DB2 Universal Database Personal Edition в OS/2 и в 32-битных системах Windows.	GC09-2969	db2i1
		db2i1x70	
<i>DB2 Personal Edition Quick Beginnings for Linux</i>	Содержит информацию по планированию, установке, перенастройке и конфигурированию DB2 Universal Database Personal Edition во всех поддерживаемых версиях Linux.	GC09-2972	db2i4
		db2i4x70	
<i>DB2 Query Patroller Installation Guide</i>	Содержит информацию по установке DB2 Query Patroller.	GC09-2959	db2iw
		db2iwx70	
<i>DB2 Warehouse Manager Installation Guide</i>	Содержит информацию по установке агентов хранилища, преобразователей хранилища и менеджера каталога данных.	GC26-9998	db2id
		db2idx70	
<b>Кроссплатформенные программы примеров в формате HTML</b>			

Таблица 35. Информация DB2 (продолжение)

Название	Описание	Номер формы	Каталог HTML
		Имя файла PDF	
Программы примеров в виде HTML	Содержит для справок программы примеров в виде HTML для языков программирования на всех платформах, поддерживаемых DB2. Эти программы примеров приводятся только в информационных целях. Не все из них доступны на всех языках программирования. Примеры HTML доступны, только если установлен клиент разработки программ DB2.  Дополнительную информацию об этих программах смотрите в книге <i>Application Building Guide</i> .	Номера формы нет	db2hs
Замечания по выпуску			
<i>Замечания по выпуску DB2</i>	Содержит самую свежую информацию, которую не успели включить в книги по DB2 Connect.	Смотрите примечание 2.	db2cr
<i>Замечания по установке DB2</i>	Содержит самую свежую информацию по установке, которую не успели включить в книги по DB2.	Доступна только на компакт-диске продукта.	
<i>Замечания по выпуску DB2</i>	Содержит самую свежую информацию о всех продуктах DB2 и их возможностях, которую не успели включить в книги по DB2.	Смотрите примечание 2.	db2ir

### Примечания:

- Символ *x* на шестой позиции в имени файла указывает язык книги. Например, имя файла *db2d0e70* говорит о том, что это английская версия книги *Administration Guide*, а имя файла *db2d0f70* соответствует французской версии этой же книги. Для обозначений языков используются на шестой позиции имени файла следующие буквы:

Язык	Обозначение
Английский	e
Болгарский	u
Бразильский португальский	b
Венгерский	h
Голландский	q
Греческий	a
Датский	d



Испанский	z
Итальянский	i
Корейский	k
Немецкий	g
Норвежский	n
Польский	p
Португальский	v
Русский	r
Словенский	l
Традиционный китайский	t
Турецкий	m
Упрощенный китайский	c
Финский	y
Французский	f
Чешский	x
Шведский	s
Японский	j

2. Последнюю информацию, которую не успели включить в книги по DB2, смотрите в Замечаниях по выпуску в формате HTML и в виде ASCII-файла. HTML-версию можно вызвать через Информационный центр или с компакт-диска продукта. Чтобы посмотреть ASCII-файл:
  - На платформах на базе UNIX смотрите файл `Release.Notes`. Он расположен в каталоге `DB2DIR/Readme/%L`, где `%L` - национальная версия, а DB2DIR:
    - `/usr/lpp/db2_07_01` в AIX
    - `/opt/IBMdb2/V7.1` в HP-UX, PTX, Solaris, и Silicon Graphics IRIX
    - `/usr/IBMdb2/V7.1` в Linux.
  - На других платформах смотрите файл `RELEASE.TXT`. Он находится в каталоге, где установлен продукт. На платформах OS/2 можно также дважды щелкнуть по папке **IBM DB2**, а затем дважды щелкнуть по значку **Release Notes**.

## Печать книг PDF

Если вы предпочитаете использовать печатные версии книг, можно напечатать файлы `.pdf` с компакт-диска публикаций по DB2. При помощи Adobe Acrobat Reader можно напечатать книгу целиком или же определенный диапазон страниц. Имена файлов для каждой книги в библиотеке приводятся в Табл. 35 на стр. 282.

Последнюю версию Adobe Acrobat Reader можно получить с Web-сайта фирмы Adobe, <http://www.adobe.com>.

Файлы PDF (расширения файлов - `.PDF`) входят в состав компакт-диска публикаций DB2. Для доступа к этим файлам:

1. Вставьте в устройство CD-ROM компакт-диск с публикациями DB2. На платформах на основе UNIX смонтируйте компакт-диск с публикациями DB2. Процедуру монтирования посмотрите в книге *Быстрый старт*.
2. Запустите Acrobat Reader.
3. Откройте требуемый файл PDF из одного из следующих мест:
  - На платформах OS/2 и Windows:  
Из каталога `x:\doc\язык`, где `x` - буква компакт-диска, а `язык` двухсимвольный код страны, соответствующий вашему языку (например, RU для русского).
  - На платформах на основе UNIX:  
Из каталога `/cdrom/doc/%L` на компакт-диске, где `/cdrom` - точка установки компакт-диска, а `%L` - имя требуемой национальной версии.

Можно также скопировать файлы PDF с компакт-диска на локальный или сетевой диск и читать их оттуда.

## **Заказ печатных копий**

Печатные копии книг DB2 можно заказать по отдельности или в комплекте (только в Северной Америке) по номеру SBOF. Чтобы заказать книги, обратитесь к вашему авторизованному дилеру или торговому представителю IBM, или позвоните по телефону 1-800-879-2755 в Соединенных Штатах или 1-800-IBM-4YOU в Канаде. Можно также заказать книги на Web-странице Publications по адресу <http://www.elink.ibm.link.ibm.com/pbl/pbl>.

Есть два комплекта книг. SBOF-8935 содержит справочную и пользовательскую информацию для DB2 Warehouse Manager. SBOF-8931 содержит справочную и пользовательскую информацию для всех остальных продуктов и возможностей DB2 Universal Database. Содержимое каждого комплекта SBOF приводится в следующей таблице:

Таблица 36. Заказ печатных книг

Номер SBOF	Содержит книги	
SBOF-8931	<ul style="list-style-type: none"> <li>• Руководство администратора: Планирование</li> <li>• Руководство администратора: Реализация</li> <li>• Руководство администратора: Производительность</li> <li>• Administrative API Reference</li> <li>• Application Building Guide</li> <li>• Application Development Guide</li> <li>• CLI Guide and Reference</li> <li>• Command Reference</li> <li>• Data Movement Utilities Guide and Reference</li> <li>• Data Warehouse Center Administration Guide</li> <li>• Data Warehouse Center Application Integration Guide</li> <li>• DB2 Connect User's Guide</li> <li>• Дополнение по установке и настройке</li> <li>• Image, Audio, and Video Extenders Administration and Programming</li> <li>• Справочник по сообщениям, Тома 1 и 2</li> </ul>	<ul style="list-style-type: none"> <li>• OLAP Integration Server Administration Guide</li> <li>• OLAP Integration Server Metaoutline User's Guide</li> <li>• OLAP Integration Server Model User's Guide</li> <li>• OLAP Integration Server User's Guide</li> <li>• Руководство по установке и использованию OLAP</li> <li>• Руководство пользователя надстройки электронных таблиц для Excel</li> <li>• Руководство пользователя надстройки электронных таблиц для Lotus 1-2-3</li> <li>• Replication Guide and Reference</li> <li>• Spatial Extender Administration and Programming Guide</li> <li>• SQL Getting Started</li> <li>• SQL Reference, Volumes 1 and 2</li> <li>• System Monitor Guide and Reference</li> <li>• Text Extender Administration and Programming</li> <li>• Troubleshooting Guide</li> <li>• Что нового</li> </ul>
SBOF-8935	<ul style="list-style-type: none"> <li>• Information Catalog Manager Administration Guide</li> <li>• Information Catalog Manager User's Guide</li> <li>• Information Catalog Manager Programming Guide and Reference</li> </ul>	<ul style="list-style-type: none"> <li>• Query Patroller Administration Guide</li> <li>• Query Patroller User's Guide</li> </ul>

## Электронная документация DB2

### Обращение к электронной справке

Для всех компонентов DB2 доступна электронная справка. Различные типы справки перечислены в следующей таблице.

Тип справки	Содержание	Как вызвать...
<i>Справка по командам</i>	Объясняет синтаксис команд процессора командной строки.	В процессоре командной строки в интерактивном режиме введите:  ? команда  где команда - ключевое слово для команды целиком.  Например, ? catalog выводит справку по всем командам CATALOG, а ? catalog database выводит справку по команде CATALOG DATABASE.
<i>Справка по Ассистенту конфигурирования клиента</i>	Объясняет задания, которые можно выполнить в окне или в записной книжке. Справка содержит обзор и предварительную информацию, которую надо знать, и описывает, как использовать управляющие элементы окна или записной книжки.	В окне или в записной книжке нажмите кнопку <b>Справка</b> или клавишу <b>F1</b> .
<i>Справка по Командному центру</i>		
<i>Справка по Центру управления</i>		
<i>Справка по Data Warehouse Center</i>		
<i>Справка по анализатору событий</i>		
<i>Справка по менеджеру каталога данных</i>		
<i>Справка по центру управления сателлитами</i>		
<i>Справка по центру сценариев</i>		

Тип справки	Содержание	Как вызвать...
Справка по сообщениям	Описывает для сообщения причину и действия, которые следует предпринять.	<p>В процессоре командной строки в интерактивном режиме введите:</p> <pre>? XXXnnnnn</pre> <p>где <i>XXXnnnnn</i> - идентификатор допустимого сообщения.</p> <p>Например, ? SQL30081 выводит справку по сообщению SQL30081.</p> <p>Чтобы смотреть справку по сообщению поэкранно, введите:</p> <pre>? XXXnnnnn   more</pre> <p>Чтобы записать справку по сообщению в файл, введите:</p> <pre>? XXXnnnnn &gt; имяфайла.рси</pre> <p>где <i>имяфайла.рси</i> - имя файла, где вы хотите сохранить справку.</p>
Справка по SQL	Объясняет синтаксис операторов SQL.	<p>В процессоре командной строки в интерактивном режиме введите:</p> <pre>help оператор</pre> <p>где <i>оператор</i> - оператор SQL.</p> <p>Например, help SELECT выводит справку по оператору SELECT.</p> <p><b>Примечание:</b> Справка по SQL недоступна на платформах на основе UNIX.</p>
Справка по SQLSTATE	Объясняет состояния SQL и коды классов.	<p>В процессоре командной строки в интерактивном режиме введите:</p> <pre>? sqlstate или ? код класса</pre> <p>где <i>sqlstate</i> - допустимый пятизначный код SQL, а <i>код класса</i> - первые две цифры sqlstate.</p> <p>Например, ? 08003 выводит справку по состоянию SQL 08003, а ? 08 выводит справку по коду класса 08.</p>

## Просмотр информации на экране

Книги, поставляемые с этим продуктом, записаны в формате HTML. Этот формат позволяет искать и просматривать информацию и поддерживает гипертекстовые ссылки. Он упрощает также совместное использование библиотеки на сайте.

Электронные книги и примеры программ можно просматривать в любом браузере, который поддерживает спецификации HTML Версии 3.2.

Чтобы просмотреть книги или примеры программ:

- Если вы работаете с инструментами администратора DB2, используйте Информационный центр.
- В браузере выберите **Файл** → **Открыть страницу**. На открытой странице приводятся описания и ссылки на информацию по DB2:

- На платформах на базе UNIX откройте страницу:

```
INSTHOME/sql1lib/doc/%L/html/index.htm
```

где %L - имя национальной версии.

- На других платформах откройте страницу:

```
sql1lib\doc\html\index.htm
```

на диске, где установлена DB2.

Если вы не установили Информационный центр, эту страницу можно открыть, щелкнув дважды по значку **Информация DB2**. В зависимости от того, в какой системе вы работаете, этот значок может находиться в основной папке продукта или в меню Windows Пуск.

### Установка браузера Netscape

Если у вас еще не установлен браузер Web, можно установить Netscape с компакт-диска Netscape, включенного в состав продукта. Чтобы получить подробные указания по установке, выполните следующие действия:

1. Вставьте в устройство CD-ROM компакт-диск Netscape.
2. На платформах на основе UNIX смонтируйте компакт-диск. Процедуру монтирования посмотрите в книге *Быстрый старт*.
3. Прочтите инструкции по установке в файле `CDNAVnn.txt`, где *nn* - двухсимвольный идентификатор языка. Этот файл находится в корневом каталоге компакт-диска.

### Доступ к информации через Информационный центр

Информационный центр обеспечивает быстрый доступ к информации о продуктах DB2. Информационный центр доступен на всех платформах, где есть инструменты администратора DB2.

Чтобы открыть Информационный центр, щелкните дважды по значку Информационный центр. В зависимости от того, в какой системе вы работаете, этот значок может находиться в основной папке продукта или в меню **Пуск**.

На платформах Windows можно также вызвать Информационный центр через панель задач и через меню **Справка DB2**.

Информационный центр дает шесть типов информации. Для обращения к информации одного из этих типов выберите соответствующую закладку.

**Задания** Основные задания, которые вы можете выполнить в DB2.

**Справочник** Справочная информация по таким элементам DB2, как ключевые слова, команды и API.

**Книги** Книги DB2.

#### **Устранение неисправностей**

Список сообщений об ошибках и рекомендуемых действий по категориям.

#### **Программы примеров**

Программы примеров, поставляемые с клиентом разработки программ DB2. Если вы не установили клиент разработки программ DB2, эта закладка не выводится.

**Web** Информация по DB2 в WWW. Чтобы посмотреть эту информацию, ваша система должна быть подключена к Web.

Когда вы выбираете пункт в одном из списков, информационный центр запускает программу просмотра для вывода информации. Этой программой может быть программа просмотра системной справки, редактор или браузер Web в зависимости от того, какую информацию вы выбрали.

Информационный центр поддерживает возможность поиска, и вы можете искать определенную тему, не просматривая книги целиком.

Для полнотекстового поиска выберите гипертекстовую ссылку в Информационном центре и откройте поисковую форму **Поиск электронной информации DB2**.

Обычно сервер поиска HTML запускается автоматически. Если поиск информации HTML не работает, вам, возможно, надо запустить сервер поиска одним из следующих способов:

#### **В Windows**

Выберите **Пуск**, затем **Программы** → **IBM DB2** → **Информация** → **Запустить сервер поиска HTML**.

**В OS/2** Щелкните дважды по папке **DB2 for OS/2**, а затем щелкните дважды по значку **Запустить сервер поиска HTML**.

Если у вас есть проблемы с использованием поиска информации HTML, посмотрите замечания по выпуску.

**Примечание:** Функция поиска недоступна в средах Linux, PTX и Silicon Graphics IRIX.

## Использование мастеров DB2

Мастера помогают вам выполнять конкретные задачи управления, ведя последовательно по шагам необходимых действий. Мастера доступны в Центре управления и в Ассистенте конфигурирования клиента. Список мастеров с соответствующими задачами приведен в следующей таблице.

**Примечание:** Мастера по созданию баз данных, индексов, конфигурированию многоузлового изменения и производительности доступны в среде многораздельных баз данных.

<b>Мастер</b>	<b>Помогает вам...</b>	<b>Как вызвать...</b>
<i>по добавлению баз данных</i>	Каталогизировать базу данных на клиентской рабочей станции	В Ассистенте конфигурирования клиента нажмите кнопку <b>Добавить</b> .
<i>по резервному копированию базы данных</i>	Создать, определить и заполнить план резервного копирования.	В Центре управления щелкните правой кнопкой мыши по базе данных, для которой вам нужна резервная копия, и выберите <b>Резервное копирование</b> → <b>Базы данных при помощи мастера</b> .
<i>по конфигурированию многоузлового изменения</i>	Конфигурировать многоузловые изменения, распределенные транзакции или двухфазное принятие.	В Центре управления щелкните правой кнопкой мыши по папке <b>Базы данных</b> и выберите <b>Многоузловое изменение</b> .
<i>по созданию баз данных</i>	Создать базу данных и выполнить основные задачи конфигурирования.	В Центре управления щелкните правой кнопкой мыши по папке <b>Базы данных</b> и выберите <b>Создать</b> → <b>Базу данных при помощи мастера</b> .
<i>по созданию таблиц</i>	Выбрать типы основных данных и создать первичные ключи для таблицы.	В Центре управления щелкните правой кнопкой мыши по значку <b>Таблицы</b> и выберите <b>Создать</b> → <b>Таблицу при помощи мастера</b> .
<i>по созданию табличных пространств</i>	Создать новое табличное пространство.	В Центре управления щелкните правой кнопкой мыши по значку <b>Табличные пространства</b> и выберите <b>Создать</b> → <b>Табличное пространство при помощи мастера</b> .
<i>Создать индекс</i>	Выбрать, какие индексы создать или отбросить для всех ваших запросов.	В Центре управления щелкните правой кнопкой мыши по значку <b>Индекс</b> и выберите <b>Создать</b> → <b>Индекс при помощи мастера</b> .



<b>Мастер</b>	<b>Помогает вам...</b>	<b>Как вызвать...</b>
<i>по настройке производительности</i>	Настроить производительность базы данных, изменив параметры конфигурации в соответствии с вашими требованиями.	<p>В Центре управления щелкните правой кнопкой мыши по базе данных, которую вы хотите настроить, и выберите <b>Конфигурировать производительность при помощи мастера</b>.</p> <p>Для многораздельной среды баз данных в окне Разделы баз данных щелкните правой кнопкой мыши по первому разделу баз данных, который вы хотите настроить, и выберите <b>Конфигурировать производительность при помощи мастера</b>.</p>
<i>по восстановлению баз данных</i>	Восстановить базу данных после сбоя. Он поможет понять, какую резервную копию использовать, и какие журналы использовать при повторе.	<p>В Центре управления щелкните правой кнопкой мыши по базе данных, которую вы хотите восстановить, и выберите <b>Восстановить</b> → <b>Базу данных при помощи мастера</b>.</p>

## Установка сервера документации

По умолчанию информация по DB2 устанавливается в вашей локальной системе. Это значит, что каждый, кому требуется доступ к информации по DB2, должен устанавливать одни и те же файлы. Чтобы держать информацию по DB2 в едином месте, выполните следующие действия:

1. Скопируйте все файлы и подкаталоги каталога `\sql1lib\doc\html` вашей локальной системы на сервер Web. Каждая книга находится в своем собственном подкаталоге, где записаны все необходимые для нее файлы HTML и GIF. Структура подкаталогов должна остаться без изменений.
2. Сконфигурируйте сервер Web на поиск файлов на новом месте. Дополнительную информацию смотрите в приложении NetQuestion руководства *Дополнение по установке и настройке*.
3. Если вы используете Java-версию Информационного центра, можно задать базовый URL для всех файлов HTML. Этот URL надо использовать для списка книг.
4. Когда вы сможете просматривать файлы книг, можно пометить закладками часто используемые темы. Вероятно, вы захотите пометить закладками следующие страницы:
  - Список книг
  - Содержания часто используемых книг

- Часто требуемые статьи, например, тему ALTER TABLE
- Форму поиска

Информацию о том, как работать с файлами электронной документации на центральном компьютере, смотрите в приложении NetQuestion руководства *Дополнение по установке и настройке*.

## Поиск электронной информации

Для поиска информации в файлах HTML используйте один из следующих способов:

- Нажмите кнопку **Поиск** в верхнем фрейме. При помощи формы поиска найдите нужную тему. Эта функция недоступна в средах Linux, PTX и Silicon Graphics IRIX.
- Нажмите кнопку **Индекс** в верхнем фрейме. При помощи индекса найдите в книге нужную тему.
- Выведите содержание или индекс справки или книги HTML, затем при помощи функции поиска браузера Web найдите в книге нужную тему.
- При помощи функции закладок браузера Web можно быстро вернуться к определенной теме.
- Используйте для поиска определенных тем функцию поиска информационного центра. Подробности смотрите в разделе “Доступ к информации через Информационный центр” на стр. 296.

---

## Приложение F. Замечания

IBM может предлагать описанные продукты, услуги и возможности не во всех странах. Сведения о продуктах и услугах, доступных в настоящее время в вашей стране, можно получить в местном представительстве IBM. Любые ссылки на продукты, программы или услуги IBM не означают явным или неявным образом, что можно использовать только продукты, программы или услуги IBM. Разрешается использовать любые функционально эквивалентные продукты, программы или услуги, если при этом не нарушаются права IBM на интеллектуальную собственность. Однако ответственность за оценку и проверку работы любых продуктов, программ и услуг других фирм лежит на пользователе.

Фирма IBM может располагать патентами или рассматриваемыми заявками на патенты, относящимися к предмету данного документа. Получение этого документа не означает предоставления каких-либо лицензий на эти патенты. Запросы по поводу лицензий следует направлять в письменной форме по адресу:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

По поводу лицензий, связанных с использованием наборов двухбайтных символов (DBCS), обращайтесь в отдел интеллектуальной собственности IBM в вашей стране или направьте запрос в письменной форме по адресу:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106, Japan

**Следующий абзац не применяется в Великобритании или в любой другой стране, где подобные заявления противоречат местным законам: КОРПОРАЦИЯ INTERNATIONAL BUSINESS MACHINES ПРЕДСТАВЛЯЕТ ДАННУЮ ПУБЛИКАЦИЮ “КАК ЕСТЬ” БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ПРЕДПОЛАГАЕМЫЕ ГАРАНТИИ СОВМЕСТИМОСТИ, РЫНОЧНОЙ ПРИГОДНОСТИ И СООТВЕТСТВИЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. В некоторых странах для определенных сделок подобные оговорки не допускаются, таким образом, это утверждение может не относиться к вам.**

Данная информация может содержать технические неточности и типографские опечатки. Периодически в информацию вносятся изменения, они будут включены в новые издания этой публикации. Фирма IBM может в любое время без уведомления вносить изменения и усовершенствования в продукты и программы, описанные в этой публикации.

Любые ссылки в данной информации на Web-сайты, не принадлежащие IBM, приводятся только для удобства и никоим образом не означают поддержки IBM этих Web-сайтов. Материалы этих Web-сайтов не являются частью данного продукта IBM и вы можете использовать их только на собственную ответственность.

IBM может использовать или распространять присланную вами информацию любым способом, как фирма сочтет нужным, без каких-либо обязательств перед вами.

Если обладателю лицензии на данную программу понадобятся сведения о возможности: (i) обмена данными между независимо разработанными программами и другими программами (включая данную) и (ii) совместного использования таких данных, он может обратиться по адресу:

IBM Canada Limited  
Office of the Lab Director  
1150 Eglinton Ave. East  
North York, Ontario  
M3C 1H7  
CANADA

Такая информация может быть предоставлена на определенных условиях (в некоторых случаях к таким условиям может относиться оплата).

Лицензированная программа, описанная в данной публикации, и все лицензированные материалы, доступные с ней, предоставляются IBM на условиях IBM Customer Agreement (Соглашения IBM с заказчиком), Международного соглашения о лицензиях на программы IBM или эквивалентного соглашения.

Приведенные данные о производительности измерены в контролируемой среде. Таким образом, результаты, полученные в других операционных средах, могут существенно отличаться от них. Некоторые показатели измерены в системах разработки и нет никаких гарантий, что в общедоступных системах эти показатели будут теми же. Более того, некоторые результаты могут быть получены путем экстраполяции. Реальные результаты могут отличаться от них. Пользователи должны проверить данные для своих конкретных сред.

Информация о продуктах других фирм получена от поставщиков этих продуктов, из их опубликованных объявлений или из других общедоступных

источников. Фирма IBM не проверяла эти продукты и не может подтвердить точность измерений, совместимость или прочие утверждения о продуктах других фирм. Вопросы о возможностях продуктов других фирм следует направлять поставщикам этих продуктов.

Все утверждения о будущих планах и намерениях IBM могут быть изменены или отменены без уведомлений, и описывают исключительно цели фирмы.

Эта информация может содержать примеры данных и отчетов, иллюстрирующие типичные деловые операции. Чтобы эти примеры были правдоподобны, в них включены имена лиц, названия компаний и товаров. Все эти имена и названия вымышлены и любое их сходство с реальными именами и адресами полностью случайно.

#### ЛИЦЕНЗИЯ НА КОПИРОВАНИЕ:

Эта информация может содержать примеры прикладных программ на языках программирования, иллюстрирующих приемы программирования для различных операционных платформ. Разрешается копировать, изменять и распространять эти примеры программ в любой форме без оплаты фирме IBM для целей разработки, использования, сбыта или распространения прикладных программ, соответствующих интерфейсу прикладного программирования операционных платформ, для которых эти примера программ написаны. Эти примеры не были всесторонне проверены во всех возможных условиях. Поэтому IBM не может гарантировать их надежность, пригодность и функционирование.

Каждая копия программ примеров или программ, созданных на их основе, должна содержать следующее замечание об авторских правах:

© (название вашей фирмы) (год). Части этого кода построены на основе примеров программ IBM Corp. © Copyright IBM Corp. \_введите год или годы\_. Все права защищены.

---

## Товарные знаки

Следующие термины (они могут быть помечены звездочкой - \*) являются товарными знаками корпорации International Business Machines в Соединенных Штатах и/или в других странах:

ACF/VTAM	IBM
AISPO	IMS
AIX	IMS/ESA
AIX/6000	LAN DistanceMVS
AIXwindows	MVS/ESA
AnyNet	MVS/XA
APPN	Net.Data
AS/400	OS/2
BookManager	OS/390
CICS	OS/400
C Set++	PowerPC
C/370	QBIC
DATABASE 2	QMF
DataHub	RACF
DataJoiner	RISC System/6000
DataPropagator	RS/6000
DataRefresher	S/370
DB2	SP
DB2 Connect	SQL/DS
DB2 Extenders	SQL/400
Сервер OLAP DB2	System/370
DB2 Universal Database	System/390
Distributed Relational Database Architecture	SystemView
DRDA	SystemView
eNetwork	VM/ESA
Extended Services	VSE/ESA
FFST	VTAM
First Failure Support Technology	WebExplorer
	WIN-OS/2

Следующие термины являются товарными знаками или зарегистрированными товарными знаками других компаний:

Microsoft, Windows и Windows NT - товарные знаки или зарегистрированные товарные знаки Microsoft Corporation.

Java, все товарные знаки и логотипы на основе Java и Solaris - товарные знаки Sun Microsystems, Inc. в Соединенных Штатах и/или в других странах.

Tivoli и NetView - товарные знаки Tivoli Systems Inc. в Соединенных Штатах и/или других странах.

UNIX - зарегистрированный товарный знак в Соединенных Штатах и в других странах, его использование лицензируется исключительно фирмой X/Open Company Limited.

Названия других компаний, продуктов и услуг (они могут быть отмечены двойной звездочкой - \*\*) могут быть товарными знаками или марками сервиса других фирм.





# Индекс

## D

DB2 Connect

использование для многоузловых  
изменений баз данных 140

DMS (пространство, управляемое  
базой данных) 12, 119

DTP (распределенная обработка  
транзакций) 156

## G

GIS (географическая информационная  
система) 57

## H

HTML

программы примеров 289

## I

IVMCATGROUP 113

IBMDEFAULTGROUP 113

IBMTMPGROUP 114

## M

Microsoft Transaction Server

включение поддержки в DB2 181

многократное использование  
соединений ODBC 184

настройка связи TCP/IP 185

объединение в пул соединений при  
помощи ADO 2.1 и более  
новых 184

поддерживаемые серверы баз  
данных DB2 182

предварительные требования для  
программного обеспечения 181

проверка установки 182

пул соединений 183

срок ожидания транзакции и  
поведение соединения DB2 182

тестирование DB2 программой  
примера 185

установка и

конфигурирование 181

## P

PDF 291

## S

SMS (пространство, управляемое  
системой) 12, 115

SNA (Systems Network  
Architecture) 146

SPM (менеджер точек  
синхронизации) 143

SYSCATSPACE 113

Systems Network Architecture  
(SNA) 146

## T

TEMPSPACE1 114

## U

UDF (пользовательские функции)  
обзор 71

USERSPACE1 113

## W

Windows

переменная реестра  
DB2CODEPAGE 253

поддерживаемые кодовые  
страницы 253

## A

авторегферентность

ограничение 85

строка 85

таблица 85

авторизация 27

обзор 89

обзор базы данных

объединения 28

агент

хранилище 52

аппаратные среды 41

логические разделы базы  
данных 47

несколько многопроцессорных  
разделов 46

несколько однопроцессорных  
разделов 45

один раздел, несколько  
процессоров 43

один раздел, один процессор 42

типы параллелизма 49

атрибут 65

аудит действий

обзор 89

аутентификация 26

аутентификация (*продолжение*)

обзор базы данных

объединения 28

## Б

база данных

использование в транзакции  
нескольких баз данных 139

каталоги 91

на системе хоста 140

обзор 9

распределенная 137

файлы 92

база данных объединения

авторизация 28

аутентификация 28

особенности проектирования 135

последовательности слияния 261

регистрозависимые имена 200

системы 31

библиотека DB2

заказ печатных копий 292

идентификаторы языков для  
книг 290

Информационный центр 296

книги 281

мастера 298

печать книг PDF 291

поиск электронной  
информации 300

последняя информация 291

просмотр информации на  
экране 295

структура 281

установка сервера

документации 299

электронная справка 293

большие объекты

определение столбца 70

большие объекты (LOB)

определение столбца 70

браузер Netscape

установка 296

## В

внешний ключ

определение 84

внешний параллелизм 38

внутренний параллелизм 38

внутрираздельный параллелизм 38

- внутрираздельный параллелизм  
(*продолжение*)  
используемый с межраздельным параллелизмом 40
- возможность параллельной обработки  
обзор 90
- восстановление 24  
версия 24  
повтор транзакций 24  
после отказа 24
- восстановление версии 24
- восстановление после сбоев 24
- восстановление с повтором транзакций 24
- временные рабочие пространства  
оценка требований размера 102
- временные табличные пространства 114  
рекомендации 128
- время  
определение 263  
форматы 266
- вторая нормальная форма 77
- вхождение объекта 66
- выбор ключевых столбцов 73
- выбор размера экстенда 127
- высокая доступность 187
- Г**  
географическая информационная система (GIS) 57
- геокодирование 60
- группы узлов 36  
IBM`CATGROUP` 113  
IBM`DEFAULTGROUP` 113  
IBM`TEMPGROUP` 114  
обзор 9  
проектирование 102
- Д**  
данные  
большой объект 97  
длинное поле 97  
информационные 51  
рабочие 51  
разделение 104
- данные LOB (большой объект)  
оценка требований размера 97
- данные атрибутов 58
- данные типа большой объект  
оценка требований размера 97
- данные типа длинное поле  
оценка требований размера 97
- дата  
определение 263  
форматы 266
- двухфазное принятие 139, 140  
обработка ошибок 151  
процесс 148
- деловые метаданные 55
- Е**  
единица работы 137  
единицы работы  
удаленные 138
- емкость 41
- Ж**  
журнальная файловая система 187
- З**  
зависимая строка 85  
зависимая таблица 84  
задачи  
хранилища данных 55
- замечания по выпуску 291
- защита 26  
учет при проектировании 90
- значения TP\_MON\_NAME 162  
значения TPM 162  
значения даты и времени  
обзор 263  
строчные представления 264
- И**  
идентификатор языка  
книги 290
- иерархия типов 70, 90
- индекс  
обзор 10
- индексное пространство  
оценка требований размера 98
- индексный ключ 10
- инструмент db2inidb 190
- интерфейс менеджеров транзакций X/Open (XA)  
распределенная обработка транзакций (DTP) 156
- информационные данные 51
- Информационный центр 296
- источник данных 31
- К**  
карта  
разделения 105
- каталог  
база данных 91  
каталог данных 55
- каталоги национальных версий платформы на основе UNIX 249
- клон базы данных  
создание 191
- ключи  
обзор 72  
разделения 107  
ключи уникальности  
обзор 72
- книги 281, 292
- кодовые страницы  
переменная реестра DB2`CODEPAGE` 253  
поддерживаемые кодовые страницы Windows 253
- команда LIST INDOUBT TRANSACTIONS 167
- контейнер  
обзор 16
- контейнеры  
добавление в табличные пространства DMS 121
- конфигурация  
многораздельная 45
- корневые типы 70
- Л**  
логические правила  
обзор 20
- логические разделы базы данных 47
- логическое проектирование базы данных  
выбор данных для записи в базу 65  
определение таблиц 67  
отношения 67
- М**  
мастер  
восстановление баз данных 299
- мастер по восстановлению 299
- мастер по добавлению баз данных 298, 299
- мастер по индексам 298
- мастер по конфигурированию многоузлового изменения 298
- мастер по настройке производительности 298
- мастер по резервному копированию баз данных 298
- мастер по созданию баз данных 298
- мастер по созданию таблиц 298
- мастер по созданию табличных пространств 298

- мастера
  - выполнение заданий 298
  - добавление баз данных 298, 299
  - индекс 298
  - конфигурирование многоузлового изменения 298
  - настройка
    - производительности 298
  - резервное копирование баз данных 298
  - создание базы данных 298
  - создать таблицу 298
  - создать табличное пространство 298
- масштабируемость 41
- межраздельный параллелизм 39
  - используемый с
    - внутрираздельным параллелизмом 40
- менеджер ресурсов
  - задание базы данных в качестве 160
- менеджер точек синхронизации (SPM) 143
- менеджер точек синхронизации (SPM) DB2 146
- менеджер транзакций 140, 142
  - BEA Tuxedo 178
  - IBM TXSeries CICS 175
  - IBM TXSeries Encina 176
    - конфигурирование DB2 176
    - конфигурирование Encina для каждого менеджера ресурсов 176
    - обращение к базам данных DB2 из программ Encina 177
  - Microsoft Transaction Server 180
- менеджеры транзакций XA
  - конфигурирование 175
- метаданные 55
- многораздельная база данных 35
- многораздельная группа узлов 36
- многораздельные конфигурации 45
- многоузловые изменения 139, 140
  - прикладные программы хоста или AS/400, обращающиеся к серверу DB2 UDB 146
- модуль Spatial Extender
  - обзор 57

## **H**

- надтипы 70
- назначение
  - производные таблицы 71

- назначение (*продолжение*)
  - строки 71
  - таблицы 71
- назначения
  - типы 70
- национальные версии
  - совместимость между сервером администратора и экземпляром 249
- неоднозначные транзакции
  - восстановление 152, 158
  - восстановление вручную 167
  - ресинхронизация 153
- несколько многопроцессорных разделов 46
- несколько однопроцессорных разделов 45
- несовместимости
  - COLNAMES (планируемая) 211
  - FK\_COLNAMES (планируемая) 210
  - PK\_COLNAMES (планируемая) 210
  - Версия 6 216
  - Версия 7 211
  - имена столбцов внешних ключей 217
  - имена столбцов первичных ключей 217
  - несоответствие столбцов 220
  - описание 209
  - планируемые 210
  - производные таблицы только для чтения (планируемые) 210
  - создание баз данных 229
  - столбец COLNAMES таблицы SYSCAT.INDEXES 219
  - столбец TEXT таблицы SYSCAT.CHECKS 219
  - столбец TEXT таблицы SYSCAT.STATEMENTS 218
  - столбец TEXT таблицы SYSCAT.VIEWS 218
  - тип данных столбцов изменен на BIGINT 219
- несовместимости для Версии 6
  - CURRENT EXPLAIN MODE 230
  - RUMBA 231
  - SQLNAME в недублированной SQLVAR 226
  - USING и SORT BUFFER 231
  - базовые каталоги SYSIBM 222
  - изменение столбцов SYSTABLE 229
  - изменения формата PC/IXF 225

- несовместимости для Версии 6 (*продолжение*)
  - коды зависимостей 221
  - привилегия SELECT для иерархии 229
  - программирование на языке Java 223
  - производные таблицы OBJCAT 221
  - размеры символьных имен 224
  - сигнатуры строчных функций SYSFUN 228
  - синтаксис FOR UPDATE 224
  - столбцы DATALINK 227
  - тип данных VARCHAR 223
  - устаревшие ключевые слова конфигурации 226
  - устаревшие параметры конфигурации базы данных 231
  - формат выходного потока монитора событий 227
- несовместимость выпусков
  - описание 209
- нормализация таблиц 76

## **O**

- обзор хранилищ данных 51
- область видимости
  - ссылочный тип 71
- обработка транзакций
  - конфигурирование менеджеров транзакций XA 175
- объекты
  - в базе данных 65
- объекты базы данных
  - база данных 9
  - группа узлов 9
  - индекс 10
  - обзор 7
  - правила именования 255
  - производная таблица 9
  - схема 11
  - таблица 9
  - таблица системного каталога 12
  - экземпляр 8
- объекты мультимедиа 66
- объекты хранения
  - контейнер 16
  - обзор 12
  - пул буферов 17
  - табличное пространство 12
- ограничение
  - NOT NULL 21
  - внешний ключ 22

- ограничение (*продолжение*)
    - первичный ключ 21
    - проверка 23
    - уникальности 21
  - ограничение NOT NULL 21
  - ограничение внешнего ключа 22
  - ограничение первичного ключа 21, 84
  - ограничение уникальности 21
  - ограничения
    - обзор 82
  - ограничения уникальности
    - обзор 82
  - один раздел
    - многопроцессорная среда 43
    - однопроцессорная среда 42
  - однопроцессорная среда 42
  - определение столбцов таблицы 69
  - оптимизатор SQL 10
  - особенности ввода/вывода
    - табличное пространство 122
  - отделение зеркальной копии 190
  - отделенная зеркальная копия
    - как резервная база данных 191
    - как резервная копия 192
  - отметка времени
    - определение 264
  - отношения
    - многие-с-одним 67
    - многие-со-многими 68
    - один-с-многими 67
    - один-с-одним 69
  - отображение
    - таблиц в табличные пространства 125
    - табличных пространств на группы узлов 125
    - табличных пространств на пулы буферов 124
  - оценка требований размера
    - данные типа большой объект 97
    - данные типа длинное поле 97
    - индексное пространство 98
    - пространство для файлов журнала 101
- П**
- параллелизм
    - ввод/вывод 38
    - внутрираздельный 38
    - запрос 38
    - и различные аппаратные среды 49
    - и создание индексов 41
    - межраздельный 39
  - параллелизм (*продолжение*)
    - обзор 35
    - типы 37
    - утилиты 41
    - утилиты автозагрузки 41
    - утилиты загрузки 41
    - утилиты резервного копирования и восстановления базы данных 41
  - параллелизм ввода/вывода 38
  - параллелизм запросов 38
  - параллелизм утилит 41
  - параметр конфигурации базы данных
    - обзор 19
  - параметр конфигурации менеджера баз данных
    - обзор 18
  - параметры конфигурации менеджера транзакций DB2 145
    - обзор 18
  - первая нормальная форма 77
  - первичный индекс 72
  - первичный ключ
    - генерация уникальных значений 74
    - обзор 72
    - определение 84
  - переменная реестра DB2CODEPAGE 253
  - перенастройка
    - база данных 203
  - перенастройка базы данных 203
  - печать книг PDF 291
  - платформы на основе UNIX
    - каталоги национальных версий 249
  - поддерживаемые
    - серверы баз данных DB2 для транзакций, координируемых MTS 182
  - поддержка CCSID с двумя направлениями письма
    - реализация в DB2 Connect 259
    - реализация в DB2 UDB 258
    - таблица CCSID 256
  - поддержка CCSID, для языков с двумя направлениями письма
    - реализация в DB2 Connect 259
    - реализация в DB2 UDB 258
    - таблица CCSID 256
  - поддержка восстановления при отказах 187
    - взаимная подмена 189
    - горячее резервирование 189
  - поддержка национальных языков (NLS)
    - значения даты и времени 263
    - наборы символов 254
    - поддержка CCSID с двумя направлениями письма 256
  - подтипы 70
  - поиск
    - электронная информация 297, 300
  - пользовательская таблица
    - ограничения страниц 95
  - пользовательские временные табличные пространства 114
  - пользовательские типы (UDT)
    - определение столбца 70
  - пользовательские функции
    - обзор 71
  - пользовательское табличное пространство 113
  - понятия реляционных баз данных
    - обзор 7
  - порядок "первый подходящий" 95
  - последняя информация 291
  - последовательности сортировки 260
  - последовательность слияния
    - общие соображения 261
    - опция collating\_sequence 262
  - особенности баз данных
    - объединения 261
    - символы тайского языка 263
  - правила вставки 86
  - правила изменения 87
  - правила именования
    - общие 195
  - правила удаления 86
  - привилегия 28
  - принятие
    - двухфазное 148
    - ошибки при двухфазном 151
  - приостановка ввода-вывода
    - поддержка постоянной доступности 190
  - проверочное ограничение 23
  - программы примеров
    - HTML 289
    - кроссплатформенные 289
  - проектирование
    - база данных объединения 135
    - табличное пространство 122
  - проектирование базы данных
    - логическое 65
    - физическое 91
  - проектирование и выбор табличных пространств 111

производная таблица  
    обзор 9  
просмотр  
    электронная информация 295  
пространственная  
    информация 57  
пространственные  
    данные 59  
пространство для файлов журнала  
    оценка требований размера 101  
пространство, управляемое базой  
    данных (DMS) 12, 119  
пространство, управляемое системой  
    (SMS) 12, 115  
процесс (в хранилище) 53  
пул буферов  
    обзор 17  
пулы буферов  
    IBMDEFAULTBP 124  
пулы соединений, MTS 183  
пустое значение 72

## **P**

рабочие данные 51  
раздел  
    база данных 35  
раздел базы данных 35  
разделение  
    данные 104  
    карта 105  
    ключ 107  
размер экстенда 16  
    выбор 127  
    определение 112  
разработка программы  
    последовательность сортировки,  
    указания 261  
распределенная база данных 137  
распределенная обработка  
    транзакций  
        менеджер ресурсов 159  
        менеджер транзакций 158  
        обработка ошибок 167  
        оператор RELEASE 167  
        особенности защиты 170  
        особенности конфигурации 171  
        особенности соединения с базой  
        данных 166  
        поддержка серверов баз данных  
        хоста или AS/400 166  
        прикладная программа 156  
распределенное требование 31  
рассеяние  
    частичное 105

регистрозависимые имена, база  
    данных объединения 200  
резервное копирование 24  
реляционные связи  
    обзор 83  
    отношения, связанные по  
    удалению 87  
    последствия для операций  
    SQL 85  
    правила SQL DELETE 86  
    правила SQL INSERT 86  
    правила SQL UPDATE 87  
реляционные типы  
    обзор 90  
реляционный цикл  
    определение 85  
реорганизация таблиц 25  
реплицируемые сводные  
    таблицы 110  
родительская  
    строка 84  
    таблица 84  
родительский  
    ключ 83, 84

## **C**

сводные таблицы  
    обзор 89  
    реплицируемые 110  
символы тайского языка  
    сортировка 263  
система баз данных  
    объединение 31  
система координат 59  
системные временные табличные  
    пространства 114  
системные объекты  
    обзор 18  
    параметр конфигурации 18  
системный журнал  
    пример интерфейса XA 175  
Советчик  
    мастера 298  
совместимость  
    раздел 109  
совместимость с разделами 109  
совместное размещение  
    таблица 109  
составной ключ 73, 84  
способ объединения 75  
среда MPP 45  
среда SMP 43  
среда кластера SMP 46  
ссылочный тип  
    обзор 70

столбцы  
    определение для таблицы 69  
столбцы идентификации  
    обзор 74  
столбцы ключа  
    идентификация 73  
строка ха\_ореп 160  
строки времени  
    определение 265  
строки даты  
    определение 264  
строки отметки времени  
    определение 266  
структурированный тип 70, 90  
схема  
    обзор 11  
схема типа "звезда" 55

## **T**

таблица  
    обзор 9  
    реорганизация 25  
таблица системного каталога  
    обзор 12  
таблица-потомок 85  
таблицы  
    нормализация 76  
    отображение в табличные  
    пространства 125  
    оценка требований размера 93  
    пользователь 95  
    проверочные ограничения 88  
    системный каталог 94  
    совместное размещение 109  
таблицы системного каталога  
    оценка начального размера 94  
табличное пространство  
    обзор 12  
табличное пространство  
    каталога 113  
табличные пространства  
    SYSCATSPACE 113  
    TEMPSPACE1 114  
    USERSPACE1 113  
    временные 114, 128  
    выбор 111  
    выбор между SMS и DMS 132  
    каталог 113, 130  
    особенности ввода/вывода 122  
    особенности нагрузки 130  
    отображение на группы  
    узлов 125  
    отображение на пулы  
    буферов 124  
    пользователь 113

- табличные пространства
    - (*продолжение*)
    - проектирование 111, 122
    - пространство, управляемое базой данных (DMS) 119
    - пространство, управляемое системой (SMS) 115
  - табличные пространства DMS
    - добавление контейнеров 121
  - табличные пространства каталога
    - рекомендации 130
  - тематическая область 52
  - типизированные производные таблицы
    - обзор 70
  - типизированные таблицы 90
    - обзор 70
  - типы данных 90
  - транзакции
    - без XA 157
    - глобальная 157
    - двухфазное принятие 157
    - свободно связанная 157
    - тесно связанная 158
  - транзакции, координируемые MTS
    - поддерживаемые серверы баз данных DB2 182
  - транзакция
    - обращение к многораздельным базам данных 167
  - требования размера, оценка
    - временные рабочие пространства 102
    - таблицы 93
  - третья нормальная форма 79
  - триггер 23
  - триггеры
    - обзор 88
- У**
- удаленная единица работы 138
  - узел агента 53
  - узел агента по умолчанию 53
  - узел координатора 35
  - узлы
    - местонахождение данных, определение 105
  - уникальность
    - индекс 10
    - ключ 83
  - уровень полномочий 28
  - установка
    - браузер Netscape 296
  - установка сервера
    - документации 299
  - устройства RAID
    - оптимизация
      - производительности 133
- Ф**
- файловая система
    - журнальная 187
  - файлы баз данных 92
  - физическая структура базы данных 91
  - физические файлы
    - SMS 118
  - физические файлы SMS 118
- Х**
- хранилища данных
    - задачи 55
    - обзор 51
    - объекты 52
  - хранилище 51
    - агент 52
    - источник 52
    - потребитель 52
    - процесс 53
  - хронологические данные
    - обзор 89
- Ц**
- Центр управления
    - запуск переведенных версий 252
  - Центр хранилищ данных 51
- Ч**
- частичное рассеяние 105
  - четвертая нормальная форма 81
- Ш**
- шаг (в хранилище) 53
  - шаг SQL 54
  - шаг пользовательской программы 55
  - шаг преобразователя 54
  - шаг программы 54
  - шаг хранилища
    - SQL 54
    - пользовательская программа 55
    - преобразователь 54
    - программа 54
- Э**
- эвристические операции 169
  - экземпляр
    - обзор 8
  - электронная информация
    - поиск 300
    - просмотр 295

---

## Как связаться с IBM

Если у вас имеется техническая проблема, пожалуйста, перед обращением к службе поддержки пользователей DB2 просмотрите еще раз и выполните действия, рекомендуемые в руководстве *Troubleshooting Guide*. В этом руководстве описано, какую информацию надо собрать, чтобы служба поддержки пользователей DB2 могла лучше помочь вам.

Чтобы получить информацию или заказать любой из продуктов DB2 Universal Database, обратитесь к представителю IBM в местном отделении или к авторизованному продавцу программных продуктов IBM.

Если вы находитесь в США, позвоните по одному из следующих номеров:

- 1-800-237-5511, чтобы обратиться в службу поддержки
- 1-888-426-4343, чтобы узнать о доступных формах обслуживания.

---

## Информация о продукте

Если вы находитесь в США, позвоните по одному из следующих номеров:

- 1-800-IBM-CALL (1-800-426-2255) или 1-800-3IBM-OS2 (1-800-342-6672), чтобы заказать продукты или получить общую информацию.
- 1-800-879-2755, чтобы заказать публикации.

**<http://www.ibm.com/software/data/>**

На страницах DB2 в WWW содержится текущая информация DB2: новости, описания продуктов, учебные планы и т.д.

**<http://www.ibm.com/software/data/db2/library/>**

DB2 Product and Service Technical Library содержит ответы на часто задаваемые вопросы, исправления, книги и свежую техническую информацию по DB2.

**Примечание:** Эта информация может быть только в английском варианте.

**<http://www.elink.ibm.com/pbl/pbl/>**

На сайте заказов International Publications приводится информация о том, как заказывать книги.

**<http://www.ibm.com/education/certify/>**

На этом сайте представлена программа Professional Certification Program IBM и приводится информация о сертификационных испытаниях для многих продуктов IBM, в том числе DB2.

**ftp.software.ibm.com**

Зарегистрируйтесь как аноним. В каталоге /ps/products/db2 можно найти демо-версии, исправления, информацию и инструменты для DB2 и многих других продуктов.

**comp.databases.ibm-db2, bit.listserv.db2-l**

В этих группах новостей пользователи обмениваются опытом работы с продуктами DB2.

**В Compuserve: GO IBMDB2**

Введите эту команду, чтобы попасть на форумы IBM DB2 Family. Через эти форумы поддерживаются все продукты DB2.

Информацию о том, как связаться с IBM из других стран, смотрите в Приложении А книги *IBM Software Support Handbook*. Этот документ можно найти в Web, обратившись по адресу: <http://www.ibm.com/support/> и выбрав ссылку на IBM Software Support Handbook у нижнего края страницы.

**Примечание:** В некоторых странах авторизованные дилеры IBM должны обращаться не в центр поддержки IBM, а в структуры поддержки дилеров.







Напечатано в Дании

SH43-0146-01



Spine information:



IBM® DB2® Universal  
Database

Руководство администратора:  
Планирование

*Версия 7*