

Начало Работы С

DB2

Express-C

Обновлено
до 9.7

Идеально подходит разработчикам и администраторам приложений



Рауль Ф. Чон

с Иэн Хейкс,
Рав Ахуджа

Предисловие:
Д-р Арвинд Кришна

Книга, написанная сообществом для сообщества

НАЧАЛО РАБОТЫ С

DB2 Express-C

Книга, написанная сообществом для сообщества

РАУЛЬ ЧОН, ИЭН ХЕЙКС, РАВ АХУДЖА

ПРЕДИСЛОВИЕ: Д-Р АРВИНД КРИШНА



ТРЕТЬЯ РЕДАКЦИЯ

Третья редакция (июнь 2009 г.)

Третье издание (октябрь 2010 г.)

Данная редакция дополнена информацией о IBM DB2 Express-C версии 9.7.2 для Linux[®], UNIX[®] и Windows[®].

© IBM Corporation, 2007, 2010. Все права защищены.

Содержание

| | |
|---|-----------|
| Об этой книге | 11 |
| Уведомления и товарные знаки..... | 11 |
| Для кого эта книга? | 13 |
| Какова структура этой книги?..... | 13 |
| Книга для сообщества | 14 |
| Авторский коллектив..... | 15 |
| Благодарности..... | 15 |
| Предисловие..... | 16 |
| ЧАСТЬ I — ОБЗОР И УСТАНОВКА..... | 17 |
| Глава 1. Что такое DB2 Express-C? | 19 |
| 1.1 Свобода разработки, развертывания и распространения... без ограничений!..... | 20 |
| 1.2 Загрузка DB2 Express-C..... | 20 |
| 1.3 Помощь пользователям и техническая поддержка | 21 |
| 1.4 Серверы DB2 | 21 |
| 1.5 Клиенты и драйверы DB2 | 22 |
| 1.6 Свобода разработки приложений..... | 23 |
| 1.7 Версии и редакции DB2..... | 24 |
| 1.8 Переход на новую редакцию DB2 | 25 |
| 1.9 Обслуживание и обновление DB2 Express-C | 25 |
| 1.10 Сопутствующее бесплатное ПО и компоненты DB2..... | 26 |
| 1.10.1 IBM Data Studio..... | 26 |
| 1.10.4 DB2 Text Search (текстовый поиск) | 27 |
| 1.10.5 Сервер приложений WebSphere Application Server — Community Edition | 27 |
| 1.11 Краткий обзор | 27 |
| Глава 2 — Сопутствующие функции и продукты | 29 |
| 2.1 Функции, включенные в подписку (FTL) DB2 Express..... | 32 |
| 2.1.1 Пакеты исправлений..... | 33 |
| 2.1.2 HADR..... | 33 |
| 2.1.3 Репликация данных | 34 |
| 2.2 Функции, не доступные в DB2 Express-C | 34 |
| 2.2.1 Сегментирование базы данных | 35 |
| 2.2.2 Концентратор соединений | 35 |
| 2.2.3 Geodetic Extender | 35 |
| 2.2.4 LBAC..... | 35 |
| 2.2.5 Менеджер рабочей нагрузки (Workload Manager, WLM) | 36 |
| 2.2.6 Глубокое сжатие | 37 |
| 2.2.7 Совместимость с SQL..... | 38 |
| 2.3 Платные продукты, связанные с DB2..... | 39 |
| 2.3.1 DB2 Connect | 39 |
| 2.3.2 InfoSphere Federation Server | 40 |
| 2.3.3 InfoSphere Replication Server..... | 41 |
| 2.3.4 Optim Development Studio (ODS) | 41 |
| 2.3.5 Optim Database Administrator (ODA) | 42 |
| 2.4 Предложения DB2 в Amazon Elastic Compute Cloud..... | 42 |

6 Начало работы с DB2 Express-C

| | |
|--|------------|
| 2.5 Краткий обзор | 42 |
| Глава 3. Установка DB2..... | 43 |
| 3.1 Предварительные требования к установке | 43 |
| 3.2 Права на установку в операционной системе..... | 43 |
| 3.3 Мастер установки | 44 |
| 3.4 Проверка правильности установки..... | 51 |
| 3.5 Автоматическая установка | 52 |
| 3.6 Краткий обзор | 54 |
| 3.7 Упражнения | 54 |
| Глава 4. Среда DB2 | 59 |
| 4.1 Конфигурирование DB2..... | 67 |
| 4.1.1 Переменные среды..... | 68 |
| 4.1.2 Файл конфигурации менеджера базы данных (dbm cfg) | 68 |
| 4.1.3 Файл конфигурации базы данных (db cfg) | 71 |
| 4.1.4 Реестр профиля DB2 | 72 |
| 4.2 Сервер администрирования DB2 (устарело) | 73 |
| 4.3 Краткий обзор | 74 |
| 4.4 Упражнения | 74 |
| Глава 5. Инструменты DB2..... | 79 |
| 5.1 IBM Data Studio..... | 81 |
| 5.2 Центр управления (устарело) | 82 |
| 5.2.1 Запуск Центра управления | 85 |
| 5.3 Редактор команд (устарело) | 86 |
| 5.3.1 Запуск Редактора команд..... | 86 |
| 5.3.2 Добавление соединения с базой данных | 87 |
| 5.4 Мастер SQL Assist (устарело) | 88 |
| 5.5 Кнопка «Показать SQL» (устарело) | 89 |
| 5.6 Центр задач (устарело) | 90 |
| 5.6.1 База данных каталога инструментов (устарело) | 91 |
| 5.7 Журнал (устарело) | 92 |
| 5.7.1 Запуск Журнала | 94 |
| 5.8 Монитор работоспособности (устарело) | 94 |
| 5.8.1 Центр работоспособности (устарело) | 95 |
| 5.9 Самонастраивающийся менеджер памяти..... | 97 |
| 5.10 Использование сценариев | 97 |
| 5.10.1 Сценарии SQL..... | 98 |
| 5.10.2 Сценарии операционной системы (командного процессора) | 99 |
| 5.11 Особенности применения в Windows Vista..... | 100 |
| 5.12 Краткий обзор | 100 |
| 5.13 Упражнения | 101 |
| ЧАСТЬ II. ИЗУЧЕНИЕ DB2: АДМИНИСТРИРОВАНИЕ БАЗЫ ДАННЫХ | 105 |
| Глава 6. Архитектура DB2..... | 107 |
| 6.1 Модель процессов DB2 | 107 |
| 6.2 Модель памяти DB2..... | 109 |
| 6.3 Модель хранения данных DB2..... | 110 |

| | |
|--|------------|
| 6.3.1 Страницы и экстенты | 111 |
| 6.3.2 Буферные пулы | 111 |
| 6.3.3 Табличные пространства | 113 |
| 6.4 Краткий обзор | 118 |
| 6.5 Упражнения | 118 |
| Глава 7. Подключаемость клиента DB2..... | 123 |
| 7.1 Каталоги DB2..... | 123 |
| 7.1.1 Каталог системной базы данных | 123 |
| 7.1.2 Каталог локальной базы данных | 124 |
| 7.1.3 Каталог узла | 124 |
| 7.1.4 Каталог DCS | 124 |
| 7.2 Ассистент конфигурирования (устарело) | 124 |
| 7.2.1 Необходимая настройка со стороны сервера | 125 |
| 7.2.2 Необходимая настройка со стороны клиента | 128 |
| 7.2.3 Создание профилей клиента и сервера | 132 |
| 7.3 Краткий обзор | 135 |
| 7.4 Упражнения | 135 |
| Глава 8. Работа с объектами базы данных | 139 |
| 8.1 Схемы..... | 139 |
| 8.2 Общие синонимы (или алиасы) | 140 |
| 8.3 Таблицы | 140 |
| 8.3.1 Типы данных..... | 141 |
| 8.3.2 Столбцы идентификации | 145 |
| 8.3.3 Объекты последовательностей..... | 146 |
| 8.3.4 Таблицы системного каталога | 147 |
| 8.3.5 Объявленные глобальные временные таблицы (Declared global temporary tables, DGTT) | 147 |
| 8.3.6 Созданные глобальные временные таблицы (Create Global Temporary Tables, CGTT)..... | 149 |
| 8.4 Представления | 150 |
| 8.5 Индексы | 151 |
| 8.5.1 Советчик по структуре (Design Advisor) | 151 |
| 8.6 Целостность на уровне ссылок..... | 152 |
| 8.7 Эволюция схем..... | 153 |
| 8.8 Краткий обзор | 155 |
| 8.9 Упражнения | 155 |
| Глава 9. Утилиты перемещения данных | 159 |
| 9.1 Утилита EXPORT | 160 |
| 9.2 Утилита IMPORT | 161 |
| 9.3 Утилита LOAD..... | 162 |
| 9.4 Утилита db2move | 163 |
| 9.5 Утилита db2look..... | 163 |
| 9.6 Краткий обзор | 166 |
| 9.7 Упражнения | 166 |
| Глава 10. Безопасность базы данных | 169 |

8 Начало работы с DB2 Express-C

| | |
|--|------------|
| 10.1 Аутентификация | 170 |
| 10.2 Авторизация | 171 |
| 10.2.1 Привилегии | 172 |
| 10.2.2 Полномочия | 172 |
| 10.2.3 Роли | 177 |
| 10.3 Особенности применения привилегий группы | 178 |
| 10.4 Группа PUBLIC | 178 |
| 10.5 Операторы GRANT и REVOKE | 179 |
| 10.6 Проверка авторизации и привилегий | 179 |
| 10.7 Расширенная безопасность в Windows | 180 |
| 10.8 Краткий обзор | 181 |
| 10.9 Упражнения | 181 |
| Глава 11. Резервное копирование и восстановление | 187 |
| 11.1 Запись в журнал базы данных | 187 |
| 11.2 Типы журналов | 188 |
| 11.3 Способы записи в журнал | 189 |
| 11.3.1 Циклическая запись в журнал | 189 |
| 11.3.2 Ведение архивных журналов | 190 |
| 11.4 Запись в журнал базы данных из Центра управления | 191 |
| 11.5 Параметры записи в журнал | 192 |
| 11.6 Резервное копирование базы данных | 193 |
| 11.7 Восстановление базы данных | 195 |
| 11.7.1 Типы восстановления | 195 |
| 11.7.2 Восстановление базы данных | 196 |
| 11.8 Прочие операции с BACKUP и RESTORE | 196 |
| 11.9 Краткий обзор | 197 |
| 11.10 Упражнения | 197 |
| Глава 12. Задачи обслуживания | 201 |
| 12.1 REORG, RUNSTATS, REBIND | 201 |
| 12.1.1 Команда REORG | 202 |
| 12.1.2 Команда RUNSTATS | 202 |
| 12.1.3 BIND/REBIND | 203 |
| 12.1.4 Задачи обслуживания в Центре управления | 204 |
| 12.2 Варианты обслуживания | 205 |
| 12.3 Краткий обзор | 207 |
| 12.4 Упражнения | 207 |
| Глава 13. Параллельное использование и блокировка | 211 |
| 13.1 Транзакции | 211 |
| 13.2 Параллельное использование | 212 |
| 13.3 Проблемы, связанные с отсутствием контроля параллельного использования | 213 |
| 13.3.1 Потерянное обновление | 213 |
| 13.3.2 Недостоверное чтение | 214 |
| 13.3.3 Неповторяющееся чтение | 214 |
| 13.3.4 Фантомное чтение | 215 |

| | |
|---|------------|
| 13.4 Уровни изоляции | 216 |
| 13.4.1 Недостоверное чтение | 216 |
| 13.4.2 Стабильность курсора | 217 |
| 13.4.3 Стабильность чтения..... | 219 |
| 13.4.4 Повторяющееся чтение..... | 219 |
| 13.4.5 Сравнение уровней изоляции..... | 219 |
| 13.4.6 Установка уровня изоляции..... | 220 |
| 13.5 Эскалация блокировок..... | 221 |
| 13.6 Мониторинг блокировок..... | 223 |
| 13.7 Ожидание блокировки | 224 |
| 13.8 Причины и обнаружение взаимоблокировки | 224 |
| 13.9 Оптимальные подходы к параллельному использованию и блокировке ... | 226 |
| 13.10 Краткий обзор | 229 |
| 13.11 Упражнения | 229 |
| ЧАСТЬ III — ИЗУЧЕНИЕ DB2: РАЗРАБОТКА ПРИЛОЖЕНИЙ..... | 235 |
| Глава 14. Введение в разработку приложений DB2 | 237 |
| 14.1 Разработка приложений DB2: общий обзор | 237 |
| 14.2 Разработка на стороне сервера | 239 |
| 14.2.1 Хранимые процедуры..... | 239 |
| 14.2.2 Пользовательские функции (UDF) | 240 |
| 14.2.3 Триггеры | 240 |
| 14.3 Разработка на стороне клиента..... | 241 |
| 14.3.1 Встроенный SQL | 241 |
| 14.3.2 Статический и динамический SQL | 242 |
| 14.3.3 CLI и ODBC | 244 |
| 14.3.4 JDBC, SQLJ и pureQuery | 247 |
| 14.3.5 Связывание и внедрение объектов, базы данных (Object Linking and Embedding, Database, OLE DB) | 249 |
| 14.3.6 ADO.NET | 250 |
| 14.3.7 PHP..... | 252 |
| 14.3.8 Ruby on Rails | 252 |
| 14.3.9 Perl..... | 253 |
| 14.3.10 Python..... | 253 |
| 14.4 XML и технология pureXML в DB2 | 253 |
| 14.5 Веб-службы..... | 254 |
| 14.6 Административные API | 255 |
| 14.7 Прочие разработки..... | 256 |
| 14.7.1 Работа с Microsoft Access и Microsoft Excel | 256 |
| 14.8 Инструменты разработки | 257 |
| 14.9 Образцы программ..... | 257 |
| 14.10 Краткий обзор | 258 |
| Глава 15. Технология pureXML в DB2..... | 259 |
| 15.1 Использование XML в базах данных..... | 260 |
| 15.2 Базы данных XML | 260 |
| 15.2.1 Базы данных с поддержкой XML | 260 |

10 Начало работы с DB2 Express-C

| | |
|---|------------|
| 15.2.2 Истинные базы данных XML..... | 261 |
| 15.3 XML в DB2..... | 262 |
| 15.3.1 Преимущества технологии pureXML..... | 263 |
| 15.3.2 Основы XPath | 265 |
| 15.3.3 Основы XQuery | 267 |
| 15.3.4 Вставка XML-документов | 269 |
| 15.3.5 Запрос XML-данных..... | 272 |
| 15.3.6 Операции соединения с SQL/XML | 279 |
| 15.3.7 Операции соединения с XQuery | 280 |
| 15.3.8 Операции обновления и удаления..... | 281 |
| 15.3.9 Индексирование XML | 283 |
| 15.4 Работа со схемами XML..... | 284 |
| 15.4.1 Регистрация XML-схем | 284 |
| 15.4.2 Подтверждение схемы XML..... | 287 |
| 15.4.3 Прочая поддержка XML..... | 288 |
| 15.6 Краткий обзор | 289 |
| 15.7 Упражнения | 289 |
| Приложение А. Устранение неисправностей..... | 291 |
| A.1 Получение более подробной информации о кодах ошибок | 292 |
| A.2 SQLCODE и SQLSTATE | 292 |
| A.3 Журнал административных уведомлений DB2..... | 293 |
| A.4 db2diag.log | 293 |
| A.5 Трассировка CLI | 294 |
| A.6 Дефекты и исправления DB2 | 294 |
| Приложение В. Справочные материалы и ресурсы | 295 |
| B.1 Справочные материалы | 295 |
| B.2 Веб-сайты: | 295 |
| B.3 Книги | 297 |
| B.4 Контактные адреса электронной почты..... | 297 |

Об этой книге

Уведомления и товарные знаки

IBM Восточная Европа/Азия
123317, Москва
Краснопресненская наб., 18
Тел.: +7 (495) 775-8800, +7 (495) 940-2000
Факс: +7 (495) 940-2070

Домашняя страница IBM находится по адресу ibm.com/ru

IBM, логотип IBM, ibm.com, AIX, AS/400, DB2, DB2 Connect, i5/OS, Informix, InfoSphere, Lotus, Omnipfind, Optim, Passport Advantage, pureXML, Rational, System i, System z, Tivoli, WebSphere и z/OS являются товарными знаками или зарегистрированными товарными знаками корпорации International Business Machines в США и (или) других странах. Если эти и другие элементы IBM, указанные как товарные знаки, обозначены при первом употреблении в данном материале символом товарного знака (® или ™), эти символы указывают на зарегистрированные в США или согласно общему законодательству товарные знаки, принадлежащие IBM на момент публикации данного материала. Такие товарные знаки могут также являться зарегистрированными товарными знаками либо товарными знаками, охраняемыми нормами общего права, в других странах.

Текущий перечень товарных знаков IBM выложен в сети Интернет на странице «Copyright and trademark information» («Авторские права и товарные знаки») по адресу ibm.com/legal/copytrade.shtml

Java, все товарные знаки и логотипы, основанные на Java, являются товарными знаками корпорации Sun Microsystems в США и (или) других странах.

Linux является зарегистрированным товарным знаком Линуса Торвальдса (Linus Torvalds) в США и (или) в других странах.

Microsoft, Excel и Windows являются товарными знаками Microsoft Corporation в США и (или) других странах.

UNIX является зарегистрированным товарным знаком The Open Group в США и (или) других странах.

Другие названия компаний, продуктов и услуг могут являться товарными знаками или знаками обслуживания других лиц и компаний.

Запрещено копировать или воспроизводить данный документ частично или полностью в любой форме или любыми способами, а также переводить его на другие языки без предварительного согласия всех вышеперечисленных владельцев авторских прав.

Компания IBM не дает никаких гарантий и не делает заявлений относительно содержимого данного документа, а также открыто отказывается от всех подразумеваемых гарантий товарной пригодности или пригодности для определенной цели. Компания IBM не несет ответственности за возможные ошибки данного документа. Содержащаяся в данном документе информация может быть изменена без уведомления. Компания IBM сохраняет за собой право вносить такие изменения или правки без уведомления других лиц. Компания IBM не обязуется обновлять информацию, содержащуюся в данном документе.

12 Начало работы с DB2 Express-C

Содержащаяся в данном документе информация о продуктах сторонних компаний получена у поставщиков таких продуктов. Компания IBM не проверяла такие продукты сторонних компаний и не может подтвердить точность их работы, совместимость и прочие заявления. Со всеми вопросами о возможностях продуктов сторонних компаний следует обращаться к поставщикам таких продуктов.

Ссылки в данной публикации на продукты, программы или услуги компании IBM не подразумевают намерения компании IBM обеспечить возможность их приобретения во всех странах, в которых ведет деятельность компания IBM.

Любая ссылка на продукт, программу или услугу IBM не подразумевает, что можно использовать только продукты, программы или услуги корпорации IBM. Допускается использование вместо них какого-либо функционально эквивалентного изделия, программы или услуги.

Аппаратное обеспечение IBM производится из новых или использованных деталей. В некоторых случаях аппаратные средства могут быть не новыми и бывшими в эксплуатации. Вне зависимости от этого применяются условия гарантии IBM.

Данная публикация служит только для общего руководства.

Информация может быть изменена без предварительного уведомления. Чтобы получить последнюю информацию о продуктах и услугах IBM, свяжитесь с местным отделом сбыта или торговым посредником IBM.

Данная публикация содержит адреса в сети Интернет, не относящиеся к компании IBM. IBM не несет ответственности за информацию, опубликованную на этих веб-сайтах.

Корпорация IBM не предоставляет консультаций в области права, учета и аудита, не заявляет и не гарантирует, что ее услуги и продукты обеспечивают соответствие каким-либо законам. Клиенты несут ответственность за соблюдение применимых законов и постановлений, включая национальные законы и постановления.

На фотографиях могут быть изображены проектные модели.

© IBM Corporation, 2011.
Все права защищены.

Для кого эта книга?

Эта книга предназначена для всех, кто работает или планирует работать с базами данных — администраторов баз данных (АБД), разработчиков приложений, консультантов, разработчиков структуры ПО, менеджеров по продуктам, инструкторов и студентов.

Какова структура этой книги?

Часть I «Обзор и установка» объясняет, что такая редакция DB2 Express-C, представляет семейство продуктов и функций DB2, помогает в процессе установки и создания баз данных, а также описывает доступные в DB2 инструменты.

Часть II «Изучение DB2: администрирование базы данных» предназначена для того, чтобы познакомить пользователя со средой, архитектурой, удаленным доступом, объектами базы данных, перемещением данных (импорт/экспорт/загрузка), безопасностью, резервным копированием и восстановлением, параллельным использованием и блокировкой DB2, а также прочими распространенными задачами обслуживания.

Часть III «Изучение DB2: разработка приложений» представляет возможности разработки приложений в DB2, в том числе разработки со стороны сервера и клиента. Также рассмотрены SQL/XML, XQuery и pureXML.

В Приложении приведена полезная информация об устранении неисправностей.

Большинство глав содержит упражнения; все входные файлы, необходимые для выполнения этих упражнений, находятся в архиве **expressc_book_exercises_9.7.zip**, который распространяется с этой книгой.

Представленные в этой книге материалы также используются в курсах, являющихся частью программы обучения **DB2 on Campus**, и соответствуют интерактивным обучающим видеопрезентациям, которые можно просмотреть по адресу www.channeldb2.com/oncampus. С более подробной информацией о программе обучения «DB2 on Campus» можно ознакомиться на веб-сайте DB2 Express-C: www.ibm.com/db2/express/students.html.

Примечание.

Чтобы узнать о программе обучения «DB2 on Campus» более подробно, просмотрите видео по адресу <http://www.channeldb2.com/video/video/show?id=807741:Video:3902>

В третью редакцию книги внесены некоторые изменения и дополнения. Обладателям второй редакции этой книги, где описывается DB2 9.5, будет несложно обнаружить внесенные изменения, соответствующие новым функциям или обновлениям DB2 версии 9.7. Для удобства все изменения отмечены таким значком:



Книга для сообщества

Эта книга создана командой DB2 Express-C. Ее онлайн-версия доступна для сообщества DB2 Express-C совершенно бесплатно. С момента своего написания книга загружалась больше 85 000 раз и была переведена на девять языков добровольцами со всего мира. Вот действительно яркий пример совместных усилий сообщества! Все желающие оставить отзыв, предложить новые или улучшить существующие материалы, или же помочь с переводом этой книги на другой язык, могут отправить сообщение с описанием своих планов на адрес db2x@ca.ibm.com, указав «DB2 Express-C book changes» в теме сообщения.

Успех этой книги послужил источником вдохновения для разработки более 25 новых бесплатных электронных книг о продуктах компании IBM и технологиях сторонних разработчиков. Все эти книги являются частью проекта ***DB2 on Campus Book Series***, запущенного в январе 2010 г.

С более подробной информацией об этой книге и проекте «DB2 on Campus Book Series» можно ознакомиться на веб-сайте IBM DB2 Express-C по адресу ibm.com/db2/express

Авторский коллектив

Перечисленные ниже люди предоставили содержимое для этой книги и внесли значительный вклад в её создание.

Рауль Ф. Чон — ведущий автор

Рауль — менеджер программы обучения «DB2 on Campus» при лаборатории IBM в Торонто.

Иэн Хейкс — соавтор и редактор

Иэн — бывший координатор сообщества DB2 Express-C. Сейчас он занимает должность специалиста в области удобства использования (юзабилити) в лаборатории IBM в Торонто.

Рав С. Ахуджа — соавтор и издаватель

Рав занимает должность старшего менеджера по продуктам DB2 в лаборатории IBM в Торонто.

Благодарности

Выражаем перечисленным ниже людям искреннюю благодарность за помощь в подготовке материалов этой книги:

- Тед Вассерман, Клара Лю и Пол Ип из лаборатории IBM в Торонто — разработка материалов, послуживших основой для этой книги.
- Дон Чамберлин и Синди Саракко — статьи IBM developerWorks об XQuery; Маттиас Никола — презентации по pureXML.
- Кевин Цап и Грант Хатчинсон — разработка технических инструкций по DB2
- Катерина Боячок и Наташа Толуб — дизайн обложки этой книги.
- Сьюзен Виссер — проверка и помощь в публикации книги.

Предисловие

Инновации — основа технологического прогресса. В компании IBM инновации всегда были неотъемлемой частью развития серверов данных. В 1960-х и 1970-х годах компания IBM первой внедряла методы управления данными, а сейчас она продолжает предоставлять инновационные технологии управления информацией, о чем свидетельствуют тысячи патентов на средства управления данными, авторами которых являются технологии IBM. Как следствие, многие крупнейшие в мире организации полагаются на продукты IBM, включая DB2, как на движущую силу своих критически важных решений по управлению данными с наиболее высокими требованиями.

Однако DB2 больше не ориентируется исключительно на крупные предприятия. С выпуском DB2 Express-C удостоенная многих наград технология DB2 стала доступной для малых и средних компаний, да еще и бесплатно! Хотя существуют и другие серверы данных с открытым исходным кодом или распространяемые бесплатно, по сравнению с ними DB2 Express-C предоставляет уникальные преимущества.

В DB2 Express-C представлено множество технологических достижений. Благодаря таким инновациям открываются новые возможности, упрощается администрирование, повышается производительность и уменьшается общая стоимость инфраструктуры.

Гибридная технология DB2 Express-C может использоваться для управления как реляционными данными, так и данными расширяемого языка разметки (XML) в *формате оригинала*. Благодаря этому DB2 является превосходной основой для нового поколения приложений на базе сервисно-ориентированной архитектуры (SOA) и Web 2.0, использующих большие объемы XML-данных. В отличие от других «бесплатных» серверов данных, DB2 Express-C не устанавливает ограничений на объем хранимых данных или количество баз данных, которое можно создать в системе. К тому же, если потребуется поддержка или помочь компании IBM, достаточно одного щелчка мыши.

Эта книга является руководством по началу работы с DB2 Express-C. Она поможет понять концепции DB2, а также даст возможность выработать навыки администрирования DB2 и разработки приложений с использованием этого сервера данных. Полученные навыки и знания будут также полезны при работе с другими расширенными редакциями DB2 для Linux®, UNIX® и Windows®.

Хотя DB2 Express-C не является продуктом с открытым исходным кодом, компания IBM всецело поддерживает и поощряет инициативы сообщества. Мне очень приятно, что эта книга подготовлена сообществом DB2 Express-C и будет бесплатно доступна всем его участникам. Искренне приветствуются попытки пользователей улучшить или обновить представленную в этой книге информацию, воспользовавшись своими навыками и опытом, а также помочь в переводе книги на другие языки, чтобы сделать её ещё более ценной для всего сообщества.



Арвинд Кришна
Генеральный директор
Департамент управления информацией, IBM Software Group

ЧАСТЬ I — ОБЗОР И УСТАНОВКА

1

Глава 1. Что такое DB2 Express-C?

Сервер данных DB2 Express-C («DB2 Express-C») входит в семейство мощных серверов данных IBM DB2 для управления реляционными и XML-данными. [DB2 Express-C](#) — бесплатная и простая в использовании редакция DB2 без функциональных ограничений. Буква «С» в названии DB2 Express-C означает «Сообщество» (от англ. «Community»). Сообщество пользователей DB2 Express-C объединилось для взаимной помощи, как в сети, так и в автономном режиме. В сообщество DB2 Express-C входят различные люди и компании, занимающиеся проектированием, разработкой, развертыванием или применением решений, основанных на базах данных. В состав сообщества входят:

- Разработчики приложений, которым нужно ПО базы данных, основанное на открытых стандартах, для построения автономных приложений, приложений «клиент — сервер», веб-приложений и приложений масштаба предприятия.
- Независимые поставщики программного обеспечения (ISV), производители аппаратного обеспечения, производители инфраструктурного стека продуктов и поставщики прочих решений, которые хотят объединить со своими решениями или встроить в них полнофункциональный сервер данных.
- Консультанты, администраторы баз данных и архитекторы ИТ, которым нужен надежный сервер данных для обучения, повышения квалификации, оценивания и создания прототипов.
- Стартапы, компании малого и среднего бизнеса, которым нужен надежный сервер данных для приложений и текущей работы.
- Люди, увлеченные базами данных, и энтузиасты передовых технологий, ожидающиеся в удобном сервере данных для разработки программ Web 2.0 и приложений нового поколения.
- Студенты, преподаватели и другие пользователи из научной сферы, которым требуется многосторонний сервер данных для обучения, организации учебных курсов, реализации проектов и исследований.

DB2 Express-C имеет такую же основную функциональность и программный код, как у других коммерческих редакций DB2 для Linux, UNIX и Windows. DB2 Express-C можно использовать на 32- или 64-разрядных системах с операционными системами Linux или Windows. Сервер данных также доступен для Solaris (x64), а его бета-версия — для Mac OS X (x64). Он может работать в системах с любым числом процессоров и объемом памяти, а также не предусматривает никаких особых требований к запоминающим устройствам или к настройке системы. DB2 Express-C также бесплатно включает в себя pureXML. pureXML — это уникальная технология DB2 для хранения и обработки документов XML в исходном формате.

1.1 Свобода разработки, развертывания и распространения... без ограничений!

В этом предложении подытожены ключевые идеи, лежащие в основе DB2 Express-C:

- **Свобода разработки:** если вы разработчик приложений и нуждаетесь в базе данных для своих продуктов, можете воспользоваться DB2 Express-C.
- **Свобода развертывания:** если вы работаете в производственной среде и нуждаетесь в системе управления данными для хранения важнейшей документации, можете воспользоваться DB2 Express-C.
- **Свобода распространения:** если вы разрабатываете приложение или инструмент, для которого требуется встроенный сервер данных, можете добавить DB2 Express-C. Даже если сервер данных DB2 Express-C встроен в ваше приложение и распространяется с каждой его копией, он все равно является совершенно бесплатным. Для распространения DB2 Express-C в составе своего приложения вам необходимо зарегистрироваться в компании IBM, однако такая регистрация также является бесплатной.
- **Никаких ограничений:** в то время, как конкурирующие решения по работе с базами данных ограничивают размер баз данных, их количество или число пользователей, в DB2 Express-C нет НИКАКИХ ограничений по объему данных. Ваша база данных может разрастаться, не нарушая при этом условия лицензионного соглашения. Кроме того, лицензия не накладывает ограничения по количеству подключений или пользователей на сервер.

Примечание.

Чтобы более подробно узнать о сервере данных DB2 Express-C, его роли в мире информации «по требованию» и Web 2.0, просмотрите эту видео-презентацию:
<http://www.channeldb2.com/video/video/show?id=807741:Video:3922>

1.2 Загрузка DB2 Express-C

Все образы DB2 Express-C можно бесплатно загрузить по адресу ibm.com/db2/express и использовать. Доступны следующие образы:

- DB2 Express-C 9.7.2 для Windows
- DB2 Express-C 9.7.2 для Windows, 64-разрядн.
- DB2 Express-C 9.7.2 для Linux
- DB2 Express-C 9.7.2 для Linux, 64-разрядн.
- DB2 Express-C 9.7.2 для Linux на Power
- DB2 Express-C 9.7.2 для Solaris x86-64
- DB2 Express-C 9.5.2 (бета-версия) для Mac OS X.

Примечание.

Для Windows также доступна упрощенная версия DB2 Express-C, ее размер на 44% меньше обычной. Данная версия доступна только на английском языке, не имеет инструментов графического интерфейса пользователя (GUI) и функции Text Search

(текстовый поиск).

1.3 Помощь пользователям и техническая поддержка

При возникновении вопросов технического характера о DB2 Express-C можно опубликовать их на [форуме DB2 Express-C](#). За этим бесплатным форумом следят эксперты компании IBM по DB2, хотя большинство ответов по собственной инициативе оставляют члены сообщества.

Компания IBM также дает пользователям возможность недорого приобрести годовую подписку на программное обеспечение сервера данных DB2 Express («DB2 Express») (также известную как лицензия с фиксированным сроком действия, или FTL – Fixed Term License). Эта подписка включает круглосуточную техническую поддержку компании IBM и получение обновлений программного обеспечения. Кроме поддержки и обслуживания программного обеспечения, за невысокую ежегодную плату (приблизительно 1990 долл. США за сервер в год на территории США — стоимость может отличаться в других странах) вы получаете возможность использовать следующие дополнительные функции: кластеризация для высокой доступности и аварийного восстановления (High Availability and Disaster Recovery, HADR), репликация SQL (для репликации данных с другими серверами DB2) и сжатие резервных копий (для создания сжатых резервных копий базы данных). Дополнительную информацию о подписке вы можете найти по адресу ibm.com/db2/express/support.html

1.4 Серверы DB2

Все редакции сервера DB2 содержат одинаковые основные компоненты; они объединены таким образом, чтобы пользователи могли выбирать нужные функции по подходящей цене. На *рис. 1.1* показаны различные редакции продукта DB2.

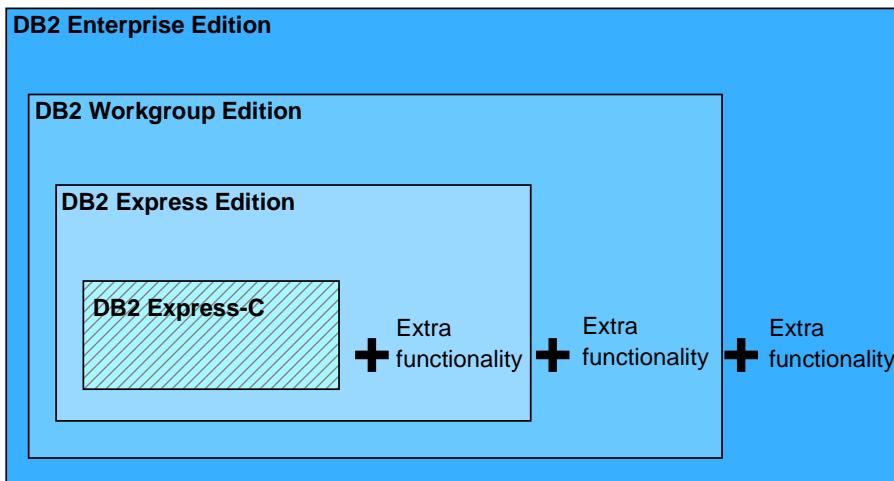


Рисунок 1.1 — Серверы DB2

Как показано на *рис. 1.1*, DB2 Express-C отличается от DB2 Express только отсутствием нескольких компонентов. Сервер данных DB2 Express-C — бесплатный для сообщества. Техническую поддержку можно получить на бесплатном интернет-

форуме, а при покупке годовой подписки (лицензии с фиксированным сроком действия (FTL) для DB2 Express) можно также воспользоваться официальной круглосуточной технической поддержкой DB2 от компании IBM.

На *рис. 1.1* также наглядно объясняется, почему с редакции DB2 Express-C легко перейти на более новую редакцию. Если вы планируете выполнить обновление до других редакций сервера DB2, все они имеют одинаковые основные компоненты. Это также означает, что любое приложение, разработанное для одной редакции, сможет без изменений работать в других редакциях. Кроме того, любые полученные в одной редакции навыки будут применимы в других редакциях.

1.5 Клиенты и драйверы DB2

Клиент DB2 включает всю необходимую функциональность для подключения к серверу DB2, однако необходимость в установке клиента DB2 возникает не всегда. К примеру, приложениям, использующим драйвер JDBC Type 4 для подключения к серверу DB2, требуется только установка драйвера JDBC. Доступно несколько разновидностей клиентов и драйверов DB2:

- Клиент «IBM Data Server Client»: наиболее полный клиент, включает инструменты графического интерфейса пользователя, драйверы.
- Клиент «IBM Data Server Runtime Client»: облегченный клиент с базовой функциональностью и драйверами.
- Пакет модулей «DB2 Runtime Client Merge Modules для Windows»: используется преимущественно для внедрения клиента DB2 времени выполнения, как части установки приложения в Windows.
- Драйвер «IBM Data Server Driver для JDBC и SQLJ»: позволяет приложениям Java™ подключаться к серверам DB2 без установки полного клиента.
- Драйвер «IBM Data Server Driver для ODBC и CLI»: позволяет приложениям, использующим интерфейсы ODBC и CLI, подключаться к серверу DB2 без установки клиента, занимающего много места на диске.
- Пакет драйверов «IBM Data Server Driver Package»: включает специальный драйвер для Windows с поддержкой среды .NET, в дополнение к ODBC, CLI и открытым исходным кодам. Ранее, этот драйвер был известен как драйвер «IBM Data Server для ODBC, CLI и .NET».

new in
V9.7

На *рис. 1.2* показаны различные доступные клиенты и драйверы DB2.

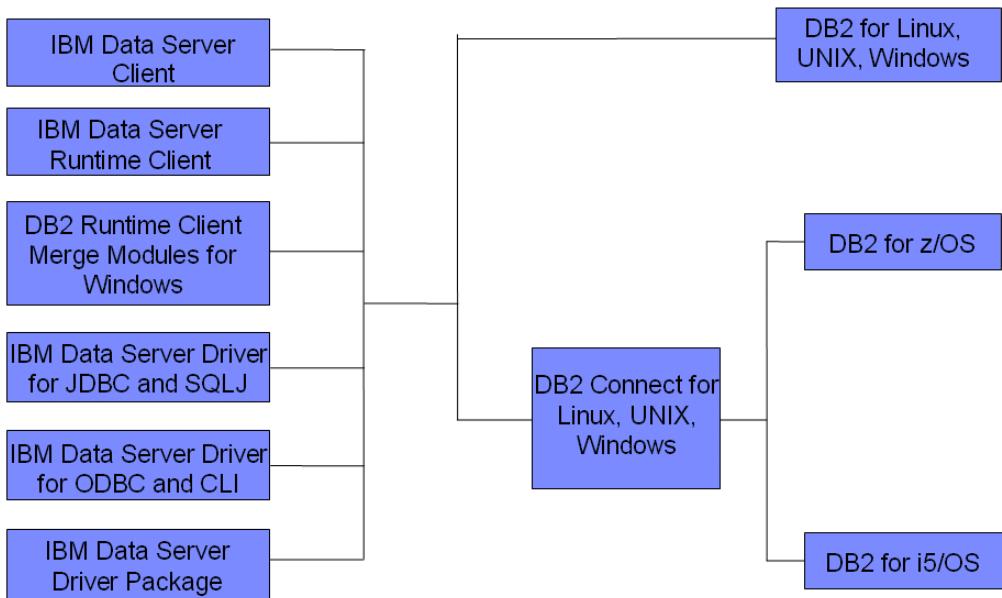


Рисунок 1.2 — Клиенты и драйверы DB2

В левой части *рис. 1.2* показаны все клиенты и драйверы DB2. Хотя все клиенты DB2 содержат требуемые драйверы, начиная с программного обеспечения сервера данных DB2 («DB2») вер. 9 мы также предоставляем драйверы отдельно. Все клиенты и драйверы DB2 распространяются бесплатно и доступны для загрузки на веб-сайте DB2 Express-C. Клиенты и драйверы могут использоваться для подключения к серверу DB2 на Linux, UNIX или Windows. Для подключения к серверу DB2 для z/OS или DB2 для i5/OS необходимо устанавливать через сервер DB2 Connect (показан в центре *рис. 1.2*). Программное обеспечение DB2 Connect («DB2 Connect») рассматривается в *Глазе 2*.

Примечание.

Хотя в этой книге описывается сервер данных DB2, клиенты IBM Data Server могут также подключаться к другим серверам данных семейства IBM, например Informix. Этим объясняется использование общего названия «Клиент IBM Data Server», а не более конкретного «Клиент DB2».

1.6 Свобода разработки приложений

DB2 предоставляет среду разработки приложений, основанную на стандартах и прозрачную для всего семейства DB2. Стандартизация SQL на всей линейке продуктов DB2 обеспечивает общий набор прикладных программных интерфейсов для доступа к базе данных.

Кроме того, каждый продукт DB2 предоставляет средства прекомпиляции SQL и прикладные программные интерфейсы (API), с помощью которых разработчики могут встраивать статические и динамические SQL-запросы в портативные приложения.

24 Начало работы с DB2 Express-C

DB2 даже имеет собственного провайдера под управлением .NET и интеграцию с инструментами Microsoft® Visual Studio.

Доступные в DB2 языки и стандарты:

- SQL, XQuery, XPath
- C/C++ (CLI, ODBC и встроенный SQL)
- Java (JDBC и SQLJ)
- COBOL
- PHP
- Perl
- Python
- Ruby on Rails
- Языки .NET
- OLE-DB
- ADO
- MS Office: Excel®, Access, Word
- Веб-службы

1.7 Версии и редакции DB2

Если вы только начинаете знакомство с DB2, разница между понятиями «версия DB2» и «редакция DB2» может быть не совсем понятна.

Каждые несколько лет компания IBM официально выпускает новую версию DB2. Версия включает новые функции и значительные улучшения продукта. В настоящий момент, компания IBM официально поддерживает DB2 версии 9. Каждая версия может иметь несколько выпусков, которые являются обновлениями и могут включать новые функциональные возможности, однако такие обновления недостаточно существенны для выпуска новой версии. Например, 9.5 и 9.7 — это выпуски DB2 версии 9. Последние несколько лет, компания IBM представляла новые выпуски DB2 каждые один-два года, а версии обычно выпускаются каждые три года или даже реже. Наиболее актуальный выпуск — версия 9.7, ставшая публично доступной (generally available – GA) в июне 2009 года. Каждый выпуск может также иметь несколько уровней изменений, которые обычно содержат исправления или соответствуют уровням пакетов исправлений, но редко — новые функциональные возможности. На момент написания книги наиболее актуальными являлись версия, выпуск и уровень изменений (version, release, modification level – V, R, M, соответственно) DB2 Express-C 9.7.0, что соответствует уровню кода 9.7 с пакетом изменений 0, т. е. сервер данных на уровне GA.

С другой стороны, редакции — это особые предложения или группы пакетов в рамках каждой версии. Выше упоминалось, что редакция является пакетом разнообразных функций, предоставляемых за определенную плату с определенными лицензионными правами. Сервер данных DB2 вер. 9.7 (также известный как DB2 9.7)

имеет несколько редакций, например DB2 Express-C 9.7, DB2 Express 9.7, DB2 Workgroup 9.7 и DB2 Enterprise 9.7 (см. *рис. 1.1*).

1.8 Переход на новую редакцию DB2

С ростом базы данных может потребоваться обновление до редакции DB2, поддерживающей более крупные аппаратные конфигурации. В таком случае очень просто перейти на другую редакцию DB2:

- При обновлении с редакции DB2 Express-C до редакции DB2 Express (лицензия с фиксированным сроком действия) на одном компьютере необходимо всего лишь применить лицензию с помощью команды db2licm.
- При обновлении до другой редакции DB2 на одном компьютере необходимо удалить DB2 Express-C и установить новую редакцию DB2. Удаление DB2 Express-C не удаляет базы данных (но, в любом случае, рекомендуем создать резервные копии).
- Если после обновления DB2 новая редакция будет установлена на другом, более мощном компьютере с такой же операционной системой, установите новую редакцию DB2 на более мощный компьютер, создайте резервные копии баз данных на старом компьютере, перенесите образы резервных копий на более мощный компьютер и восстановите их на более мощном компьютере. Также необходимо сохранить настройки конфигурации экземпляра (dbm cfg) на старом компьютере и применить их на более мощном. Команды резервного копирования и восстановления рассмотрены более подробно в *Главе 11 «Резервное копирование и восстановление»*. Настройки конфигурации экземпляра (dbm cfg) рассматриваются более подробно в *Главе 5 «Среда DB2»*.
- В любом случае, модификация приложения-клиента не потребуется.

1.9 Обслуживание и обновление DB2 Express-C

Установочные образы DB2 Express-C периодически обновляются. Обычно, такие обновления происходят при появлении новых выпусков или версий, или же при накоплении значительного количества исправлений ошибок продукта. Раньше, обновления DB2 Express-C выпускались один раз в год. Однако, обратите внимание на то, что, поскольку DB2 Express-C является бесплатным безгарантийным предложением, не предусмотрено никаких официальных отладочных версий или регулярных запланированных пакетов исправлений (которые выпускаются несколько раз в год). После выхода обновления или нового выпуска DB2 Express-C предыдущие выпуски DB2 Express-C больше не поддерживаются.

Как уже обсуждалось ранее, если требуется доступ к исправлениям безопасности и регулярным запланированным обновлениям программного обеспечения или пакетам исправлений ошибок, компания IBM предлагает годовую подписку на лицензию DB2 Express (FTL). После покупки подписки, установку DB2 Express-C можно обновить с помощью лицензионного ключа для FTL, который дает право пользоваться технической поддержкой DB2 и предоставляет доступ к обновлениям и пакетам исправлений, когда действует лицензия подписки. Подписка на лицензию также дает

26 Начало работы с DB2 Express-C

право бесплатно воспользоваться обновлениями версии, а при необходимости — возможность сохранить определенную версию или выпуск и просто применять пакеты исправлений и исправления безопасности, пока поддерживается соответствующий выпуск и действует годовая подписка.

1.10 Сопутствующее бесплатное ПО и компоненты DB2

Все программное обеспечение, доступное на странице загрузки DB2 Express-C (www.ibm.com/db2/express/download.html), является бесплатным. Кроме программного обеспечения DB2 Express-C, можно также бесплатно загрузить и использовать другие пакеты полезных программ:

- Надстройки Visual Studio
- DB2 Spatial Extender

В работе также могут пригодиться дополнительные стартовые пакеты инструментов на основе DB2 Express-C, которые можно загрузить с веб-сайта IBM Alphaworks (www.alphaworks.ibm.com/datamgmt):

- Стартовый пакет инструментов для DB2 на Rails (www.alphaworks.ibm.com/tech/db2onrails/)
- Стартовый пакет инструментов Web 2.0 для DB2 (www.alphaworks.ibm.com/tech/web2db2)

В качестве бесплатного упрощенного сервера приложений компания IBM может предложить:

- Сервер приложений WebSphere Application Server — Community Edition (WAS CE)

Примечание.

Хотя в DB2 пространственные возможности доступны уже почти 10 лет, мало кто из пользователей знает о них. Функцию DB2 Spatial Extender можно бесплатно использовать во всех редакциях DB2, в том числе и DB2 Express-C. Spatial Extender дает возможность работать с пространственными и геодезическими данными, используя SQL. Эта возможность поможет ответить на такие вопросы, как «Где находится ближайшая торговая точка для каждого клиента, проживающего в Торонто и в прошлом году потратившего на наши продукты больше 3000 долл. США?» DB2 Spatial Extender можно использовать даже для медицинских приложений. К примеру, эта функция поможет ответить на вопрос «Какой характер имеют злокачественные клетки на МРТ мозга?»

С более подробной информацией можно ознакомиться в разделе [О функции DB2 Spatial Extender](#) в информационном центре баз данных IBM DB2 для Linux, UNIX и Windows.

1.10.1 IBM Data Studio

IBM Data Studio — это инструмент на базе Eclipse, который позволяет управлять базами данных и помогает разрабатывать XQuery-запросы, SQL-сценарии, пользовательские функции и хранимые процедуры. В него включен интегрированный

отладчик. Кроме того, IBM Data Studio дает возможность работать с диаграммами моделирования физических данных (physical data modeling – PDM) для понимания взаимосвязи сущностей между таблицами. Этот инструмент, поддерживающий принцип перетаскивания экраных объектов (drag and drop – «перетащи и оставь») и не требующий программирования, также может помочь в разработке и публикации данных в форме веб-служб. IBM Data Studio заменяет такие инструменты DB2, как Центр управления и Редактор команд, которые в настоящее время являются устаревшими (они включены в DB2, но больше не разрабатываются). IBM Data Studio более подробно рассматривается в Главе 5 «Инструменты DB2».

1.10.4 DB2 Text Search (текстовый поиск)

DB2 Text Search — это дополнительный интегрированный компонент DB2. Он работает на базе технологии IBM OmniFind и дает возможность выполнять мощный, быстрый и подробный полнотекстовый поиск в текстовых документах, в том числе любых документах XML, хранящихся в DB2 в исходном формате. Этот компонент использует лингвистическую обработку для поиска различных словоформ искомого термина в тексте. К примеру, при поиске слова «study» DB2 Text Search также найдет такие словоформы, как «studies» или «studied».

Чтобы установить компонент DB2 Text Search, воспользуйтесь пользовательской установкой DB2 Express-C и выберите функцию DB2 Text Search в категории «Server support» (Поддержка сервера).

Примечание.

Такая же функциональность доступна в расширении DB2 под названием Net Search Extender (NSE). По сравнению с DB2 Text Search, расширение NSE считается устаревшим.

1.10.5 Сервер приложений WebSphere Application Server Community Edition

WebSphere Application Server Community Edition (WASCE) от IBM — это упрощенный сервер приложений Java EE 5, доступный бесплатно. Построенный на технологии Apache Geronimo, он использует последние инновации сообщества разработчиков открытого ПО для предоставления интегрированной, доступной и гибкой платформы разработки и развертывания Java-приложений. Оформление годовой подписки дает право на дополнительную техническую поддержку по продукту WASCE.

1.11 Краткий обзор

Редакция DB2 Express-C является лучшим продуктом в своем классе и распространяется совершенно бесплатно. Этот продукт предоставляет свободу разработки, развертывания и распространения без каких-либо ограничений по размеру базы данных, при этом включая такую же основную функциональность и технологию pureXML, как и прочие редакции DB2. DB2 Express-C поддерживает множество клиентов, драйверов и языков программирования, а также предоставляет удобную возможность обновления до других редакций DB2.

2

Глава 2 — Сопутствующие функции и продукты

В этой главе описаны функции DB2, входящие в состав годовой подписки на лицензию DB2 Express (лицензия с фиксированным сроком действия или FTL). Также описаны функции, доступные в других редакциях DB2, в некоторых случаях — за дополнительную плату.

Различия между бесплатной (безгарантийной) редакцией DB2 Express-C и DB2 Express с годовой подпиской представлены в *Таблице 2.1* ниже.

| Функция | Бесплатная (безгарантийная) редакция | Платная подписка* (FTL-лицензия) |
|--|--|---|
| Основные возможности DB2 | Да | Да |
| Бесплатные инструменты администрирования | Да | Да |
| Бесплатные инструменты разработки | Да | Да |
| Возможности автономной работы | Да | Да |
| Функция «pureXML» | Да | Да |
| Бесплатная поддержка сообщества*** | Да | Да |
| Официальная круглосуточная поддержка компании IBM | Нет | Да |
| Пакеты исправлений | Нет | Да |
| Высокая доступность (HA) | Нет | Да |
| Репликация данных SQL | Нет | Да |
| Сжатие резервных копий | Нет | Да |
| Макс. использование процессора | 2 ядра | 4 ядра (макс. 2 сокета) |
| Макс. использование памяти | 2 ГБ | 4 ГБ |
| Доступность обновлений | Полное обновление при выпуске новых редакций, обычно раз в год | Исправления безопасности и пакеты исправлений несколько раз в год |
| Доступ к установочным образам предыдущих версий/выпусков | Нет, доступны только образы текущего выпуска и бета-версий | Да, посредством IBM Passport Advantage |
| Цена на сервер в год** | 0 | 2995 дол. США |

Таблица 2.1: Сравнение БЕСПЛАТНОЙ редакции DB2 Express-C и платной подписки (FTL-лицензия)

30 Начало работы с DB2 Express-C

* Функции, предусмотренные подпиской, доступны только во время действия подписки.

** Стоимость подписки указана для территории США и может быть изменена без предварительного уведомления. Стоимость может различаться в зависимости от страны.

*** Бесплатной поддержкой сообщества можно воспользоваться на интернет-форуме.

В Таблице 2.2 перечислены функции продукта и их наличие в различных редакциях DB2 9.7. Функции, которые можно приобрести отдельно, указаны для соответствующих редакций DB2 и выделены светло-серым цветом.

| Функция | Платная подписка (FTL) DB2 Express | DB2 Express Edition | DB2 Workgroup Server Edition (WSE) | DB2 Enterprise Server Edition (ESE) |
|--|---|-----------------------------------|---|---|
| Однородная репликация SQL | Да | Да | Да | Да |
| Однородная интеграция | Да | Да | Да | Да |
| Net Search Extender, DB2 Text Search (текстовый поиск) | Да | Да | Да | Да |
| Spatial Extender | Да | Да | Да | Да |
| Сжатие резервных копий | Да | Да | Да | Да |
| Технология «pureXML» | Да | Да | Да | Да |
| Аварийное восстановление высокой доступности (HADR) | Да | Функция высокой доступности | Да | Да |
| Tivoli System Automation | Да | | Да | Да |
| Службы расширенного копирования | Нет | | Да | Да |
| Онлайн- реорганизация | Нет | | Да | Да |

| | | | | |
|---|-----|-----|-----|--|
| MQT – материализованные таблицы запросов | Нет | Нет | Нет | Да |
| MDC – многомерная кластеризация | Нет | Нет | Нет | Да |
| Параллелизм запросов | Нет | Нет | Нет | Да |
| Концентратор соединений | Нет | Нет | Нет | Да |
| Сегментирование таблиц | Нет | Нет | Нет | Да |
| Governor | Нет | Нет | Нет | Да |
| Сжатие: уровень строки, индекса, XML, временной таблицы | Нет | Нет | Нет | Функция оптимизации хранения данных |
| Доступ на основе меток (LBAC) | Нет | Нет | Нет | Функция расширенного контроля доступа |
| Geodetic Extender | Нет | Нет | Нет | Функция управления геодезическим и данными |
| Query Patroller | Нет | Нет | Нет | Функция оптимизации производительности |
| Управление рабочей нагрузкой DB2 | Нет | Нет | Нет | |
| Performance Expert | Нет | Нет | Нет | |
| Однородная Q-репликация | Нет | Нет | Нет | Функция однородной репликации для ESE |
| Разбиение базы данных | Нет | Нет | Нет | Нет |

Таблица 2.2: Редакции DB2 версии 9.7: поддерживаемые функции и возможности

32 Начало работы с DB2 Express-C

В других редакциях DB2 доступны следующие функции:

Платные функции редакции DB2 Express

- Функция высокой доступности (High Availability - HA).

Бесплатные функции редакции DB2 Workgroup:

- Аварийное восстановление высокой доступности (High Availability and Disaster Recovery – HADR), Tivoli System Automation, онлайн-реорганизация, службы расширенного копирования.
- Доступность DB2 на дополнительных платформах UNIX: AIX, Solaris и HP-UX.

Бесплатные функции редакции DB2 Enterprise:

- Сегментирование таблиц (по диапазонам значений)
- MQT MQT - материализованные таблицы запросов
- MDC - многомерная кластеризация
- Параллелизм запросов
- Концентратор соединений
- Governor

Платные функции редакции DB2 Enterprise

- Функция оптимизации хранения данных (включает сжатие)
- Расширенный контроль доступа (тщательная и расширенная защита)
- Оптимизация производительности (Workload Management, Performance Expert, Query Patroller)
- Управление геодезическими данными (анализ географического местоположения)

Платные продукты, связанные с DB2:

- DB2 Connect
- Редакции InfoSphere Warehouse
- InfoSphere Balanced Warehouse
- WebSphere Federation Server
- WebSphere Replication Server

2.1 Функции, включенные в платную подписку (FTL) DB2 Express

В данном разделе описаны пакеты исправлений DB2, а также функции HADR и репликация SQL.

2.1.1 Пакеты исправлений

Пакет исправлений DB2 представляет собой набор исправлений кода, применяемый к установленному продукту DB2 для устранения различных проблем, обнаруженных после выпуска продукта. Если установлена лицензия по подписке, пакеты исправлений можно загрузить и установить бесплатно. Обычно они появляются каждые четыре месяца или в другие оговоренные сроки.

Чтобы загрузить последний пакет исправлений, посетите веб-сайт технической поддержки DB2 по адресу http://www.ibm.com/software/data/db2/support/db2_9/

2.1.2 HADR

Аварийное восстановление высокой доступности (High-Availability Disaster Recovery, HADR) — это функция надежности базы данных, которая обеспечивает высокую доступность и аварийное восстановление, как при частичных, так и при полных местных отказах. Обычно, среда HADR состоит из двух серверов данных, первичного и вспомогательного (которые могут находиться в разных местах). На первичном сервере хранится исходная база данных, доступ к которой получают клиентские приложения. По мере обработки транзакций в первичной базе данных записи журнала базы данных автоматически отправляются на вспомогательный сервер через сеть. На вспомогательном сервере хранится зеркальная копия первичной базы данных, которая обычно создается путем резервного копирования первичной базы данных и её восстановления на вспомогательной системе. Когда журналы первичной базы данных попадают на вспомогательный сервер, они воспроизводятся и применяются к вспомогательной базе данных. Посредством постоянного воспроизведения записей журнала, вспомогательная база данных поддерживает синхронизированную копию первичной базы данных, пригодную для использования в случае отказа первичной.

Решение HADR, полностью поддерживаемое DB2, предоставляет следующие возможности:

- Молниеносное восстановление после отказа с полной прозрачностью для пользователей и клиентских приложений
- Полная атомарность транзакций для предотвращения потери данных
- Обновление систем или приложений без ощутимого прерывания работы
- Удаленное восстановление системы после отказа, обеспечивающее надежную защиту центра обработки данных от возможных локальных аварийных ситуаций
- Простое управление с помощью графических инструментов DB2
- Все вышеперечисленное с минимальным влиянием на общую производительность системы

Примечание.

Чтобы ознакомиться с принципами работы HADR, перейдите по ссылке:

<http://www.ibm.com/software/data/db2/express/demo.html>

34 Начало работы с DB2 Express-C

new in
V9.7

В DB2 версии 9.7 появится возможность разрешить клиентам считывать данные с резервного сервера («read-on-standby»). Предположительно, эта функция будет введена в DB2 9.7 с пакетом исправлений 1.

2.1.3 Репликация данных

Эта функция позволяет осуществлять репликацию данных между исходным сервером, на котором фиксируются изменения данных, и целевым сервером, на котором такие изменения применяются. На *рис. 2.1* представлен обзор принципа работы репликации.

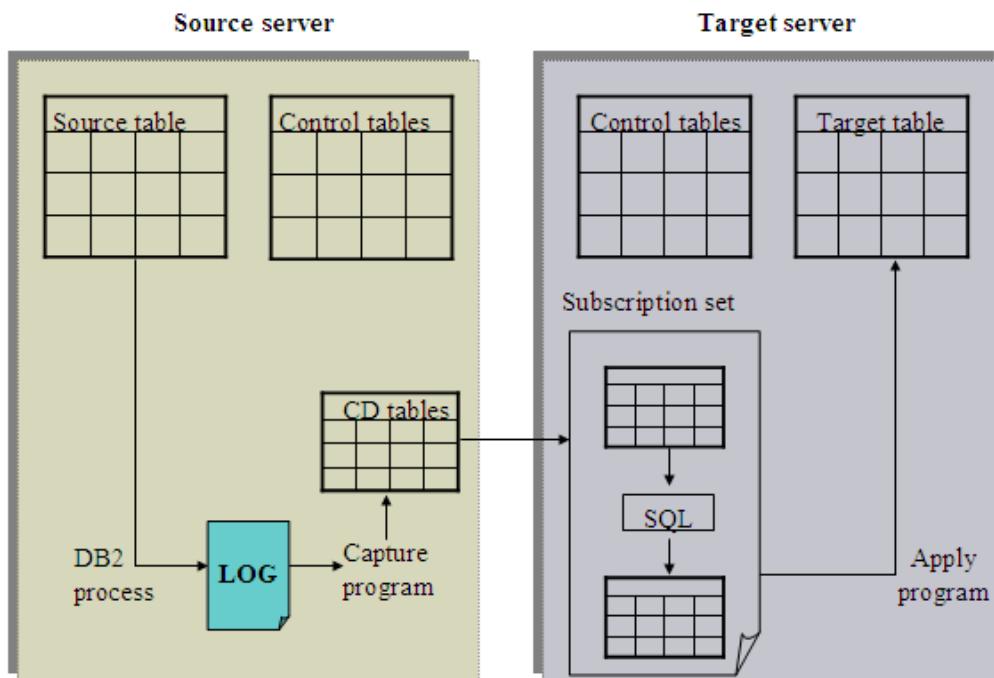


Рисунок 2.1. Репликация SQL

На *рис. 2.1* показаны два сервера — исходный (Source) и целевой (Target). На исходном сервере программа захвата (Capture) получает изменения, внесенные в базу данных. На целевом сервере программа Apply применяет изменения к копии базы данных. Репликация полезна для ряда задач, использующих реплицированные данные, включая разгрузку, наполнение хранилищ данных и витрин данных, а также проверку истории изменений. С помощью функции репликации SQL, можно реплицировать данные между DB2 Express и другими серверами данных компании IBM, в том числе работающих в других системах Linux, UNIX, z/OS и i5/OS.

2.2 Функции, не доступные в DB2 Express-C

В этом разделе описаны некоторые функции, доступные в других редакциях DB2, но отсутствующие в DB2 Express-C или годовой подписке на лицензию DB2 Express.

2.2.1 Сегментирование базы данных

Функция разбиения базы данных (Database Partitioning Feature, DPF) обеспечивает распределенную обработку запросов в кластере серверов данных. Она доступна только в редакциях InfoSphere Warehouse и дает возможность распределить данные в нескольких сегментах или узлах базы данных, расположенных на нескольких различных серверах. Функция DPF основана на архитектуре без разделения ресурсов, в которой каждый сегмент базы данных на собственных независимых дисках хранит отдельное подмножество всех данных.

Каждый подключаемый к кластеру базы данных компьютер добавляет мощность обработки данных, благодаря своим центральным процессорам (ЦП), памяти и дискам, что позволяет разбить крупные задачи и сложные запросы на более мелкие части и распределить их между узлами базы данных для параллельного выполнения. В результате, показатели параллельности использования и времени реакции значительно выше, чем при размещении базы данных на одном сервере. Функция DPF особенно полезна в крупных информационных хранилищах и при рабочей нагрузке бизнес-аналитики, поскольку могут действовать объемы данных от нескольких сотен гигабайт до нескольких сотен терабайт.

2.2.2 Концентратор соединений

Концентратор соединений — это функция, обеспечивающая поддержку большого числа параллельно подключенных пользователей. Раньше для каждого соединения с базой данных требовался отдельный агент базы данных. Концентратор соединений предлагает концепцию «логического агента», благодаря которой один агент может управлять несколькими соединениями. Агенты описаны более подробно в Главе 6 «Архитектура DB2».

2.2.3 Geodetic Extender

Функция DB2 Geodetic Extender доступна в редакции DB2 Enterprise Server за отдельную плату. Это расширение значительно упрощает разработку приложений для бизнес-аналитики и электронного правительства, которым требуется анализ географического местоположения. DB2 Geodetic Extender может сконструировать виртуальный глобус любого масштаба. Основная часть информации о местоположении собирается с помощью навигационных систем, охватывающих весь мир, таких как глобальная навигационная спутниковая система (GPS), и может быть представлена в виде координат широты/долготы (геокода). С помощью функции DB2 Geodetic Extender такие, такие бизнес-данные, как адреса, можно конвертировать в геокоды, а корпоративные приложения работают лучше, если хранят данные в такой «непроекционной» форме, оставляя проекции карт (земли на плоскую поверхность) там, где они и должны быть: на уровне отображения, для просмотра и печати карт.

2.2.4 LBAC

Система управления доступом на уровне меток (Label-based access control, LBAC) предоставляет возможность структурированного управления безопасностью на уровне строк и столбцов. Для предоставления доступа к данным в таблице используются метки, привязанные, как к пользовательским сессиям, так и строкам или столбцам данных. На *рис. 2.2* показан принцип работы LBAC.

36 Начало работы с DB2 Express-C

| No LBAC | SEC=254 | SEC=100 | SEC=50 | ID | SALARY |
|---------|---------|---------|--------|-----|--------|
| | | | | 255 | 60000 |
| | | | | 100 | 50000 |
| | | | | 50 | 70000 |
| | | | | 50 | 45000 |
| | | | | 60 | 30000 |
| | | | | 250 | 56000 |
| | | | | 102 | 82000 |
| | | | | 100 | 54000 |
| | | | | 75 | 33000 |
| | | | | 253 | 46000 |
| | | | | 90 | 83000 |
| | | | | 200 | 78000 |

Рисунок 2.2. Пример работы LBAC

На рисунке в таблице `EMP` имеется столбец `SALARY` (заработка плата) и внутренний столбец `ID`, содержащий метку для данной строки. Остальные столбцы на рисунке используются только в иллюстративных целях. При выполнении показанного на рисунке запроса будут отображаться разные строки в зависимости от имеющейся у пользователя метки. Столбец под заголовком «No LBAC» показывает строки, которые были бы выбраны без применения LBAC. Как видим, выбраны все строки с заработной платой выше или равной 50 000.

Теперь допустим, что пользователь, сделавший запрос, имеет метку 100. Мы видим строки, выбранные в этом случае, в третьем столбце слева. В данном случае, DB2 найдет строки, в которых заработка плата выше или равна 50 000, а затем проверит метки безопасности этих строк. К примеру, в первой строке указаны зарплата 60 000 и идентификатор метки 255. Поскольку пользователь имеет идентификатор метки 100, т. е. меньше чем 255, он не сможет увидеть эту строку, и, соответственно, она не будет включена в результаты выполнения запроса.

Система управления доступом на уровне меток должна внедряться администратором безопасности с полномочиями «SECADM».

2.2.5 Менеджер рабочей нагрузки (Workload Manager, WLM)

WLM управляет рабочими нагрузками в базе данных на основе приоритетов пользователя и приложения с учетом доступности ресурсов и порогов рабочей нагрузки. Менеджер дает возможность регулировать рабочую нагрузку и запросы базы данных, чтобы важные запросы с высоким приоритетом выполнялись в первую очередь, не давая «неконтролируемым» запросам монополизировать системные ресурсы, обеспечивая тем самым эффективную работу системы. В DB2 9.7 менеджер WLM расширен еще больше и предоставляет более мощные возможности, чем инструменты Query Patroller и DB2 Governor, доступные в предыдущих версиях DB2.

2.2.6 Глубокое сжатие

DB2 поддерживает несколько типов сжатия:

- Сжатие пустых значений (NULL) и значений по умолчанию

Этот тип сжатия применяется к столбцам, строки которых обычно имеют значения NULL или системные значения по умолчанию, например «0», которые не требуют дискового пространства

- Многомерная кластеризация (Multi-dimensional Clustering, MDC)

Таблицы MDC — это таблицы, в которых физические страницы данных кластеризованы в нескольких измерениях. Они используют блочные индексы, что является, в некотором роде, способом сжатия индексов, поскольку происходит указание на блок записей, а не на единичную запись.

- Сжатие резервных копий базы данных

Этот способ применяется к образам резервных копий. Сжимаются табличные пространства индексов и данные типа LOB (Large Object – большой объект).

- Сжатие строк данных

Сжатие строк работает по принципу замены повторяющихся последовательностей в строке данных намного меньшим символом. Информация о преобразовании такого меньшего символа хранится в словаре вместе с начальной последовательностью. Сжатие строк может значительно повысить производительность в режимах нагрузки при вводе/выводе, подразумевая, что с диска в память (и наоборот) можно перенести больше строк, поскольку они имеют меньший размер. Еще одним преимуществом является экономия пространства хранилища, что обычно составляет наибольшую часть расходов компаний на информационные технологии. С точки зрения рабочих нагрузок, зависящих от центрального процессора, в результате сжатия строк могут возникнуть некоторые дополнительные расходы, поскольку перед обработкой сжатые строки необходимо восстановить. Обратите внимание еще и на то, что данные журнала, созданные для сжатых записей, также будут храниться в сжатом формате.

При доступе к столбцам XML и данным LOB-типа DB2 обычно не использует буферный пул (память), а выполняет операции ввода/вывода непосредственно на диск. Причиной этого является то, что обычно столбцы XML и данные LOB-типа имеют большой размер; соответственно, их запись в память вытеснила бы из памяти нужные страницы. Однако в DB2 9.5 для небольших XML-документов (меньше 32 КБ) допускается встраивание XML. Иными словами, небольшие XML-документы могут храниться с основными строками таблицы, а не в отдельном хранилище в памяти объекте, известном как XDA. Такой подход имеет два преимущества: во-первых, теперь доступ к XML-документам можно получить через буферный пул; во-вторых, XML-документы также могут быть задействованы в сжатии строк данных.

DB2 9.7 предлагает следующие улучшения сжатия:

- Внутренние объекты XDA (где хранятся XML) теперь также могут быть сжаты.
- Индексы и временные таблицы (системные и пользовательские) могут быть сжаты.
- Данные LOB-типа могут встраиваться подобно XML.

new in
V9.7

2.2.7 Совместимость с SQL

Большинство поставщиков придерживаются стандартов SQL 92 и SQL/PSM, однако в их продуктах поддерживаются не все функции этих стандартов, а с другой стороны — часть новых функций в стандарты не включена. Благодаря представленной в DB2 9.7 функции совместимости SQL теперь кроме собственного синтаксиса SQL PL сервер DB2 поддерживает большинство синтаксических конструкций PL/SQL, поддерживаемых другими поставщиками реляционных СУБД. На рис. 2.3 подытожен принцип такой поддержки.

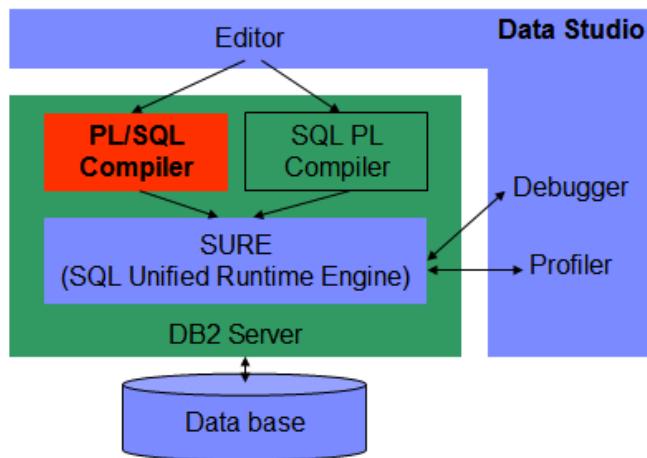


Рисунок 2.3. Поддержка PL/SQL в DB2

На рисунке видно разработанный и встроенный в ядро DB2 компилятор PL/SQL.

Функция совместимости SQL также предусматривает поддержку инструмента CLPPlus. CLPPlus — это инструмент командной строки, позволяющий выполнять SQL и другие команды. Он похож на существующий в DB2 процессор командной строки (Command Line Processor, CLP). На рис. 2.4 представлен инструмент CLPPlus.

```

c:\>clpplus
CLP Plus: Release 1.0
Copyright (c) 2008, IBM CORPORATION. All rights reserved.

SQL> connect srielau@localhost:50000/stmtconc
Enter Password:
Connected to DB2/NT SQL09070 <localhost:50000/stmtconc> AS srielau

SQL> select sysdate from dual;

1
-----
2008-08-02 19:38:34.0
SQL>
  
```

Рисунок 2.4. Инструмент CLPPlus

Также была встроена поддержка большинства типов данных PL/SQL, таких как BINARY_INTEGER, RAW и пр. Другие типы данных Oracle, такие как VARCHAR2,

поддерживаются даже без функции совместимости SQL, но их поддержку необходимо включить с помощью переменной реестра DB2_COMPATIBILITY_VECTOR. Типы данных Oracle и эта переменная реестра рассмотрены более подробно в последующих главах этой книги.

Описанные выше функции совместимости SQL в настоящее время доступны в редакциях DB2 9.7 Workgroup и Enterprise. В ближайшем будущем, планируется их добавление в редакцию DB2 Express (с годовой подпиской на лицензию — FTL).

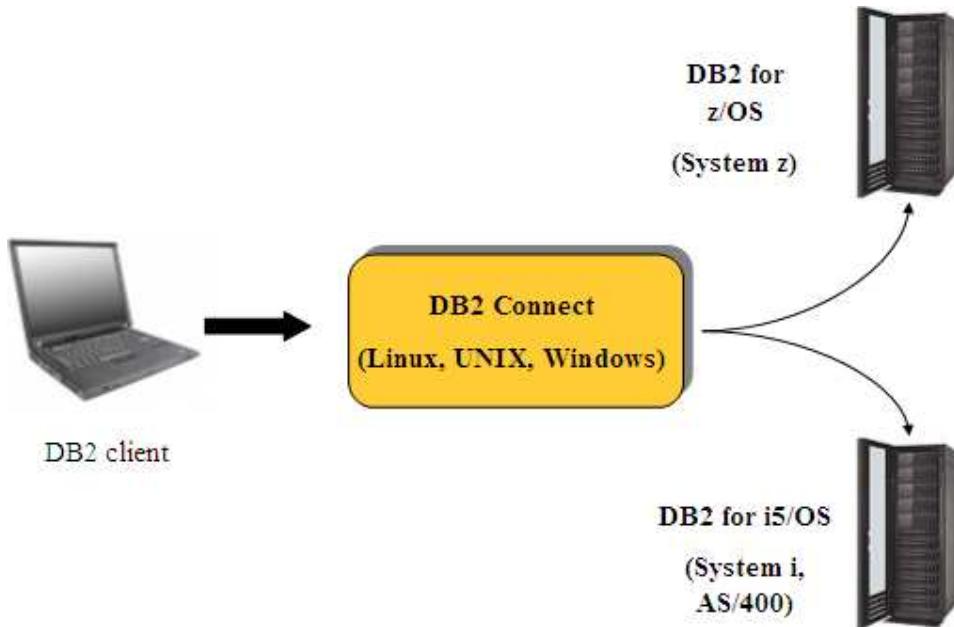
Хотя функции поддержки PL/SQL и CLPPlus не доступны в DB2 Express-C 9.7, другие включенные функции упрощают перевод приложений Oracle на DB2. Среди них — новые типы данных, новые скалярные функции, поддержка модулей и принятой на текущий момент (*currently committed*, CC) семантики для уровня изоляции по установленному курсору (*cursor stability*, CS). Эти функции рассмотрены в последующих главах этой книги.

2.3 Платные продукты, связанные с DB2

В этом разделе кратко описаны платные продукты и предложения, которые можно использовать с DB2.

2.3.1 DB2 Connect

DB2 Connect — это платное программное обеспечение, позволяющее клиенту DB2 для Linux, UNIX или Windows подключаться к серверу DB2 для z/OS или DB2 для i5/OS (см. *рис. 2.5*). DB2 Connect не требуется при подключении в обратном направлении, т. е. при подключении DB2 для z/OS или DB2 для i5/OS к серверу DB2 для Linux, UNIX или Windows. Доступны две основные редакции DB2 Connect в зависимости от потребностей подключения: DB2 Connect Personal Edition и DB2 Connect Enterprise Edition.

**Рисунок 2.5. DB2 Connect**

2.3.2 InfoSphere Federation Server

Ранее известный как WebSphere Information Integrator (для поддержки интеграции), WebSphere Federation Server дает возможность объединить базы данных, в результате чего можно выполнять запросы, поддерживающие объекты других систем реляционных баз данных. К примеру, купив WebSphere Federation Server, можно выполнить запрос, показанный в *Листинге 2.1* ниже.

```
SELECT *
FROM   Oracle.Table1 A
       DB2.Table2 B
       SQLServer.Table3 C
WHERE
      A.col1 < 100
      and B.col5 = 1000
      and C.col2 = 'Test'
```

Листинг 2.1. Федерированный запрос

На *рис. 2.6* проиллюстрированы сферы применения WebSphere Federation Server.

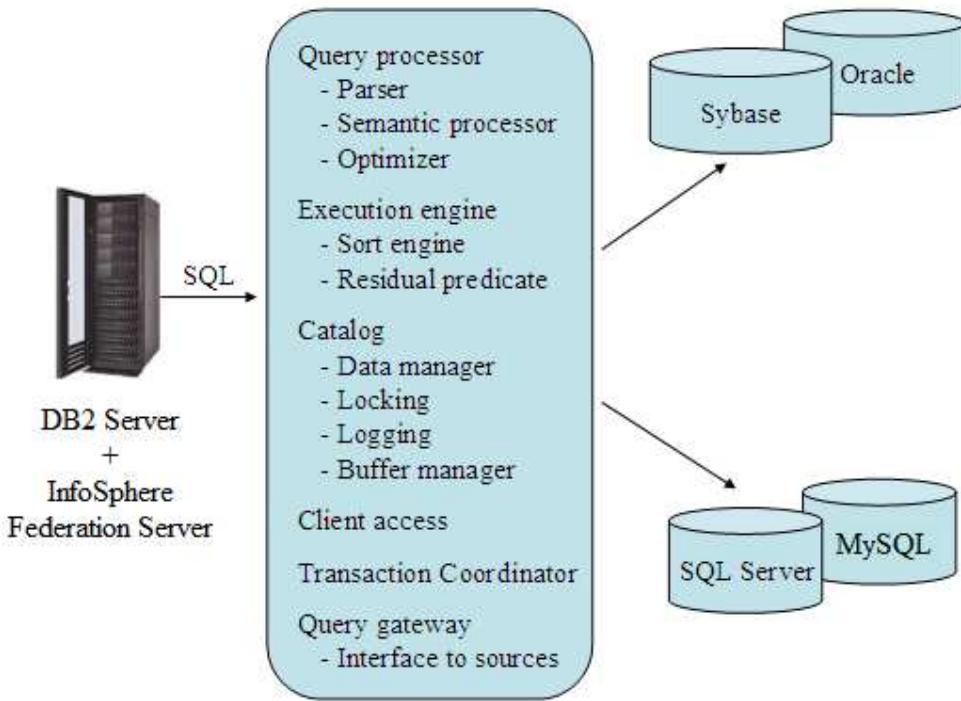


Рисунок 2.6. InfoSphere Federation Server

Поддержка интеграции с системами управления реляционными базами данных, входящими в семейство продуктов IBM, встроена в DB2 Express-C. Иными словами, продукт WebSphere Federation Server не нужен, к примеру, для выполнения запроса между двумя базами данных DB2 или между базами данных DB2 и Informix (Informix входит в семейство продуктов IBM).

2.3.3 InfoSphere Replication Server

Известный ранее как WebSphere Information Integrator (для поддержки репликации), InfoSphere Replication Server дает возможность выполнять SQL-репликацию записей базы данных при использовании серверов данных сторонних разработчиков. Также включена функция, известная как Q-репликация, для репликации данных с помощью очередей сообщений.

2.3.4 Optim Development Studio (ODS)

Известный ранее как Data Studio Developer, ODS — это инструмент на базе платформы Eclipse, который может легко интегрироваться с Data Studio и использовать ту же инсталляцию Eclipse. ODS помогает в создании баз данных для разработки, используя копирование и вставку элементов из существующих баз данных Oracle или DB2.

2.3.5 Optim Database Administrator (ODA)

Известный ранее как Data Studio Administrator, ODA — это инструмент на базе платформы Eclipse, который может легко интегрироваться с Data Studio и использовать ту же инсталляцию Eclipse. ODA предоставляет возможность управления изменениями, а также упрощает автоматизацию изменений схем базы данных.

2.4 Предложения DB2 в Amazon Elastic Compute Cloud

Следует заметить, что компания IBM начала сотрудничество с Amazon Web Services (AWS) для поддержания работы DB2 в веб-службе Amazon Elastic Compute Cloud (EC2). AWS предоставляет комплекс интегрированных служб, которые формируют вычислительную платформу «в облаке» и доступны по принципу оплаты по мере использования. Иными словами, AWS позволяет «арендовать» вычислительные мощности (виртуальные серверы и хранилища), а оплата осуществляется только за реально использованные мощности. К примеру, предположим, что для нормального функционирования базы данных вы используете один виртуальный сервер EC2, а в периоды пиковой или сезонной нагрузки вам на несколько часов требуется дополнительный сервер данных. В таком случае вы заплатите AWS только за дополнительный сервер данных и только за те несколько часов, когда он использовался.

IBM предлагает три варианта развертывания DB2 на облачной платформе Amazon:

- Образ DB2 Express-C AMI (Amazon Machine Image) для оценки и разработки.
- Готовый к работе образ AMI для DB2 Express и DB2 Workgroup с оплатой по мере использования.
- Пользовательский образ AMI, создаваемый с применением имеющихся лицензий DB2.

Чтобы получить более подробную информацию о начале работы с DB2 в Amazon EC2, посетите веб-страницу ibm.com/db2/cloud

2.5 Краткий обзор

DB2 Express-C предоставляет бесплатную, простую в использовании и мощную основу для разработки приложений баз данных, их развертывания в производственном режиме, а также встраивания в решения сторонних разработчиков и распространения в их составе. DB2 Express-C является идеальным вариантом для тех, кого устраивает поддержка сообщества, отсутствие новейших исправлений и расширенных функций. Если же требуется официальная техническая поддержка компании IBM, регулярные обновления программного обеспечения (пакеты исправлений) или поддержка использования дополнительных ресурсов и высокодоступной кластеризации, компания IBM предлагает подписку на лицензию DB2 Express (FTL) за умеренную годовую плату. Тем, кому требуются еще более широкие функциональные возможности для критически важных рабочих задач и масштабных приложений баз данных, компания IBM предлагает более масштабируемые редакции DB2 и соответствующие продукты. Таким образом, можно начать с DB2 Express-C и постепенно расширяться по мере роста потребностей бизнеса.

3

Глава 3. Установка DB2

Установить DB2 довольно просто: для типичной установки достаточно выбрать настройки по умолчанию, и за короткое время сервер DB2 будет готов к работе.

Для начала загрузите образ DB2 Express-C, соответствующий используемой платформе, с веб-сайта DB2 Express-C (www.ibm.com/db2/express).

3.1 Предварительные требования к установке

Сервер данных DB2 Express-C доступен для Linux, Sun Solaris (x64) и Microsoft Windows 2003, XP, Vista и Windows 7. Также доступна бета-версия для Mac OS X. Поддерживаемые архитектуры процессоров: 32-разрядная, 64-разрядная и PowerPC (Linux). Для работы с DB2 на других платформах (например, UNIX) необходимо приобрести одну из других вышеописанных редакций сервера данных. Требования к операционным системам для работы со всеми редакциями DB2 также приведены в этом документе: www.ibm.com/software/data/db2/udb/sysreqs.html

Что касается аппаратного обеспечения, DB2 Express-C можно установить на системы с любым количеством процессорных ядер и объемом памяти, однако для бесплатной версии с безгарантийной лицензией используется не больше двух ядер и 2 ГБ памяти, а для версии с платной подпиской DB2 Express — не больше четырех ядер и 4 ГБ памяти. Системы могут быть физическими или виртуальными, созданными посредством сегментирования или использования ПО виртуальной машины. Конечно, можно работать и на небольших системах, например с одним процессором и 1 ГБ памяти.

С наиболее актуальной информацией о требованиях DB2 Express-C к аппаратному обеспечению можно ознакомиться на веб-странице DB2 Express-C по адресу www.ibm.com/software/data/db2/express/about.html

3.2 Права на установку в операционной системе

Для установки DB2 Express-C в операционных системах Linux или Windows необходимо иметь соответствующие права пользователя.

В **Linux** для установки DB2 Express-C нужны права «root» (суперпользователя). Можно установить DB2 Express-C, не имея прав «root», но в таком случае функциональность продукта будет ограничена. Например, при установке без прав «root» невозможно создать больше экземпляров, чем значение по умолчанию, указанное во время установки.

В **Windows** учетная запись пользователя должна принадлежать к группе администраторов компьютера, на который выполняется установка. Кроме того, в Windows 2008, Windows Vista и более новых версиях установку может выполнить пользователь без прав администратора, однако мастер установки DB2 запросит административные учетные данные.

Идентификатор пользователя, выполняющего установку, должен принадлежать к группе администраторов домена, если для установки требуется создание или проверка учетной записи домена.

Для выполнения установки также можно воспользоваться встроенной локальной системной учетной записью, хотя это не рекомендуется. Локальная системная учетная запись не запрашивает пароль, но при этом, не имеет доступа к сетевым ресурсам.

Учетная запись пользователя также должна иметь право на «Доступ к компьютеру из сети».

Примечание.

Ознакомьтесь с видеопрезентацией об установке DB2 Express-C по этой ссылке. Хотя в данной презентации речь идет о DB2 9.5, процесс установки DB2 9.7 отличается только цветом панелей окна установки:

<http://www.channeldb2.com/video/video/show?id=807741:Video:4442>

3.3 Мастер установки

Есть несколько способов установки DB2 Express-C, но проще всего воспользоваться мастером установки DB2 на основе графического интерфейса пользователя. Загрузив и распаковав образ DB2 Express-C, запустите мастер установки следующим образом:

- Windows: Выполните файл `setup.exe`, расположенный в каталоге EXP/image/
- Linux: Выполните команду `db2setup`, расположенную в каталоге exp/disk1/

DB2 Express-C можно легко установить, придерживаясь инструкций мастера установки DB2. В большинстве случаев, настроек по умолчанию вполне достаточно, необходимо всего лишь принять условия лицензии; нажмите кнопку *Next* (Далее) до появления кнопки «Finish», а затем нажмите кнопку *Finish* (Готово). Через несколько минут установка завершится, и сервер данных DB2 будет готов к работе.

На рис. 3.1 показана панель запуска установки DB2. Нажмите *Install a Product* (Установить продукт) и выберите *Install New* (Новая установка), чтобы установить в системе новую копию DB2 Express-C. Если DB2 Express-C или другие редакции DB2 устанавливались ранее, может отображаться кнопка «*Work with Existing*» (Работать с существующими). DB2 позволяет установить несколько разных версий или выпусков продукта.

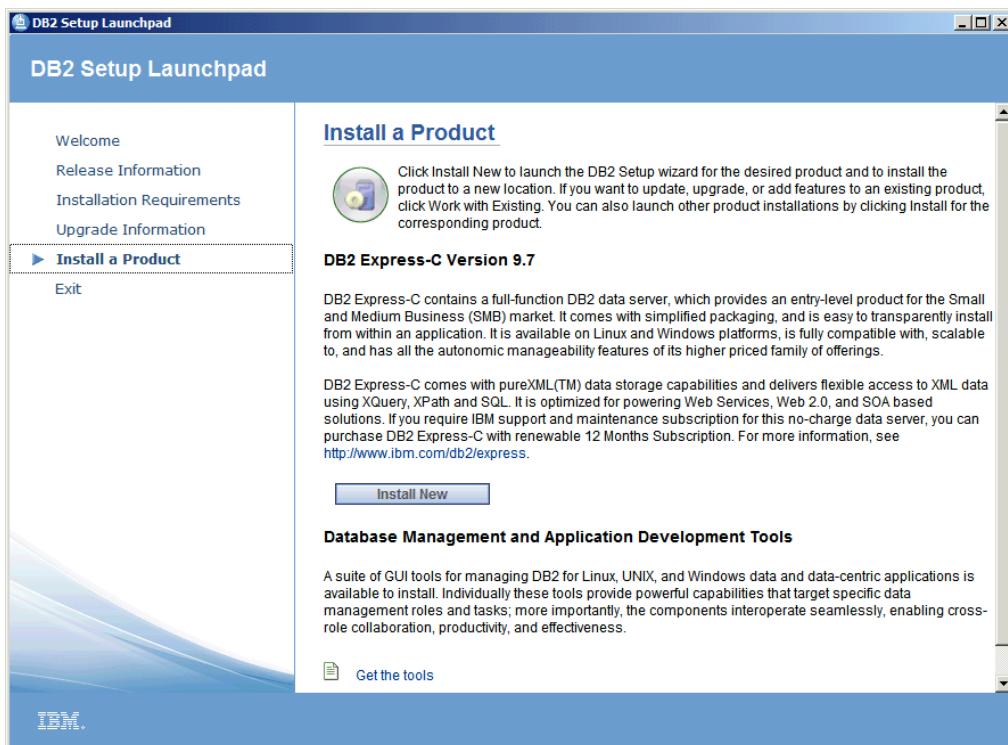


Рисунок 3.1. Панель запуска установки DB2

46 Начало работы с DB2 Express-C

Обычно, после принятия условий лицензии достаточно выбрать вариант установки «*Typical*» (Стандартная, по умолчанию), как показано на рис. 3.2. Если необходимо добавить компонент текстового поиска DB2, выберите вариант установки «*Custom*» (Пользовательская).

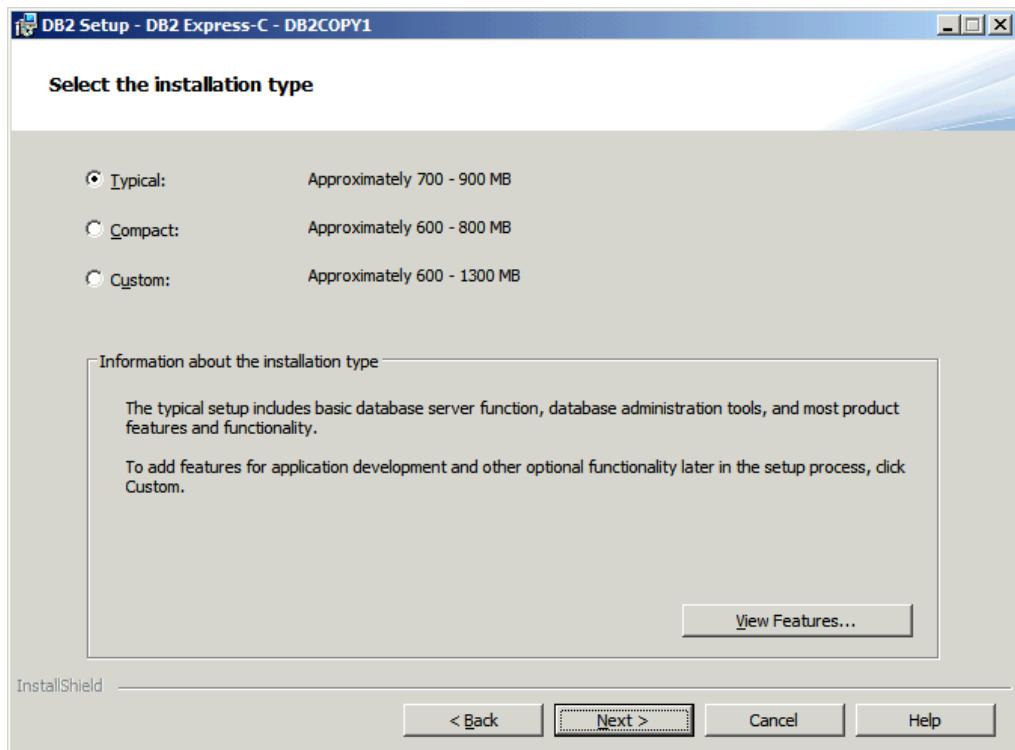


Рисунок 3.2. Типы установки

При выполнении следующего шага, как показано на рис. 3.3, можно установить продукт, создать файл ответов или же выполнить оба действия. Файлы ответов описаны в разделе 3.4 «Автоматическая установка». По умолчанию используется вариант «*Install IBM DB2 Express Edition on this computer and save my settings in a response file*» (Установить IBM DB2 Express Edition на компьютер и сохранить текущие настройки в файле ответов).

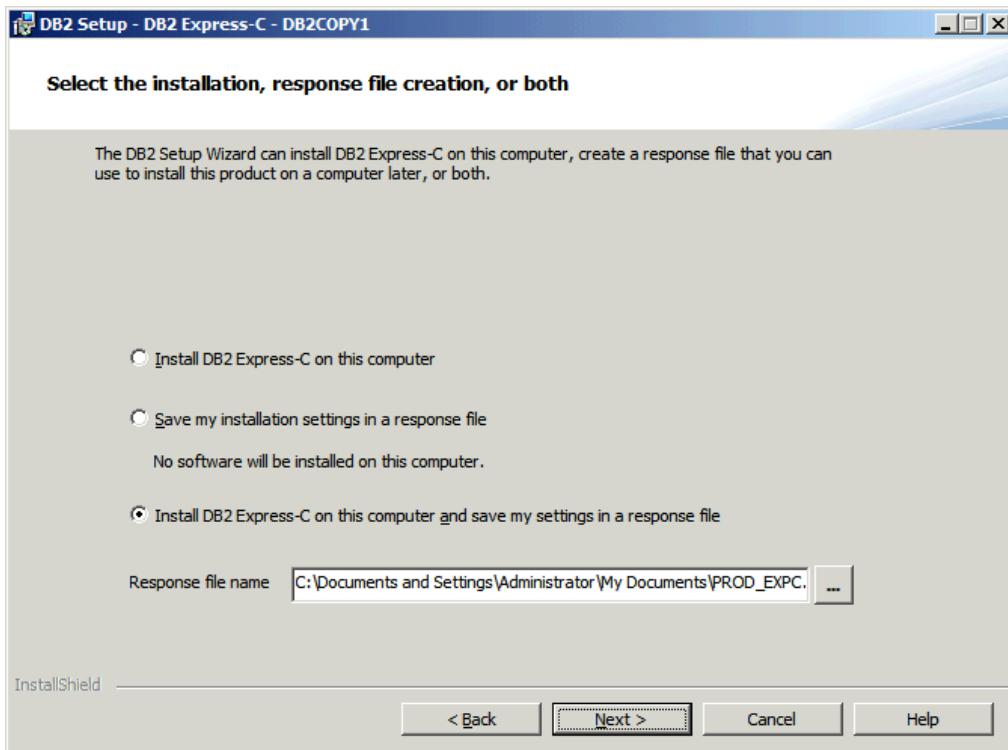


Рисунок 3.3. Выбор установки

48 Начало работы с DB2 Express-C

Выберите значения по умолчанию в следующих нескольких окнах. Когда появится панель, показанная на *рис. 3.4*, нужно ввести идентификатор пользователя, который будет использоваться для настройки и запуска текущего экземпляра и прочих служб.

Если используется идентификатор существующего пользователя, этот пользователь должен входить в группу локальных администраторов Windows.

Если идентификатор не принадлежит существующему пользователю, новый пользователь будет создан как локальный администратор. Если идентификатор пользователя не принадлежит к домену, поле домена можно оставить пустым.

По умолчанию в Windows используется имя идентификатора пользователя db2admin. В Linux по умолчанию создается идентификатор пользователя с именем db2inst1.

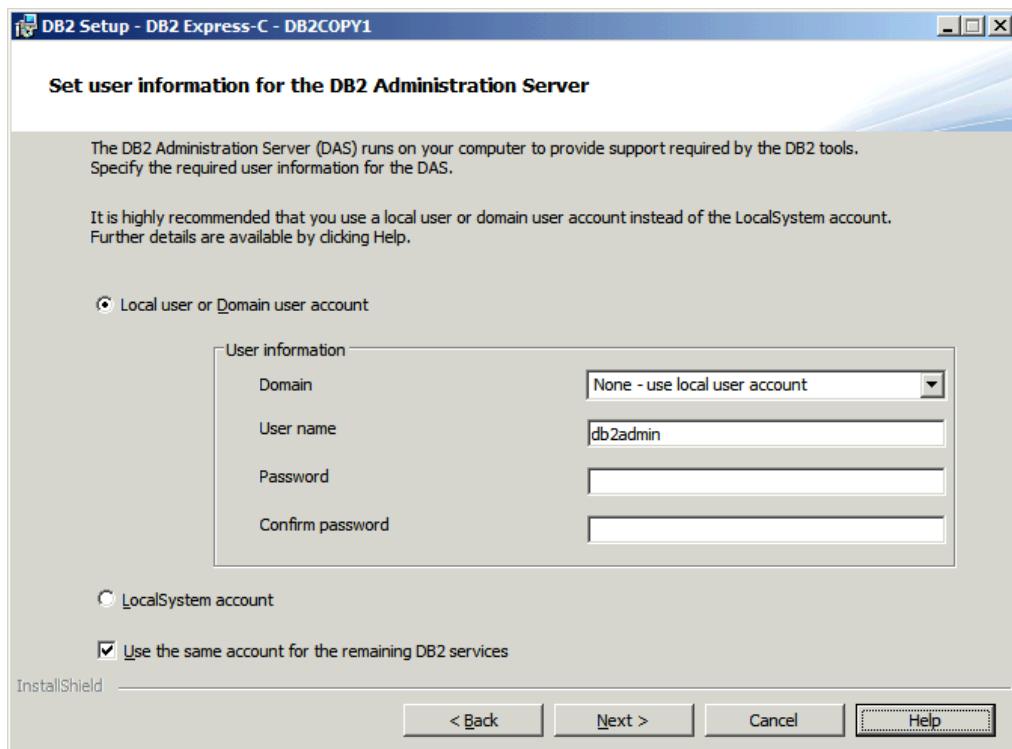


Рисунок 3.4. Определение информации о пользователе для сервера администрирования DB2

Наконец, как показано на *рис. 3.5*, мастер установки отобразит итоговую информацию о предстоящей установке и разнообразные параметры конфигурации, заданные на предыдущих шагах. После нажатия кнопки *Finish* (Готово) начнется установка, и программные файлы будут размещены в системе.

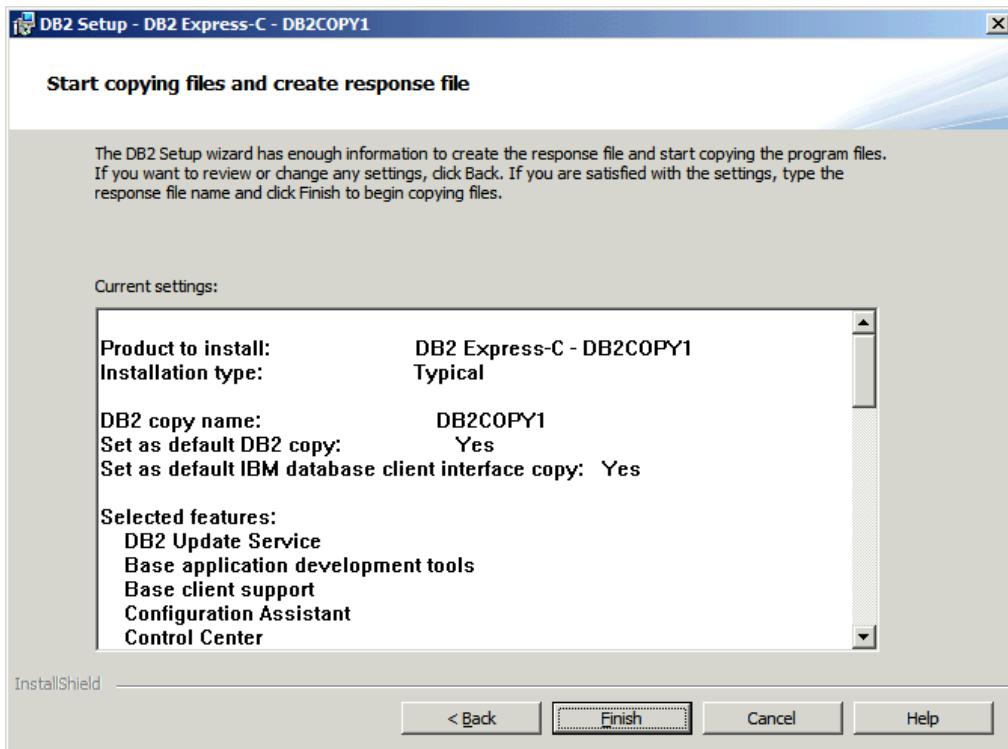


Рисунок 3.5. Итоговая информация о предстоящей установке

После завершения установки откроется окно, показанное на *рис. 3.6*, с информацией о результатах процесса установки и дальнейшими шагами, необходимыми для её завершения.

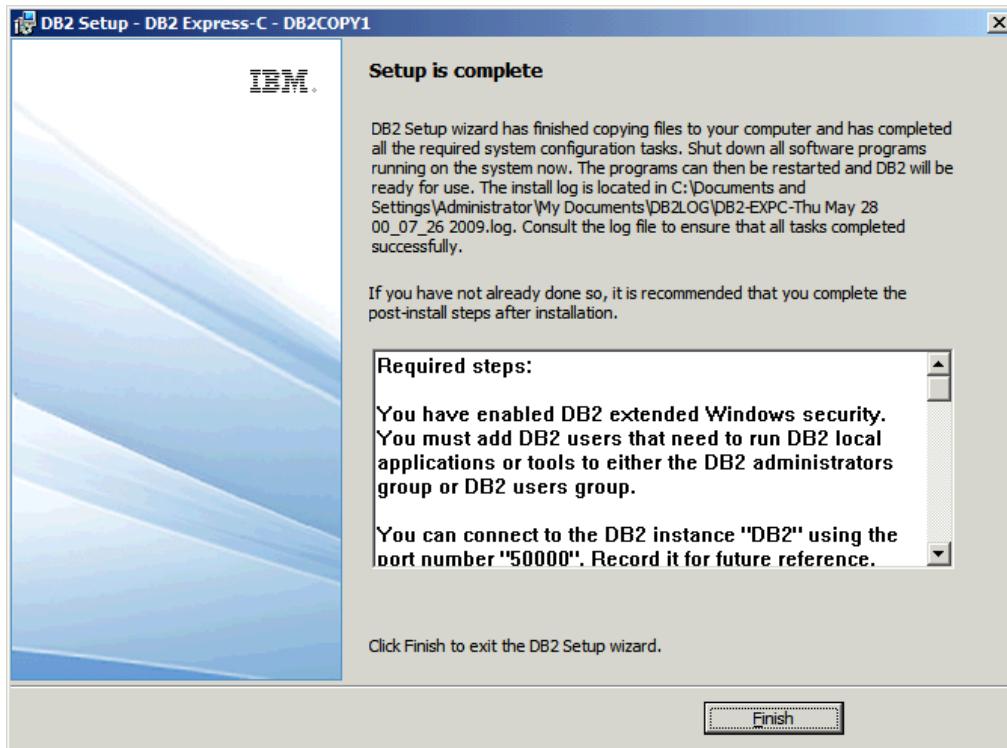


Рисунок 3.6. Установка завершена

После того, как вы нажмёте кнопку *Finish* (Готово) в окне результатов установки, показанном на рис. 3.6, запустится приложение *DB2 Первые шаги* (см. рис. 3.7).

Это небольшое приложение указывает несколько вариантов начала работы в DB2, среди которых — создание образца базы данных по умолчанию (с соответствующим именем `SAMPLE`) или создание собственной базы данных. Если вы не хотите исследовать возможности DB2 с помощью приложения *Первые шаги*, можете закрыть это окно и открыть его в любое удобное время позже.

Чтобы вручную запустить *Первые шаги DB2* в Windows, выберите *Пуск -> Все программы -> IBM DB2 -> DB2COPY1 (Default) -> Set-up Tools -> First Steps* (DB2COPY1 (по умолчанию) -> Инструменты настройки -> Первые шаги) или выполните команду `db2fs` в командной строке.

В Linux выполните команду `db2fs` в окне терминала.

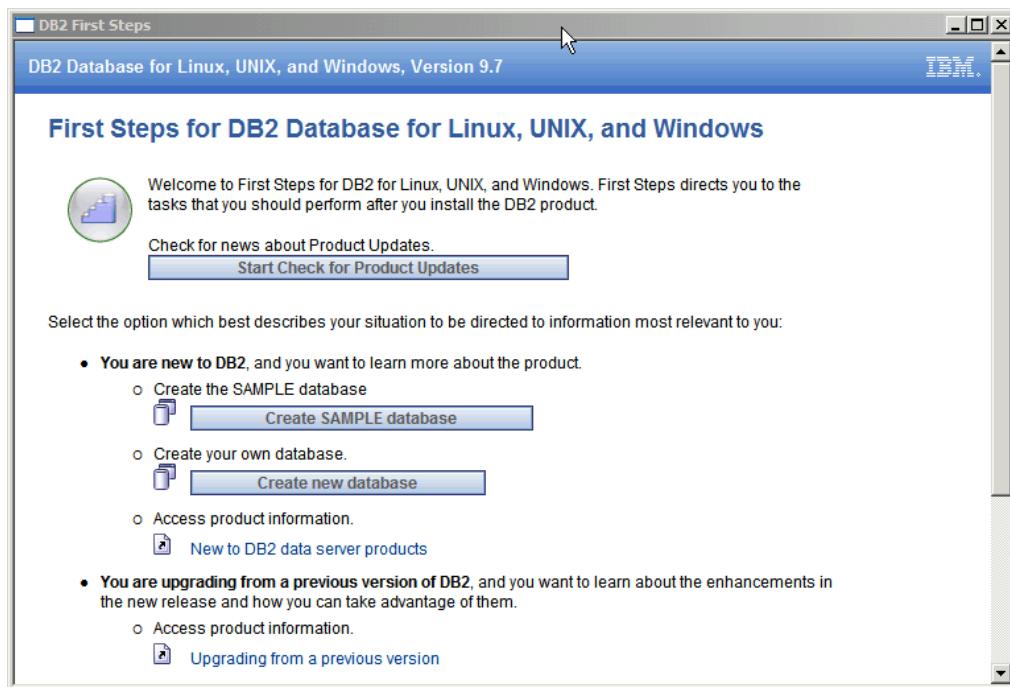


Рисунок 3.7. Первые шаги

3.4 Проверка правильности установки

После установки DB2 можно выполнить три команды в командном окне DB2 (для Windows) или в терминале (для Linux) для проверки правильности установки:

- `db2level`: эта команда отображает информацию об установленном продукте DB2, уровень пакета исправлений и прочие сведения;
- `db2licm -l`: эта команда выводит перечень всех лицензий, относящихся к установленным продуктам DB2;

**new in
V9.7**

- **db2val:** это новая команда, доступна в DB2 9.7. Она подтверждает установку, проверяя основную функциональность установленной копии DB2. Эта команда проверяет согласованность всех экземпляров, а также исправность инструментов создания и подключения баз данных.

На рис. 3.8 ниже приведен пример результатов выполнения этих трех команд.

```

C:\Program Files\IBM\SQLLIB\BIN>db2level
DB210851 Instance "DB2" uses "32" bits and DB2 code release "SQL09070" with
level identifier "08010107".
Informational tokens are "DB2 v9.7.0.441", "s090521", "NT3297", and Fix Pack
"0".
Product is installed at "C:\PROGRA~1\IBM\SQLLIB" with DB2 Copy Name "DB2COPY1"

C:\Program Files\IBM\SQLLIB\BIN>db2licm -l
Product name: "DB2 Express-C"
License type: "Unwarranted"
Expiry date: "Permanent"
Product identifier: "db2expc"
Version information: "9.7"
Max number of CPUs: "2"
Max amount of memory (GB): "2"

C:\Program Files\IBM\SQLLIB\BIN>db2val
DBI1379I The db2val command is running. This can take several minutes.
DBI1333I Installation file validation for the DB2 copy DB2COPY1
was successful.
DBI1339I The instance validation for the instance DB2 was
successful.
DBI1343I The db2val command completed successfully. For details, see
the log file C:\DOCUME~1\ADMINI~1\MYDOCU~1\DB2LOG\db2val-Fri May 29 04_1
40 2009.log.
C:\Program Files\IBM\SQLLIB\BIN>

```

Рисунок 3.8. Команды db2level, db2licm и db2val для подтверждения правильности установки

На рисунке результаты команды **db2level** показывают, что используется DB2 9.7 (DB2 v9.7.0.441) с пакетом исправлений 0, т. е. код сервера DB2 установлен на начальном уровне (GA) без применения исправлений. Команда **db2licm -l** показывает, что установлена редакция DB2 Express-C с постоянной безгарантийной лицензией, которая поддерживает использование до 2-х ядер и до 2 ГБ памяти. Результаты команды **db2val** говорят сами за себя.

Примечание.

Чтобы в любое время проверить непротиворечивость базы данных, воспользуйтесь служебной программой **INSPECT**.

3.5 Автоматическая установка

Бывают ситуации, когда нужно установить клиент DB2 на несколько компьютеров или же необходимо встроить сервер данных DB2 в разрабатываемое приложение и

устанавливать его в процессе установки такого приложения. В таких случаях, лучше всего использовать автоматическую установку DB2.

DB2 может устанавливаться автоматически благодаря использованию файлов ответов, в которых установочная информация хранится как обычные текстовые опции. В *Листинге 3.1* показана часть образца файла ответов.

```
PROD=UDB_EXPRESS_EDITION
LIC AGREEMENT=ACCEPT
FILE=C:\Program Files\IBM\SQLLIB\
INSTALL_TYPE=TYPICAL
LANG=EN
INSTANCE=DB2
DB2 .NAME=DB2
DEFAULT_INSTANCE=DB2
DB2 .SVCENAME=db2c_DB2
DB2 .DB2COMM=TCPIP
...
```

Листинг 3.1. Образец файла ответов

Существует несколько способов создания файла ответов:

- Установите DB2 Express-C на компьютер один раз с помощью мастера установки DB2. В одном из первых окон мастера (см. *рис. 3.3*) можно установить флажок, чтобы сохранить свои ответы в процессе установки в файл ответов. По окончании установки мастер создаст файл ответов с заданным именем в указанном каталоге. Это текстовый файл, и позже его можно отредактировать вручную.
- Отредактируйте образец файла ответов, предоставляемый с образом DB2 Express-C. Этот образец файла (обозначенный расширением .rsp) расположен в каталоге db2/platform/samples/.
- В Windows можно воспользоваться командой создания файлов ответов:

```
db2rspgn -d <целевой каталог>
```

Наконец, для автоматической установки DB2 с помощью файла ответов в Windows выполните следующую команду:

```
setup -u <имя файла ответов>
```

В Linux выполните команду:

```
db2setup -r <имя файла ответов>
```

3.6 Краткий обзор

В этой главе рассмотрена подробная информация об установке DB2 Express-C. Эта редакция DB2 доступна для Linux, Solaris и большинства версий Windows, а также может работать на 32- и 64-разрядных системах и архитектуре Power PC. Обсудив права пользователя, необходимые для установки DB2 в системе, мы рассмотрели линейный процесс установки с помощью графического интерфейса пользователя мастера установки DB2. Затем были рассмотрены действия, необходимые после установки, в том числе запуск приложения DB2 Первые шаги и подтверждение установки. В завершение, мы рассмотрели процесс создания и использования автоматической установки с помощью файлов ответов DB2.

3.7 Упражнения

В этом упражнении нужно установить DB2 Express-C и создать базу данных SAMPLE.

Цель

Прежде, чем приступать к изучению всех функций и инструментов, необходимо установить DB2 Express-C в своей системе. В этом упражнении будет выполнена базовая установка DB2 Express-C в Windows. Такой же мастер установки доступен в Linux; соответственно, установка на этой платформе выполняется аналогично.

Процедура

1. Получите образы DB2 Express-C. Загрузите соответствующий образ DB2 Express-C с веб-сайта DB2 Express-C (www.ibm.com/db2/express). Распакуйте файлы в любой удобный каталог.
2. Найдите файлы. Перейдите к каталогу (или диску), в который были распакованы файлы установки продукта DB2.
3. Откройте панель запуска установки DB2. Откройте панель запуска установки DB2, дважды щелкнув файл `setup.exe`. В Linux выполните команду `db2setup` от имени «root». В левой части панели запуска нажмите *Install Product* (Установить продукт).
4. Воспользуйтесь мастером установки DB2. Мастер установки DB2 проверяет соответствие системы всем требованиям и выполняет поиск установленных версий DB2. Нажмите *Install New* (Новая установка), чтобы запустить мастер, а затем нажмите *Next* (Далее).
5. Ознакомьтесь с лицензионным соглашением. Прочтите и примите условия лицензионного соглашения (выберите пункт «*I Accept...*» (Я принимаю...)) и нажмите кнопку *Next* (Далее) для продолжения.
6. Выберите тип установки. Для этого упражнения выберите вариант установки *Typical* (Стандартная) — этот вариант выбран по умолчанию. Вариант *Compact* (Компактная) выполняет базовую установку, а *Custom* (Пользовательская) позволяет выбрать, какие функции будут установлены. Нажмите кнопку *Next* (Далее), чтобы продолжить.

-
7. Выберите установку, создание файла ответов или оба действия. Оставьте выбранный по умолчанию вариант — будет установлен продукт DB2 и создан файл ответов. Нажмите кнопку *Next* (Далее), чтобы продолжить.
 8. Выберите каталог для установки. В этом окне можно выбрать диск и каталог в системе для установки программного кода DB2. Убедитесь в наличии достаточного объема свободного пространства. Для этого примера используйте настройки диска и каталога по умолчанию (см. ниже):

Диск: С:

Каталог: C:\Program Files\IBM\SQLLIB

Нажмите кнопку *Next* (Далее), чтобы продолжить.

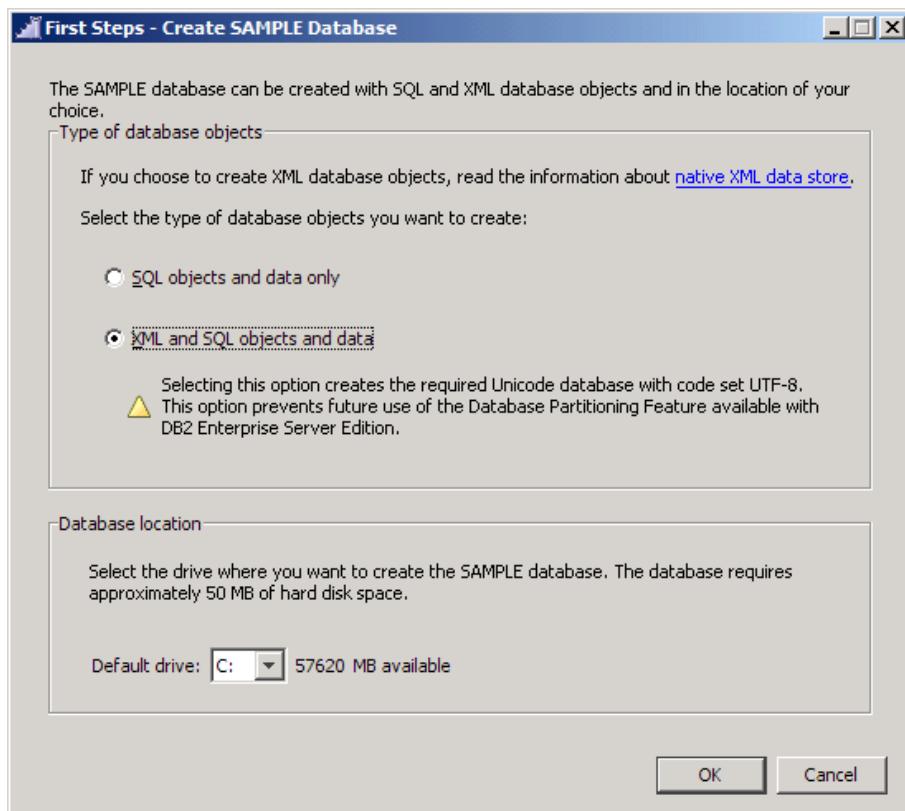
9. Укажите информацию о пользователе. После установки DB2 Express-C определенные процессы DB2 выполняются как системные службы. Для работы этих служб требуется учетная запись операционной системы. В среде Windows рекомендуется использовать учетную запись `db2admin` по умолчанию. Если учетная запись пользователя еще не существует, DB2 создаст её в операционной системе. Также можно указать существующую учетную запись, но она должна иметь полномочия локального администратора. Рекомендуем использовать предложенные по умолчанию варианты. Обязательно укажите пароль учетной записи. В Linux используйте выбранный по умолчанию идентификатор пользователя `db2inst1` для владельца экземпляра, `db2fenc1` — для защищенного пользователя и `dasusr1` — для пользователя сервера администрирования DB2. Нажмите кнопку *Next* (Далее), чтобы продолжить.
10. Сконфигурируйте экземпляр DB2. Экземпляр DB2 можно рассматривать как контейнер для баз данных. Экземпляр должен существовать до того, как в нем будет создаваться база данных. При установке в Windows автоматически создается экземпляр `DB2`. В среде Linux по умолчанию создается экземпляр `db2inst1`. Экземпляры рассматриваются в последующих разделах этой книги.

По умолчанию экземпляр DB2 настроен на прием соединений TCP/IP через порт 50000. Используемый по умолчанию протокол и порт можно изменить, нажав кнопку *Configure* (Конфигурировать). В данном примере рекомендуем использовать настройки по умолчанию. Нажмите кнопку *Next* (Далее), чтобы продолжить.

11. Начните установку. Проверьте итоговую информацию о ранее выбранных параметрах установки. Нажмите кнопку *Finish* (Готово), чтобы начать копирование файлов в установочный каталог. DB2 также выполнит некоторые процессы начального конфигурирования.
12. Первые шаги. После завершения установки отобразится утилита запуска «Первые шаги». Приложение «Первые шаги» можно запустить позже с помощью команды `db2fs`.
13. База данных `SAMPLE` — это база данных, которую можно использовать для тестирования возможностей продукта. Её можно создать в приложении «Первые шаги», нажав кнопку *Создать базу данных SAMPLE*. В результате

56 Начало работы с DB2 Express-C

нажатия этой кнопки откроется окно, показанное ниже. Выберите второй вариант «XML and SQL objects and data» (Данные и объекты XML и SQL). Базу данных SAMPLE также можно создать с помощью команды `db2sampl -xml -sql`.



14. Через несколько минут можно проверить, создана ли база данных. Откройте инструмент Центр управления DB2, для этого выберите: *Пуск -> Программы -> IBM DB2 -> DB2COPY1 (Default) -> General Administration Tools -> Control Center (DB2COPY1 (по умолчанию))* -> Общие инструменты управления -> Центр управления. Центр управления также можно запустить с помощью команды `db2cc`. При первом запуске Центра управления во всплывающем окне необходимо выбрать вид Центра управления. Оставьте значение по умолчанию (Расширенный) и нажмите **OK**. На левой панели разверните папку *Все базы данных*. Если в этой папке нет базы данных SAMPLE, обновите окно, выбрав *Вид -> Обновить*.
15. Перезагрузите компьютер. Этот шаг не является обязательным. Хотя этот шаг не упоминается в официальной установочной документации DB2, рекомендуем перезагрузить систему (по возможности, хотя бы Windows), чтобы обеспечить успешный запуск всех процессов и правильно очистить ресурсы памяти.

16. Проверьте правильность установки DB2, выполнив следующие команды: **db2level**, **db2licm** и **db2val**. В Windows в меню Пуск откройте командное окно DB2: *Пуск -> Все программы -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Window (DB2COPY1 (по умолчанию) -> Инструменты командной строки -> Командное окно)*. В командном окне (или командном процессоре в Linux) введите **db2level** и проверьте результат. Повторите эти действия для команды **db2licm -l**. Затем выполните команду **db2val**. Успешное завершение команды **db2val** свидетельствует о том, что установка выполнена правильно. Если возникли ошибки, просмотрите файл журнала, указанный в сообщении об ошибке, чтобы получить более подробную информацию. Результаты выполнения этих трех команд должны соответствовать данным, показанным на рис. 3.8 выше.

4

Глава 4. Среда DB2

Среда DB2 включает разные объекты базы данных и файлы конфигурации. На рис. 4.1 представлен обзор различных команд и инструментов для работы с DB2, а в правой части выделена среда DB2. Именно эта часть рисунка рассматривается в данной главе. В левой части рисунка показаны различные команды DB2, а также операторы SQL, SQL/XML и XQuery, используемые для взаимодействия с сервером данных DB2. В центральной части рисунка отображены имена различных инструментов, используемых для взаимодействия с сервером данных DB2.

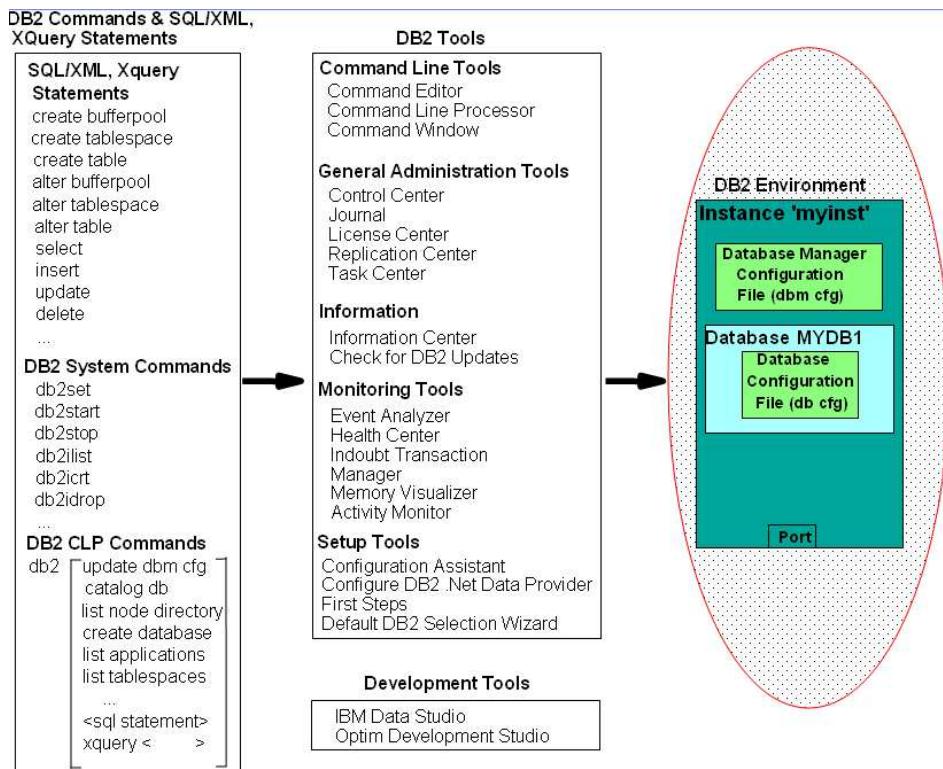


Рисунок 4.1. Общий обзор DB2: среда DB2

Примечание.

Чтобы просмотреть видеопрезентации о среде DB2, перейдите по этим ссылкам:
<http://www.channeldb2.com/video/video/show?id=807741:Video:4029>
<http://www.channeldb2.com/video/video/show?id=807741:Video:4042>

60 Начало работы с DB2 Express-C

Чтобы описать среду DB2, для начала рассмотрим каждый составляющий компонент отдельно. На рис. 4.2 представлен сервер данных DB2 после установки DB2 Express-C 9.7.

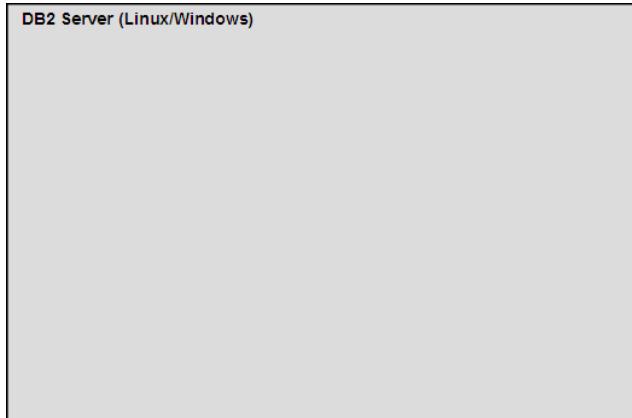


Рисунок 4.2. Представление сервера DB2 после установки DB2 Express-C 9.7

В процессе установки в Windows по умолчанию создается экземпляр с именем DB2 (db2inst1 в Linux). На рис. 4.3 ему соответствует зеленый блок слева. Экземпляр — это просто независимая среда, в которой могут работать приложения и создаваться базы данных. На сервере данных можно создать несколько экземпляров и использовать их в разных целях. К примеру, один экземпляр можно использовать для хранения производственных баз данных, другой — баз данных среды тестирования, еще один может использоваться в качестве среды разработки. Каждый экземпляр является независимым, т. е. выполнение операций в одном из них не влияет на остальные.

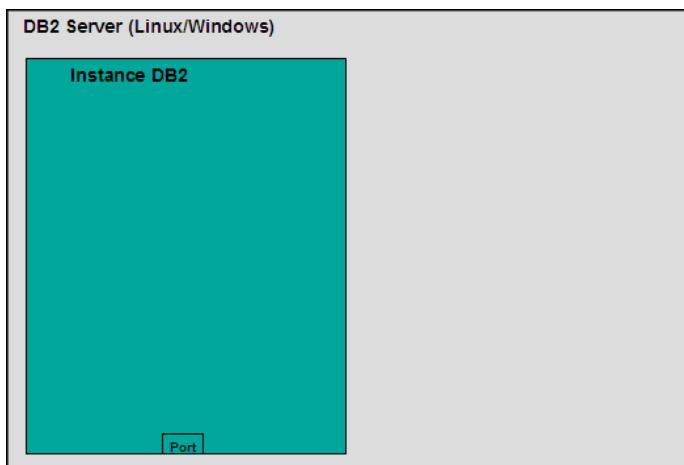


Рисунок 4.3. Экземпляр DB2, созданный по умолчанию

Для создания нового экземпляра DB2 воспользуйтесь командой `db2icrt <имя экземпляра>`, заменив `<имя экземпляра>` любым именем из восьми символов. К примеру, для создания экземпляра `myinst` используется команда `db2icrt myinst`.

На рис. 4.4 новый экземпляр с именем `myinst` показан как отдельный зеленый блок справа.

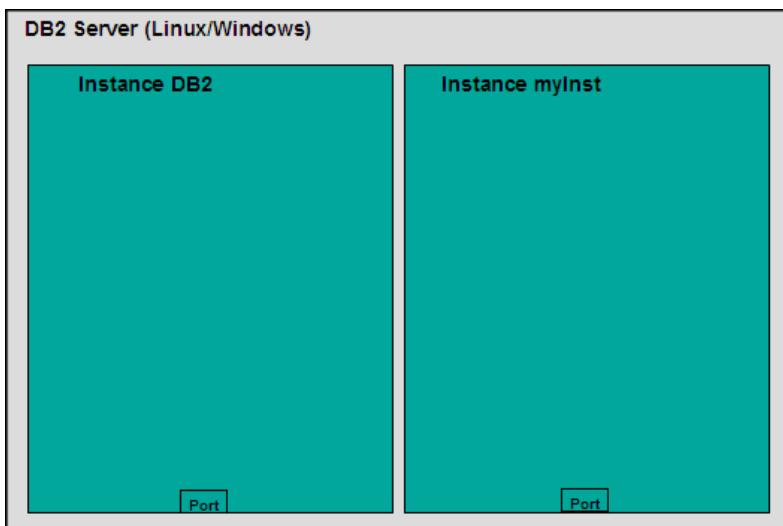


Рисунок 4.4. Сервер DB2 с двумя экземплярами

Обратите внимание, что каждый экземпляр имеет уникальный номер порта. Это помогает различать экземпляры, если необходимо подключиться к определенному экземпляру базы данных с удаленного клиента по TCP/IP. Если используется командное окно DB2, можно любой из экземпляров DB2 сделать активным, выполнив следующую команду операционной системы Windows:

```
set db2instance=myinst
```

Обратите внимание на необходимое отсутствие пробелов до и после знака равенства (`=`). Если теперь в этом примере создать базу данных в командном окне, она будет создана в экземпляре `myinst`.

Чтобы вывести список всех экземпляров системы, выполните команду:

```
db2ilist
```

В Linux имя экземпляра должно совпадать с именем пользователя операционной системы; соответственно, чтобы перейти к другому экземпляру, достаточно сменить пользователя. Такой пользователь является владельцем экземпляра. Можно выйти из системы и войти в нее под именем владельца соответствующего экземпляра или же использовать команду `su`.

В Таблице 4.1 приведены некоторые полезные команды уровня экземпляров.

62 Начало работы с DB2 Express-C

| Команда | Описание |
|-------------------------------|---------------------------------------|
| <code>db2start</code> | Запустить текущий экземпляр |
| <code>db2stop</code> | Остановить текущий экземпляр |
| <code>db2icrt</code> | Создать новый экземпляр |
| <code>db2idrop</code> | Удалить экземпляр |
| <code>db2ilist</code> | Показать список экземпляров в системе |
| <code>db2 get instance</code> | Показать текущий активный экземпляр |

Таблица 4.1. Полезные команды DB2 на уровне экземпляров

Некоторые из перечисленных выше команд можно также выполнить через Центр управления. К примеру, если в Центре управления раскрыть папку Экземпляры и щелкнуть правой кнопкой мыши по требуемому экземпляру, можно выбрать пункт Запуск, что соответствует выполнению команды `db2start` в командном окне DB2, или Остановить, что соответствует команде `db2stop` (см. рис. 4.5).

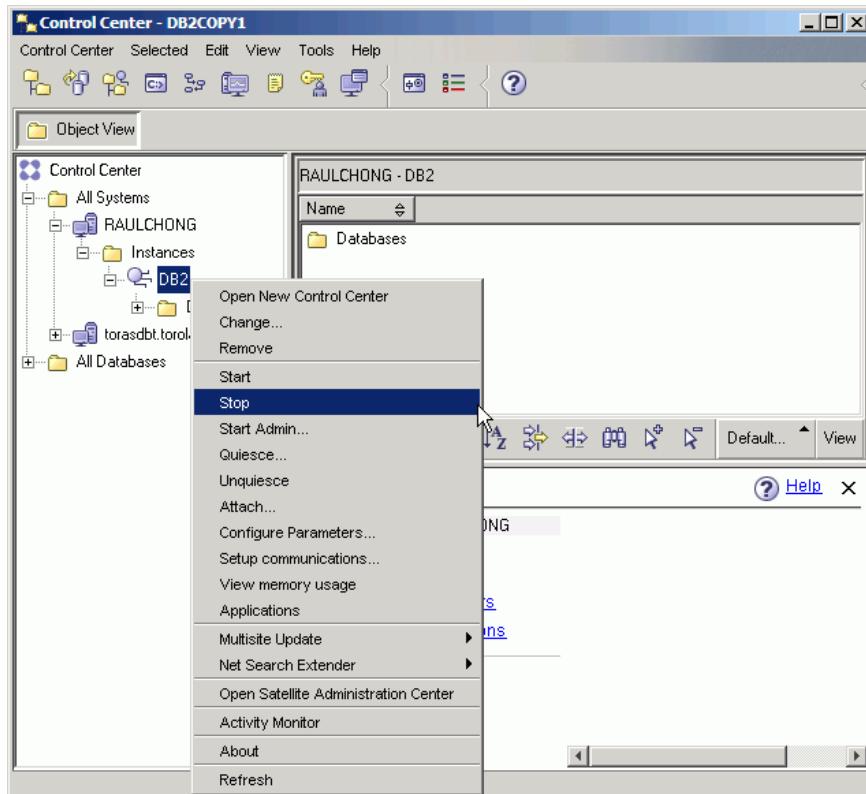


Рисунок 4.5. Команды экземпляра в Центре управления

Чтобы создать базу данных в активном экземпляре, в командном окне DB2 выполните команду:

```
db2 create database mydb1
```

Чтобы перечислить все созданные базы данных, выполните команду:

```
db2 list db directory
```

В рамках одного экземпляра можно создать много баз данных. База данных — это совокупность таких объектов, как таблицы, представления, индексы и пр. Базы данных являются независимыми единицами и, соответственно, не используют объекты совместно с другими базами данных. На рис. 4.6 представлена база данных **MYDB1**, созданная в экземпляре DB2.

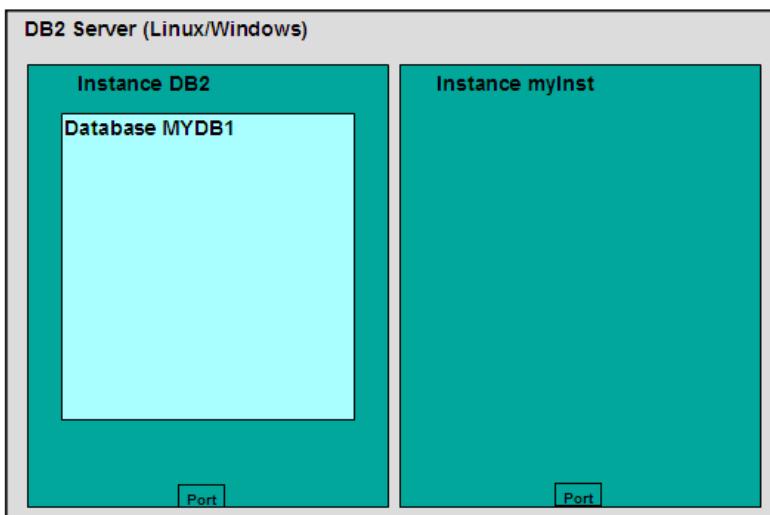


Рисунок 4.6. База данных MYDB1, созданная в экземпляре DB2

В Таблице 4.2 описаны некоторые команды, которые можно использовать на уровне баз данных.

| Команда/SQL-оператор | Описание |
|---|---|
| db2 create database | Создать новую базу данных |
| db2 drop database | Удалить базу данных |
| db2 connect to <имя_базы_данных> | Подключиться к базе данных |
| db2 create table/create view/create index | Операторы SQL для создания таблиц, представлений и индексов, соответственно |

Таблица 4.2. Команды/SQL-операторы на уровне баз данных

64 Начало работы с DB2 Express-C

Если нужно создать еще одну базу данных с таким же именем (*MYDB1*), но в экземпляре *myinst*, необходимо выполнить следующие команды в командном окне DB2:

```
db2 list db directory  
set db2instance=myinst  
db2 create database mydb1  
set db2instance=db2
```

На рис. 4.7 показана новая база данных *MYDB1*, созданная в экземпляре *myinst*.

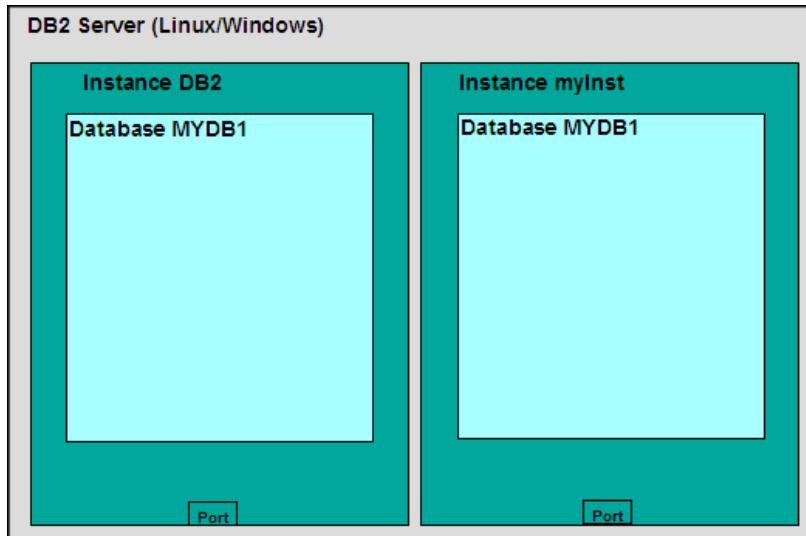


Рисунок 4.7. База данных MYDB1, созданная в экземпляре myInst

При создании базы данных несколько объектов создаются по умолчанию: табличные пространства, таблицы, буферный пул и файлы журнала. Создание этих объектов занимает некоторое время, поэтому команда `create database` выполняется несколько минут. В левой части рис. 4.8 показаны три табличных пространства, созданные по умолчанию. Табличные пространства описаны более подробно в Главе 6 «Архитектура DB2»; на данном этапе будем рассматривать табличные пространства как логический слой между логическими таблицами и физическими ресурсами, такими как диски и память.

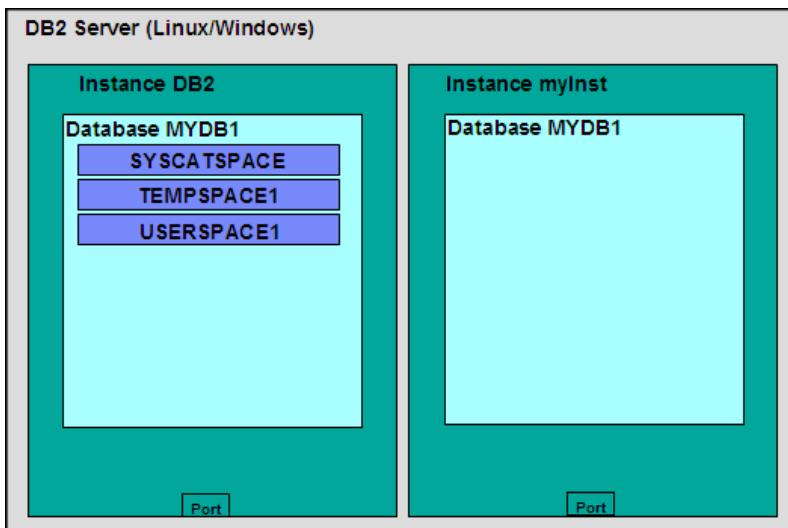


Рисунок 4.8. Табличные пространства, созданные по умолчанию при создании базы данных

Табличное пространство **SYSCATSPACE** содержит таблицы системного каталога. В других реляционных СУБД системный каталог называется словарем данных. По сути, он содержит системную информацию, которую нельзя изменять или удалять, иначе база данных не сможет корректно работать. Табличное пространство **TEMPSPACE1** используется в DB2, когда требуется дополнительное пространство для выполнения некоторых операций, таких как сортировка. Табличное пространство **USERSPACE1** обычно используется для хранения пользовательских таблиц базы данных, если при создании таблицы не указано другое табличное пространство.

Также можно создать собственные табличные пространства, воспользовавшись оператором **CREATE TABLESPACE**. На рис. 4.9 показано табличное пространство **MYTBL1**, созданное в базе данных **MYDB1** в экземпляре DB2. При создании табличного пространства указывают, какие диски и память (буферный пул) будут использоваться. Соответственно, при наличии «горячей» таблицы, т. е. таблицы, которая очень часто используется, можно выделить для неё самые быстрые диски и больше всего памяти, назначив ей табличное пространство с такими характеристиками.

На рис. 4.9 показаны еще два объекта, создаваемых по умолчанию: буферный пул **IBMDEFAULTTP** и файлы журнала.

Буферный пул — это, по сути, кэш-память, используемая базой данных. Можно создать один или несколько буферных пулов, но обязательно должен существовать хотя бы один буферный пул, размер страницы которого соответствует размеру страницы существующих табличных пространств. Страницы и их размеры рассмотрены более подробно в Главе 6 «Архитектура DB2».

Файлы журнала используются для восстановления. Во время работы с базой данных на соответствующих дисках сохраняется информация, но кроме этого все выполняемые с данными операции сохраняются в файлах журнала. Журналы можно рассматривать как временные файлы, в которых выполняется «автоматическое сохранение». Журналы описаны более подробно в Главе 11 «Резервное копирование и восстановление».

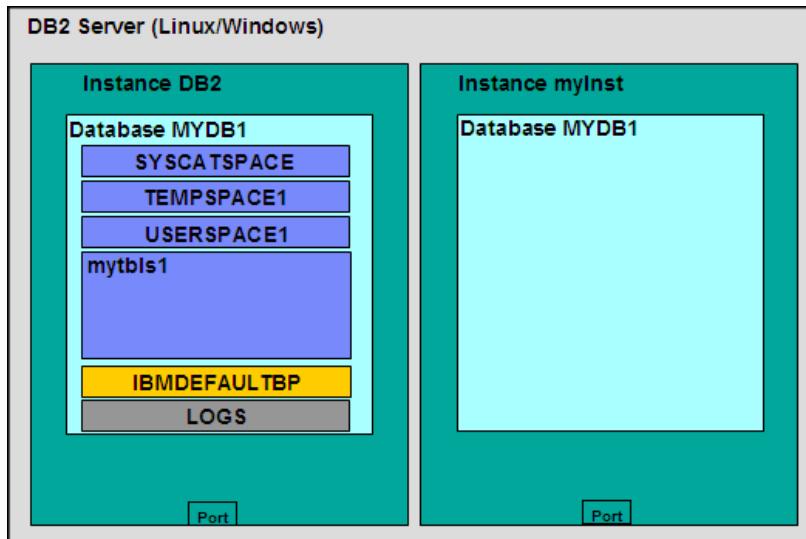


Рисунок 4.9. Буферный пул и журналы, созданные по умолчанию

Ранее упоминалось, что экземпляры являются независимыми средами, и потому в нескольких экземплярах можно создать базы данных с одинаковым именем. Как и экземпляры, базы данных также являются независимыми единицами; соответственно, объекты одной базы данных не имеют никакого отношения к объектам другой. На рис. 4.10 показано табличное пространство *mytbls1* в базах данных *MYDB1* и *SAMPLE* в экземпляре *db2*. Такое возможно, поскольку базы данных являются независимыми единицами. Обратите внимание, что в связи с ограниченным пространством на рис. 4.10 не показаны прочие создаваемые по умолчанию объекты базы данных *SAMPLE*.

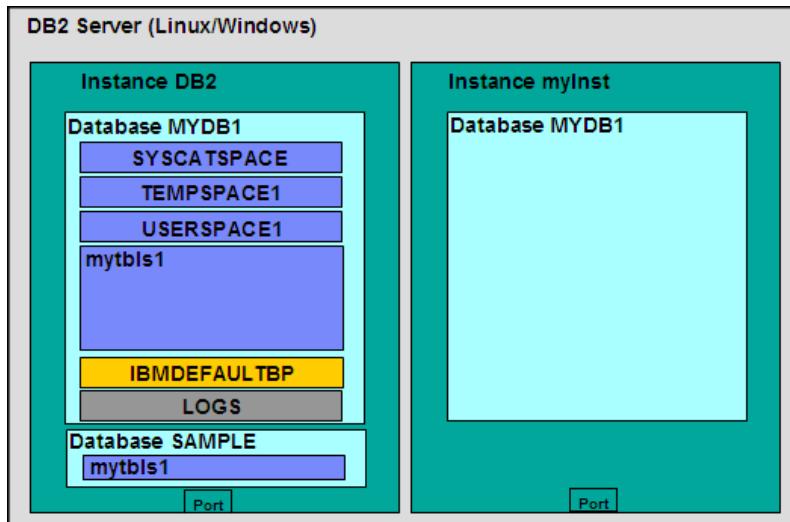


Рисунок 4.10. Табличные пространства с одинаковым именем в разных базах данных

Создав табличное пространство, можно создать в нем такие объекты, как таблицы, представления и индексы. Это показано на рис. 4.11.

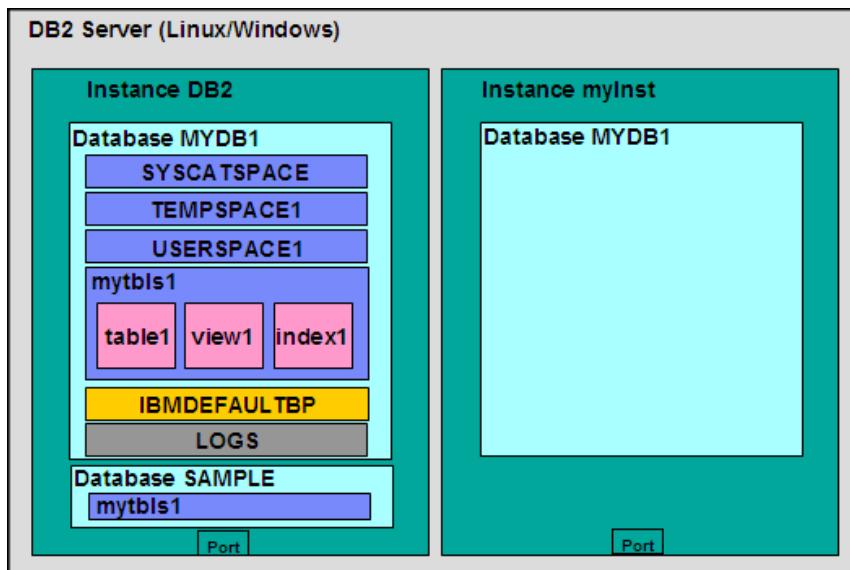


Рисунок 4.11. Таблицы, представления и индексы, созданные в табличном пространстве

4.1 Конфигурирование DB2

Параметры DB2 можно настроить с помощью инструмента «Советчик по конфигурированию». Чтобы открыть советчик по конфигурированию из Центра управления, щелкните правой кнопкой мыши базу данных и выберите пункт *Советчик по конфигурированию*. В зависимости от ответов на некоторые вопросы о системных ресурсах и рабочей нагрузке советчик по конфигурированию предоставит список параметров DB2, которые следовало бы изменить, а также укажет предпочтительные значения для каждого из параметров. Если хотите узнать о конфигурировании DB2 более подробно, продолжайте читать; в противном случае можете просто воспользоваться советчиком по конфигурированию, и сервер DB2 будет готов к работе!

Конфигурация сервера DB2 может быть задана на четырех различных уровнях:

- переменные среды;
- файл конфигурации менеджера базы данных (dbm cfg);
- файл конфигурации базы данных (db cfg);
- реестр профиля DB2.

Это также показано на рис. 4.12. Обратите внимание на то, где на рисунке расположен каждый из блоков. К примеру, переменные среды задаются на уровне

операционной системы сервера, а параметры файла конфигурации менеджера базы данных — на уровне экземпляра. Параметры конфигурации базы данных задаются на уровне базы данных, а реестр профиля DB2 — на уровне операционной системы или экземпляра.

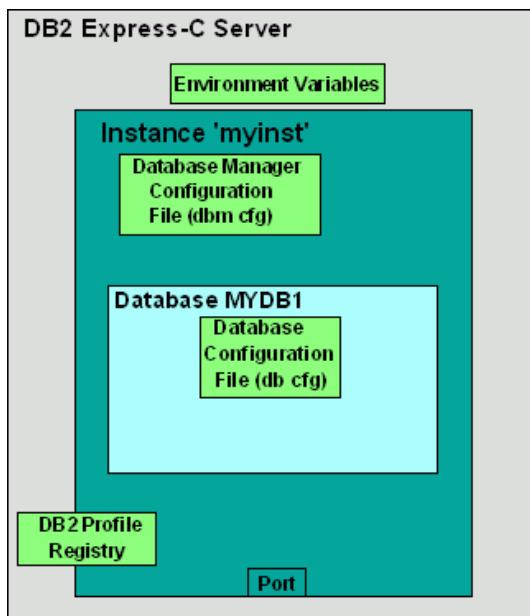


Рисунок 4.12 — Конфигурирование DB2

4.1.1 Переменные среды

Переменные среды — это переменные, установленные на уровне операционной системы. Одна из ключевых переменных среды — `DB2INSTANCE`. Эта переменная обозначает активный экземпляр, в котором выполняется работа и к которому будут применяться команды DB2. К примеру, чтобы установить `myinst` в качестве активного экземпляра, в Windows можно выполнить следующую команду операционной системы:

```
set db2instance=myinst
```

4.1.2 Файл конфигурации менеджера базы данных (dbm cfg)

Файл конфигурации менеджера базы данных (dbm cfg) включает параметры, влияющие на экземпляр и все базы данных, содержащихся в нем. Файл конфигурации менеджера базы данных можно просмотреть или изменить с помощью командной строки или через Центр управления DB2.

Для работы с dbm cfg через Центр управления выберите объект экземпляра в папке экземпляров центра управления, щелкните правой кнопкой мыши и во всплывающем меню выберите пункт *Конфигурировать параметры*. Это показано на рис. 4.13.

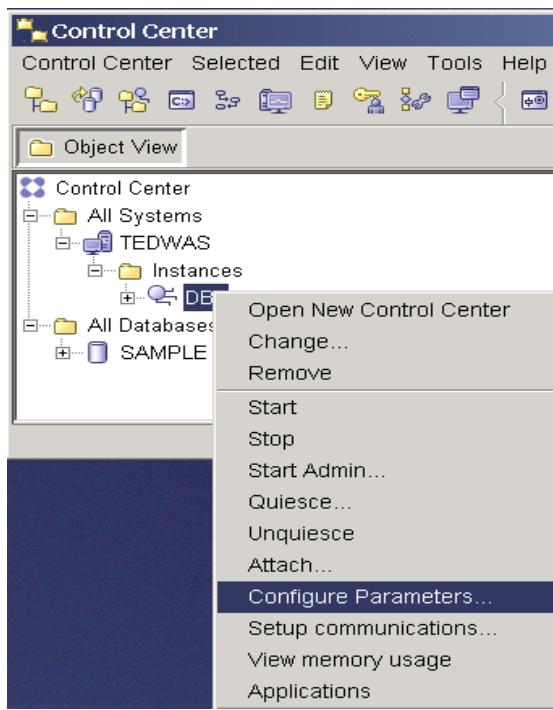


Рисунок 4.13. Конфигурирование dbm cfg из Центра управления

После выбора пункта *Конфигурировать параметры* откроется изображенное на рис. 4.14 окно со списком параметров dbm cfg.

70 Начало работы с DB2 Express-C

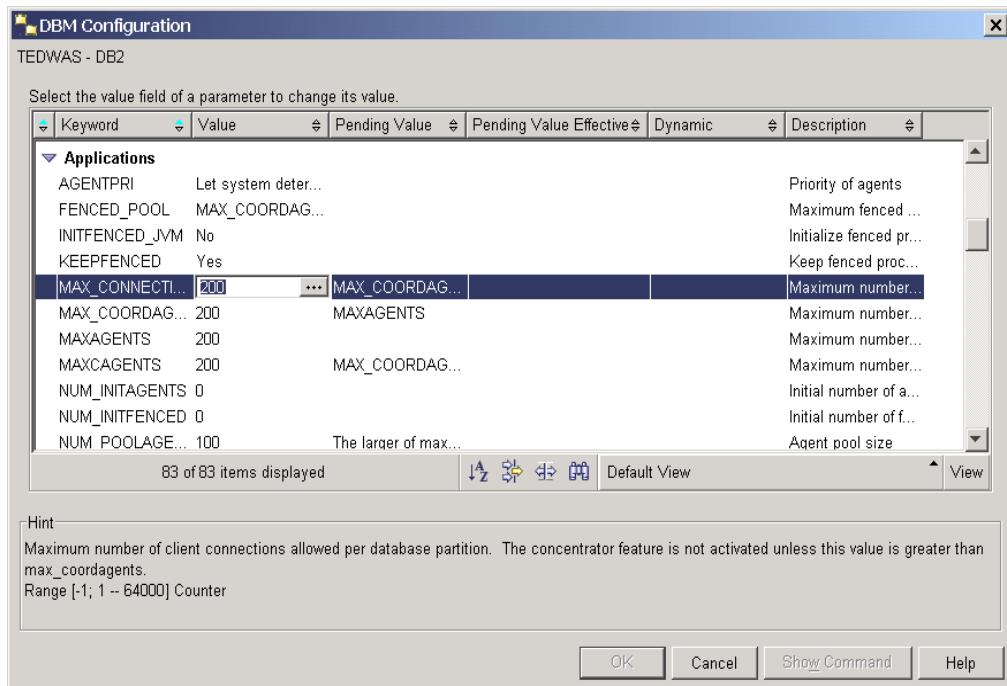


Рисунок 4.14. Диалоговое окно dbm cfg

Многие параметры являются динамическими, т. е. внесенные изменения сразу же вступают в силу; однако есть параметры, для изменения которых необходимо остановить и снова запустить экземпляр. Это можно сделать в командной строке с помощью команд `db2stop` и `db2start`.

Перед остановкой экземпляра все приложения должны отключиться. Для принудительной остановки экземпляра можно воспользоваться командой `db2stop force`.

Экземпляр также можно остановить через Центр управления, щелкнув на объект экземпляра и выбрав пункт *Остановить* или *Запуск*.

В Таблице 4.3 представлены некоторые полезные команды для управления `dbm cfg` из командной строки.

| Команда | Описание |
|--|--|
| <code>db2 get dbm cfg</code> | Извлечение информации о <code>dbm cfg</code> |
| <code>db2 update dbm cfg using <имя_параметра> <значение></code> | Обновление значения параметра <code>dbm cfg</code> |

Таблица 4.3. Команды для работы с dbm cfg

4.1.3 Файл конфигурации базы данных (db cfg)

Файл конфигурации базы данных (db cfg) включает параметры, влияющие на определенную базу данных. Файл конфигурации базы данных можно просмотреть или изменить с помощью командной строки или через Центр управления DB2.

Для работы с db cfg через Центр управления выберите объект базы данных в папке баз данных центра управления, щелкните правой кнопкой мыши и во всплывающем меню выберите пункт *Конфигурировать параметры*. Это показано на рис. 4.15.

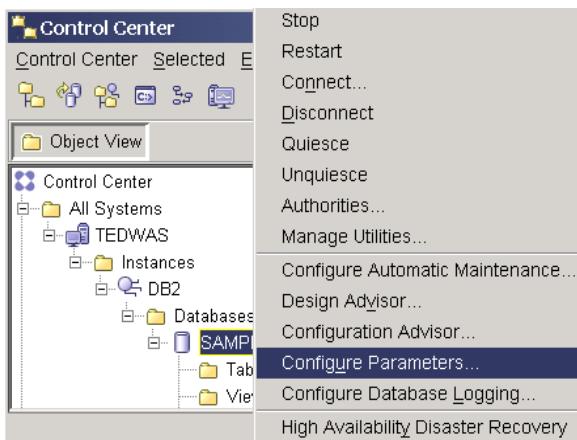


Рисунок 4.15. Конфигурирование db cfg через Центр управления

После выбора пункта *Конфигурировать параметры* откроется изображенное на рис. 4.16 окно со списком параметров db cfg.

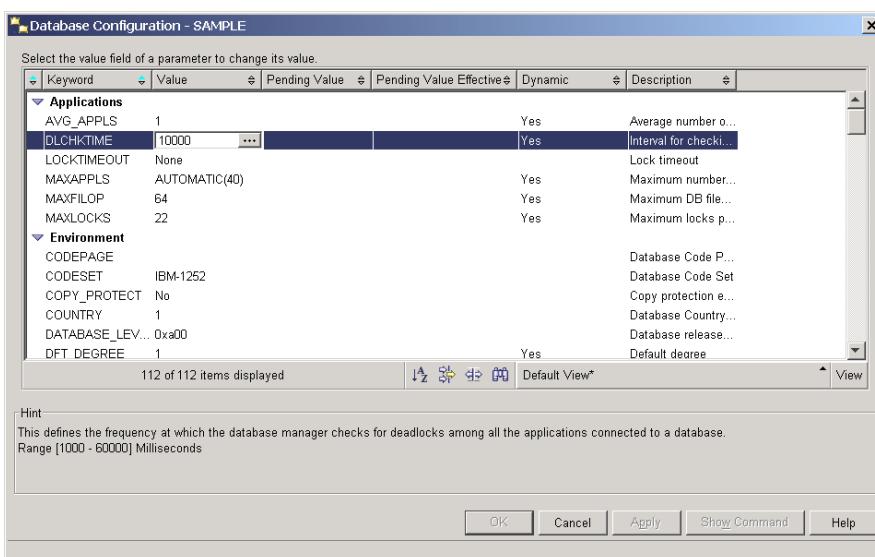


Рисунок 4.16. Список db cfg

72 Начало работы с DB2 Express-C

В Таблице 4.4 представлены некоторые полезные команды для управления db cfg из командной строки.

| Команда | Описание |
|---|---|
| <code>get db cfg for <имя_базы_данных></code> | Извлечение информации о db cfg для заданной базы данных |
| <code>update db cfg for <имя_базы_данных> using <имя_параметра> <значение></code> | Обновление значения параметра db cfg |

Таблица 4.4. Команды для работы с db cfg

4.1.4 Реестр профиля DB2

Переменные реестра профиля DB2 включают параметры, которые могут зависеть от платформы и устанавливаться глобально (для всех экземпляров) или на уровне экземпляра (для одного определенного экземпляра).

В Таблице 4.5 представлены некоторые полезные команды для управления реестром профиля DB2.

| Команда | Описание |
|---|---|
| <code>db2set -all</code> | Вывод списка всех заданных переменных реестра профиля DB2 |
| <code>db2set -lr</code> | Вывод списка всех переменных реестра профиля DB2 |
| <code>db2set <параметр>=<значение></code> | Присваивание параметру заданного значения |

Таблица 4.5. Команды для работы с реестром профиля DB2

В Таблице 4.6 представлены некоторые наиболее часто используемые переменные реестра DB2

| Переменная реестра | Описание |
|--------------------|--|
| DB2COMM | Определяет, какие менеджеры передачи данных запускаются при запуске менеджера базы данных |
| DB2_EXTSECURITY | Для Windows: предотвращает несанкционированный доступ к DB2, блокируя системные файлы DB2 |
| DB2_COPY_NAME | Хранит имя используемой в данное время копии DB2. Чтобы переключиться на другую установленную копию DB2, выполните команду <code>installpath\bin\db2envvar.bat</code> (данная переменная не может использоваться с этой целью). |

Таблица 4.6. Часто используемые переменные реестра профиля DB2

К примеру, чтобы разрешить соединение через протокол TCP/IP, установите для переменной реестра DB2COMM значение TCPIP, как показано ниже:

```
db2set db2comm=tcpip
```

4.2 Сервер администрирования DB2 (устарело)

Сервер администрирования DB2 Administration Server (DB2 Administration Server, DAS) — это процесс-демон, работающий на сервере DB2, чтобы разрешить удаленным клиентам администрировать сервер DB2 в графическом режиме. DAS требуется только при использовании графических инструментов DB2 как локально, так и удаленno. На одном физическом компьютере может находиться только один DAS (см. рис. 4.16).

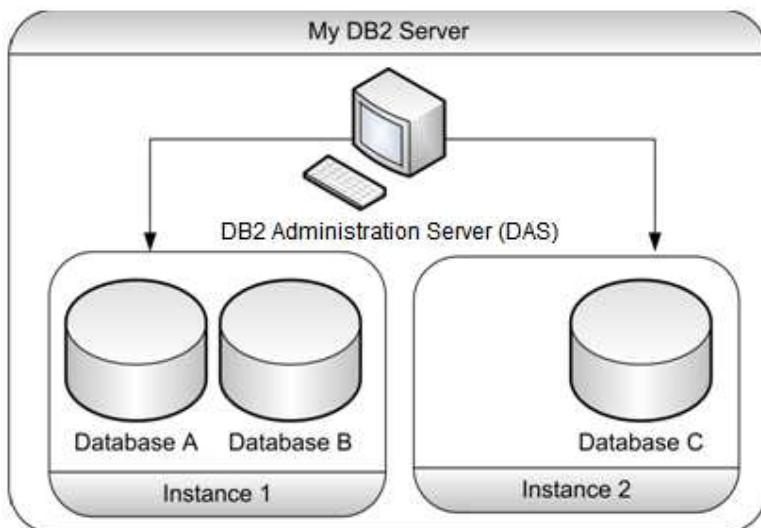


Рисунок 4.16. Сервер администрирования DB2 (DAS)

4.3 Краткий обзор

В этой главе была рассмотрена среда DB2, в частности принципы и вопросы создания экземпляров и баз данных, а также наиболее распространенные команды для работы с ними. Затем мы обсудили другие ключевые аспекты экземпляров, в том числе табличные пространства: три доступных типа табличных пространств, таблицы, представления и индексы, которые можно создать в табличных пространствах, буферные пулы и журналы.

В заключение мы обсудили структуру конфигурирования DB2 и способы её изменения на четырёх уровнях через переменные среды, файл конфигурации менеджера базы данных, файл конфигурации базы данных и реестр профиля DB2.

4.4 Упражнения

Упражнения этого раздела дадут возможность изучить принципы, рассмотренные в данной главе, а также познакомят с некоторыми инструментами DB2.

Часть 1. Создание новой базы данных с помощью мастера по созданию баз данных

В этой части будет создана база данных с помощью мастера по созданию баз данных в Центре управления.

Процедура

1. В окне объектов в левой части Центра управления щелкните правой кнопкой мыши на папке *Все базы данных* и выберите пункт *Создать базу данных -> С автоматическим обслуживанием*. Откроется Мастер по созданию баз данных.
2. Укажите имя и расположение базы данных на странице *Имя* мастера. Используйте следующие значения:

| | |
|------------------------------|--|
| Имя базы данных: | EXPRESS |
| Диск по умолчанию (Windows): | C: |
| Путь по умолчанию: (Linux): | /home/db2inst1 |
| Алиас: | если это поле оставить пустым, по умолчанию будет установлено значение EXPRESS |
| Комментарий: | это поле является необязательным, его можно не заполнять |

Нажмите кнопку *Далее*, чтобы перейти к следующей странице мастера.

new in
V9.7

Примечание. В Windows по умолчанию для создания базы данных может использоваться только диск, а не путь. Чтобы использовать для создания базы данных путь, установите такую переменную реестра DB2: DB2_CREATE_DB_ON_PATH

3. Не вносите никаких изменений на странице *Хранение* и нажмите кнопку *Далее*.
4. На странице *Обслуживание* оставьте значение по умолчанию (*Да, я могу задать окно для обслуживания в автономном режиме...*) и нажмите *Далее*.
5. Задайте время окна обслуживания в автономном режиме на странице *Временные характеристики* мастера. Установите для окна обслуживания время начала 01:00 и длительность 6 часов с понедельника до четверга. Нажмите кнопку *Далее*.
6. Настройте уведомления на странице *Почтовый сервер* мастера. DB2 может автоматически отправлять уведомление на пейджер или электронный адрес при возникновении проблемы или обнаружении опасных условий. Если необходимо настроить эту функцию, укажите доступный SMTP-сервер, который будет использоваться DB2 для отправки электронных сообщений. В данном упражнении мы не используем SMTP-сервер, поэтому не заполняйте это поле и нажмите *Далее*.

Проверьте выбранные параметры на странице *Сводка* мастера. Нажмите кнопку *Готово*, чтобы запустить процесс создания базы данных. Обычно создание базы данных длится несколько минут, и в это время отображается индикатор выполнения.

Часть 2. Работа с экземплярами, базами данных и конфигурированием

В этой части мы создадим новый экземпляр и базу данных, а также изменим параметры конфигурирования на сервере DB2 в Windows. Это можно сделать в Центре управления или командном окне DB2. Ниже предоставлены инструкции для командного окна DB2.

Процедура

1. Откройте командное окно DB2, выбрав *Пуск -> Все программы -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Window (DB2COPY1)* (по умолчанию) -> Инструменты командной строки -> Командное окно). Командное окно также можно запустить быстрее, выбрав *Пуск -> Выполнить* и напечатав `db2cmd`.
2. В командном окне DB2 создайте новый экземпляр с именем `newinst`.

```
db2icrt newinst
```

3. Переключитесь на экземпляр `newinst` и убедитесь в том, что он действительно является текущим экземпляром. Затем запустите его.

```
set db2instance=newinst (Совет. Не используйте пробелы до и после знака «=>!»)
```

```
db2 get instance (Проверка того, что newinst является текущим экземпляром)
```

```
db2start
```

76 Начало работы с DB2 Express-C

4. Создайте базу данных `newdb` со значениями по умолчанию в экземпляре `newinst`. Это может занять несколько минут, поскольку DB2 создает внутренние объекты базы данных и выполняет начальное конфигурирование.

```
db2 create database newdb
```

5. Установите и разорвите соединение с новой базой данных `newdb`. Затем удалите её.

```
db2 connect to newdb  
db2 terminate  
db2 drop db newdb
```

6. Остановите текущий экземпляр `newinst`.

```
db2stop
```

7. Выведите список всех экземпляров в системе.

```
db2ilist
```

8. Переключитесь на экземпляр DB2 и убедитесь в том, что переключение действительно выполнено.

```
set db2instance=db2  
db2 get instance
```

9. Удалите экземпляр `newinst`.

```
db2idrop newinst
```

10. Узнайте текущее значение параметра `dbm cfg FEDERATED`. По умолчанию должно быть установлено значение `NO`.

```
db2 get dbm cfg
```

Совет. В Linux можно выполнить следующее: `db2 get dbm cfg | grep FEDERATED`

11. Установите для параметра `dbm cfg FEDERATED` значение `YES` и убедитесь в том, что изменение внесено.

```
db2 update dbm cfg using FEDERATED YES
```

Поскольку параметр `FEDERATED` не является динамическим, изменения вступают в силу только после остановки и повторного запуска экземпляра. Однако прежде чем останавливать экземпляр, необходимо убедиться в отсутствии подключений. Один из способов сделать это — выполнить следующие команды:

```
db2 force applications all  
db2 terminate
```

Перезапустите экземпляр и проверьте новое значение параметра FEDERATED:

```
db2stop  
db2start  
db2 get dbm cfg
```

12. Подключитесь к базе данных SAMPLE с помощью идентификатора пользователя и пароля, которые используются в операционной системе.

```
db2 connect to sample user <идентификатор_пользователя> using <пароль>
```

13. Проверьте, сколько приложений запущено в текущем экземпляре.

```
db2 list applications
```

14. Откройте еще одно командное окно DB2 и снова подключитесь к базе данных SAMPLE, не указывая идентификатор пользователя и пароль. Затем проверьте количество подключений.

```
db2 connect to sample  
db2 list applications
```

15. Прервите подключение в одном из командных окон DB2. Это пример того, как администратор баз данных может принудительно прекратить работу определенного пользователя (который, вероятно, неравномерно потребляет системные ресурсы).

```
db2 force application (<хэндл окна приложения для db2bp.exe>)
```

Хэндл окна приложения — это номер или «хэндл» окна приложения, которое необходимо принудительно остановить. Этот номер можно получить из результатов выполнения команды `db2 list applications`. Приложение db2bp.exe соответствует командному окну DB2.

16. Убедитесь в том, что подключение одного из командных окон DB2 прервано. Если не известно, какое из командных окон DB2 было отключено, выполните следующий оператор для обоих окон.

```
db2 select * from staff
```

Командное окно DB2, которое было отключено, выдаст сообщение об ошибке с кодом SQL1224N. Другое командное окно DB2 должно снова отобразить результаты запроса.

17. Удалите и восстановите DAS, затем запустите его.

```
db2admin stop  
db2admin drop  
db2admin create  
db2admin start
```

18. Взгляните на текущее значение переменной реестра DB2COMM.

```
db2set -all
```

19. Сбросьте переменную реестра DB2COMM и проверьте результат.

```
db2set db2comm=  
db2stop
```

(Совет. Если установлены подключения, команда db2stop выдаст сообщение об ошибке. Что нужно сделать? Для решения этой проблемы см. предыдущий шаг.)

```
db2start  
db2set -all
```

20. Задайте для переменной реестра DB2 DB2COMM значения `tcpip` и `npipe` в используемом экземпляре и проверьте новые значения.

```
db2set db2comm=tcpip,npipe  
db2stop  
db2start  
db2set -all
```

21. Проверьте текущее значение параметра `db cfg LOGSECOND`, затем установите значение 5 и проверьте новое значение.

```
db2 connect to sample  
db2 get db cfg  
db2 update db cfg using LOGSECOND 5  
db2 get db cfg
```

5

Глава 5. Инструменты DB2

В этой главе представлены некоторые инструменты, используемые с DB2. Начиная с DB2 версии 9.7, большинство описанных в этой главе инструментов считаются устаревшими — они все еще поддерживаются, но уже не будут улучшаться и могут быть исключены из будущих выпусков продукта. Заменой для таких инструментов служит IBM Data Studio.

На рис. 5.1 красным овалом выделена та часть, о которой рассказывается в этой главе.

new in

V9.7

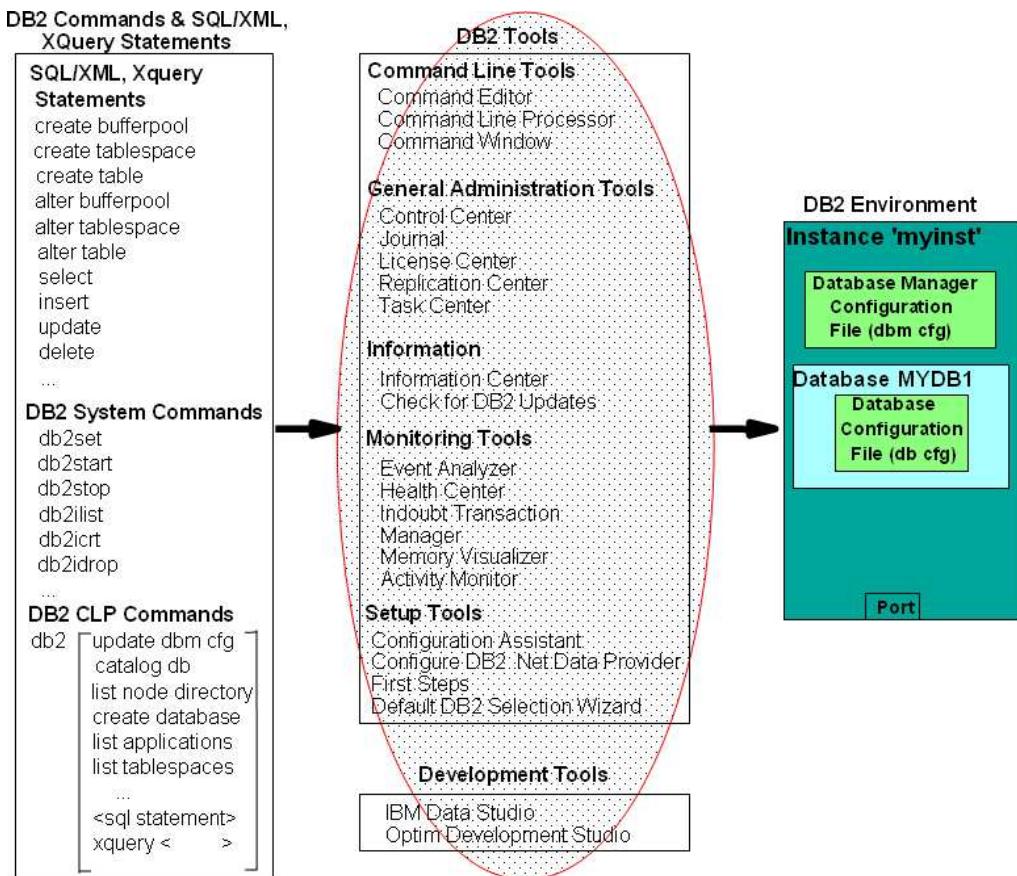


Рисунок 5.1. Общий обзор DB2: инструменты DB2

Примечание.

Чтобы просмотреть видео-презентации об инструментах DB2 и написании сценариев, перейдите по этим ссылкам:

<http://www.channeldb2.com/video/video/show?id=807741:Video:4202>

<http://www.channeldb2.com/video/video/show?id=807741:Video:4182>

На рис. 5.2 показаны все инструменты DB2, доступные через ярлыки IBM DB2 в меню Пуск. Большинство этих инструментов одинаковы в Linux и Windows.

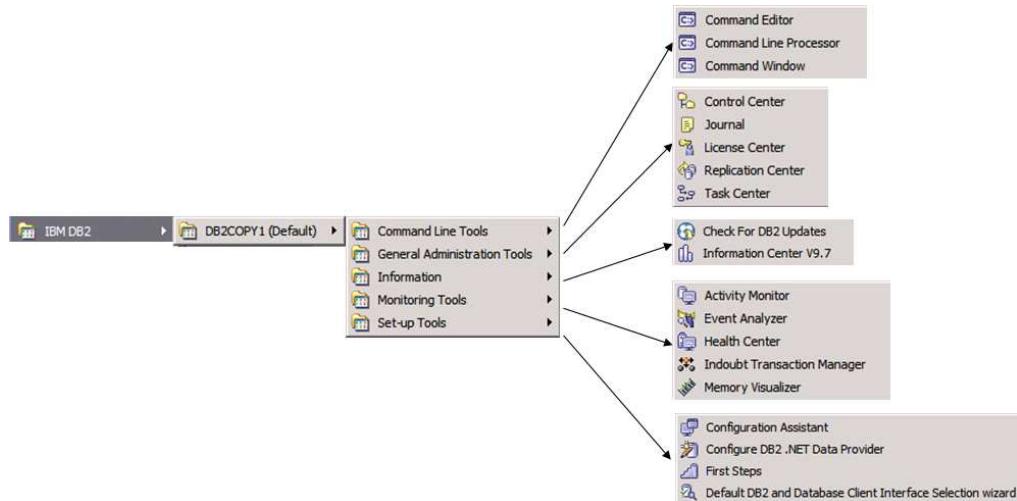


Рисунок 5.2. Инструменты DB2 из меню Пуск IBM DB2

В Таблице 5.1 перечислены команды для запуска наиболее популярных инструментов в Linux или Windows. Также указано, какие инструменты считаются устаревшими в DB2 9.7.

| Название инструмента | Команда | Устарело? |
|---|---------|-----------|
| Редактор команд | db2ce | Да |
| Процессор командной строки | db2 | Нет |
| Командное окно (только на платформах Windows) | db2cmd | Нет |
| Центр управления | db2cc | Да |
| Центр задач | db2tc | Да |
| Центр работоспособности | db2hc | Да |

| | | |
|----------------------------|-------|-----|
| Ассистент конфигурирования | db2ca | Да |
| Первые шаги | db2fs | Нет |

Таблица 5.1. Инstrumentальные команды для запуска некоторых инструментов DB2

new in
V9.7

5.1 IBM Data Studio

Начиная с DB2 9.7 IBM Data Studio является основным инструментом администрирования и разработки баз данных в DB2. Инструмент Data Studio бесплатный. Он работает в Linux и Windows, а также входит в состав [портфеля решений компании IBM для интегрированного управления данными](#). Разработка Data Studio выполняется по отдельному графику, не привязанному к выпускам DB2, однако выпуски этих продуктов согласовываются настолько часто, насколько возможно. К примеру, DB2 9.7 и IBM Data Studio 2.2 были выпущены в один день в июне 2009 года.

На рис. 5.3 показан внешний вид IBM Data Studio.

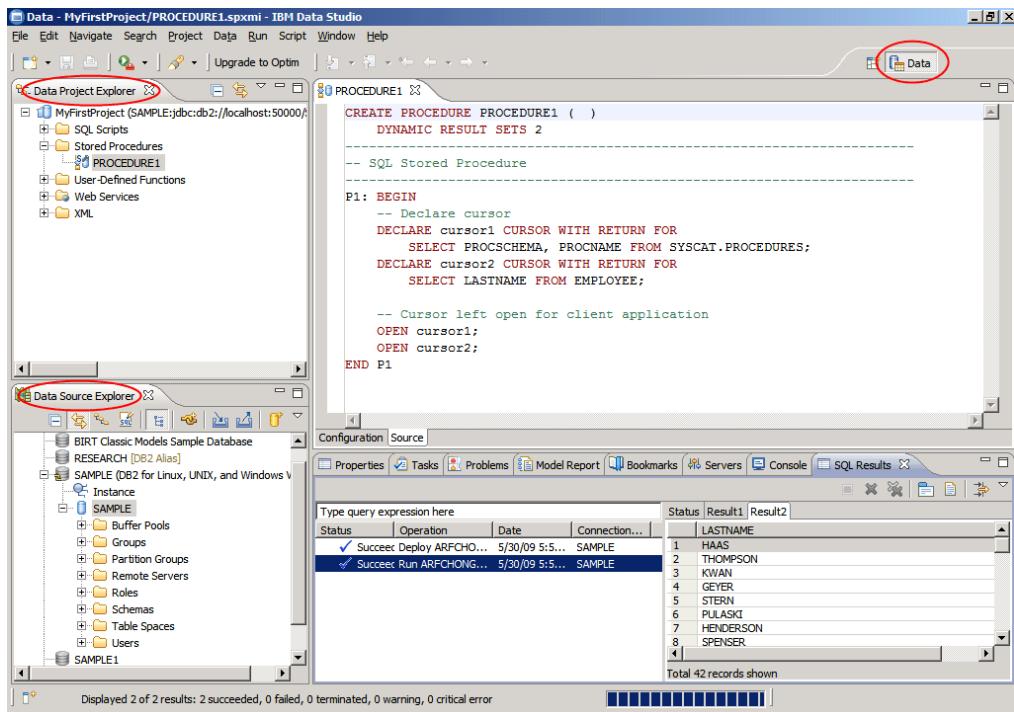


Рисунок 5.3. IBM Data Studio

Пользователи, знакомые с Eclipse, обратят внимание на то, что инструмент Data Studio основан на Eclipse. Как правило, работа в Data Studio осуществляется в окне перспективы Data (Данные) (выделено на рисунке в верхнем правом углу). Можно

также переключиться к перспективе Java, если разрабатывается Java-программа. На рисунке выделено два представления:

- Data Project Explorer (слева вверху).
- Data Source Explorer (слева внизу).

Представление Data Project Explorer (Проводник проектов данных) используется разработчиками баз данных для работы с SQL-схемариями, XQuery, хранимыми процедурами, пользовательскими функциями и веб-службами данных.

Представление Data Source Explorer (Проводник источников данных) используется администраторами баз данных для управления экземплярами и базами данных DB2. С помощью этого представления можно использовать большинство функциональных возможностей, ранее доступных в Центре управления.

На нашем рисунке представление под заголовком PROCEDURE1 является редактором для процедуры, выделенной в Data Project Explorer. В зависимости от выполняемой задачи, открываются редакторы и другие окна, позволяющие либо программировать, либо осуществлять конфигурирование.

Доступны два варианта IBM Data Studio:

- отдельный пакет;
- IDE-пакет.

Отдельный пакет намного меньше, чем пакет IDE, но не поддерживает разработку веб-служб данных и не может быть расширен до других продуктов IBM на базе Eclipse, таких как InfoSphere Data Architect. В остальном их интерфейс и возможности одинаковы.

С помощью IBM Data Studio можно также работать с другими серверами данных, например Informix. Компании, работающие с несколькими серверами данных и имеющие небольшую команду администраторов или разработчиков баз данных, теперь могут работать и управлять ими с помощью одного инструмента, что очень удобно.

Примечание.

Чтобы получить более подробную информацию о Data Studio, ознакомьтесь с бесплатной электронной книгой [Начало работы с IBM Data Studio для DB2](#), которая является частью данной серии книг.

5.2 Центр управления (устарело)

До выхода DB2 9.7 основным инструментом DB2 для администрирования баз данных был Центр управления (см. *рис. 5.4*).

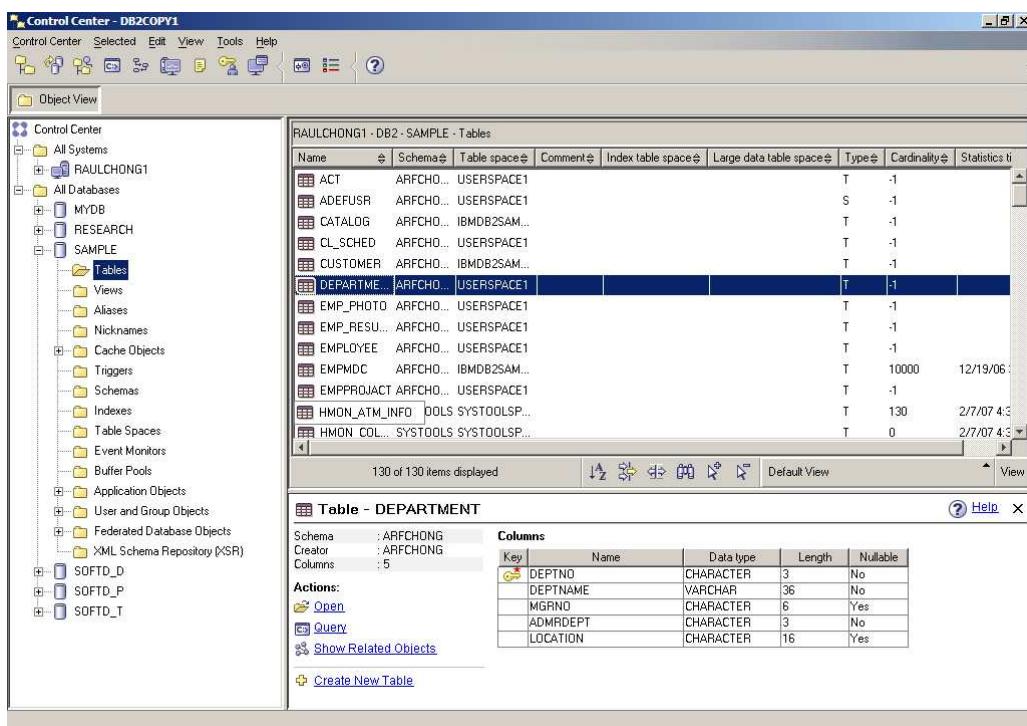


Рисунок 5.4. Центр управления DB2

Центр управления — это централизованный инструмент администрирования, позволяющий:

- просматривать системы, экземпляры, базы данных и объекты баз данных;
- создавать и модифицировать базы данных и объекты баз данных, а также управлять ими;
- запускать другие графические инструменты DB2.

На панели с левой стороны визуально представлена иерархия объектов базы данных в системе(-ах) в виде отдельных папок для таблиц, представлений и т. п. Если дважды щелкнуть по папке (например, по папке таблицы, как показано на рис. 5.4), на панели вверху справа будет выведен список всех соответствующих объектов, в данном случае — всех таблиц, связанных с базой данных SAMPLE. Если выбрать одну из таблиц на верхней правой панели, на нижней правой панели будет отображена более подробная информация о ней.

Если щелкнуть правой кнопкой мыши на других папках или объектах дерева объектов, откроется соответствующее им меню. К примеру, если щелкнуть правой кнопкой мыши на экземпляре и выбрать пункт Конфигурировать параметры, можно просмотреть и обновить файл конфигурации менеджера базы данных. Аналогично, правый щелчок на базе данных и выбор пункта Конфигурировать параметры позволит просмотреть и

обновить файл конфигурации базы данных. Среда DB2 и параметры конфигурирования рассматриваются более подробно в *Главе 5 «Среда DB2»*.

При первом запуске Центра управления система предложит выбрать предпочтительный вид. От выбора вида зависит отображение объектов базы данных и их параметров. На *рис. 5.5* показано диалоговое окно «Вид Центра управления».

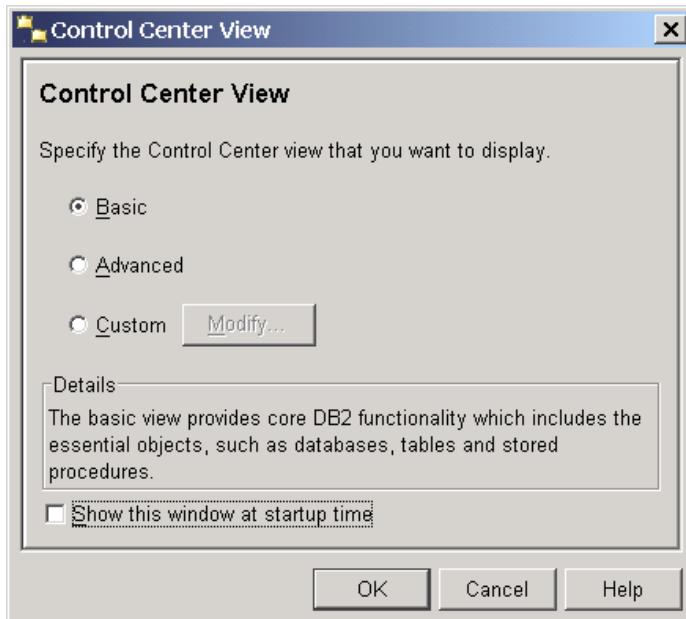


Рисунок 5.5. Диалоговое окно «Вид Центра управления» DB2

Основной вид предоставляет только главные функциональные возможности DB2, а расширенный отображает больше параметров и функций. Пользовательский вид дает возможность настроить отображение требуемых функций, параметров и объектов.

Чтобы повторно открыть диалоговое окно «Вид Центра управления», выберите *Инструменты -> Настроить центр управления*, как показано на *рис. 5.6*.

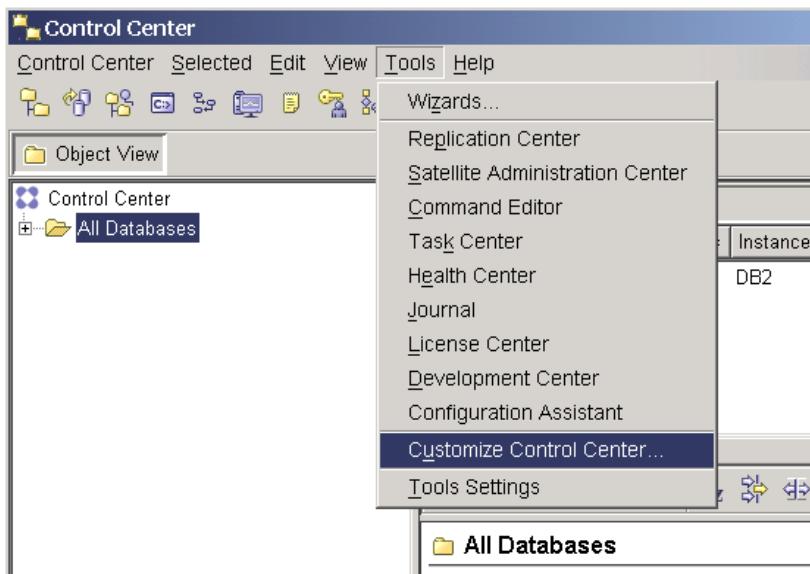


Рисунок 5.6. Настройка Центра управления

5.2.1 Запуск Центра управления

Есть много способов запустить Центр управления:

- перейти через меню *Пуск* в Windows;
- выполнить `db2cc` в командной строке;
- нажать пиктограмму Центра управления  на панели инструментов любого другого GUI-инструмента DB2;
- с помощью пиктограммы DB2 на панели задач Windows, как показано на рис. 5.7 (щелкните правой кнопкой мыши на зеленую пиктограмму DB2 и выберите пункт меню «Центр управления DB2»).

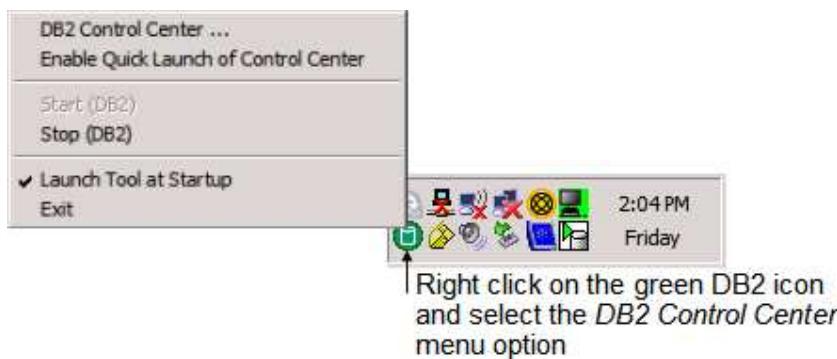


Рисунок 5.7. Запуск Центра управления DB2 с панели задач Windows

5.3 Редактор команд (устарело)

С помощью Редактора команд DB2 можно выполнять команды DB2, операторы SQL и XQuery, анализировать план выполнения операторов, а также просматривать или обновлять наборы результатов запросов. На рис. 5.8 показан Редактор команд и описаны его элементы.

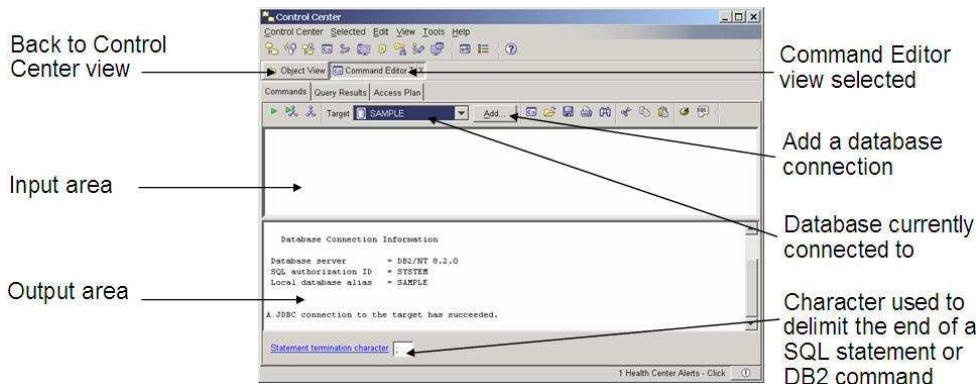


Рисунок 5.8. Редактор команд DB2

В области ввода можно задать несколько операторов, при условии, что каждый оператор заканчивается завершающим символом. Если нажать кнопку выполнения (первая кнопка слева на рис. 5.9), все операторы будут поочерёдно выполнены. Если же выделить отдельный оператор, будет выполнен только он. Для выполнения операторов SQL необходимо соединение с базой данных, однако один из операторов может быть оператором соединения.

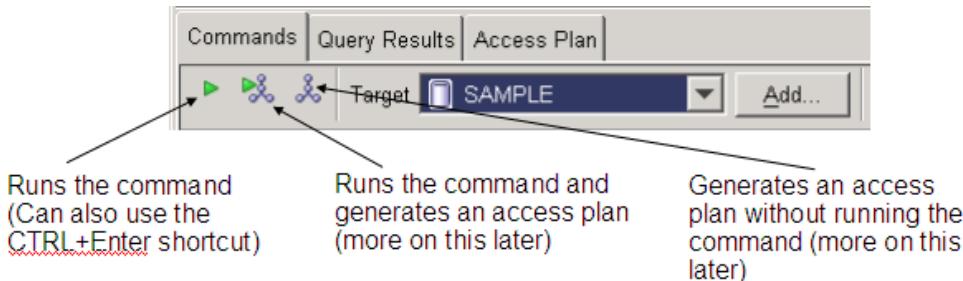


Рисунок 5.9. Редактор команд — вкладка «Команды»

5.3.1 Запуск Редактора команд

Редактор команд можно запустить несколькими способами:

- Из меню Пуск в Windows: Пуск -> Все программы -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Editor (DB2COPY1 (по умолчанию)) -> Инструменты командной строки -> Редактор команд.
- Из командной строки: ввести db2ce.
- Из меню «Инструменты» в Центре управления.

- В Центре управления:
 - щелкните правой кнопкой мыши на пиктограмме базы данных **SAMPLE** на панели дерева объектов Центра управления и выберите пункт меню Запрос;
 - каждый раз при выборе объекта, поддерживающего запросы (база данных, таблица и т. п.), можно запускать Редактор команд, нажимая на ссылку Запрос на панели информации об объектах Центра управления.
 - В Центре управления нажмите пиктограмму Редактора команд  на панели инструментов Центра управления, как показано на рис. 5.10.

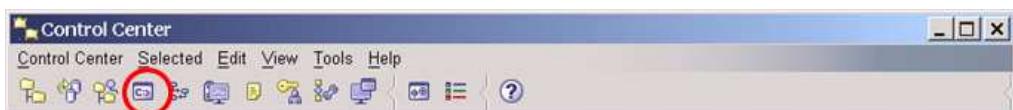


Рисунок 5.10. Пиктограмма Редактора команд в Центре управления

5.3.2 Добавление соединения с базой данных

Чтобы добавить соединение с базой данных, нажмите кнопку Добавить (см. рис. 5.8). Откроется диалоговое окно, показанное на рис. 5.11.

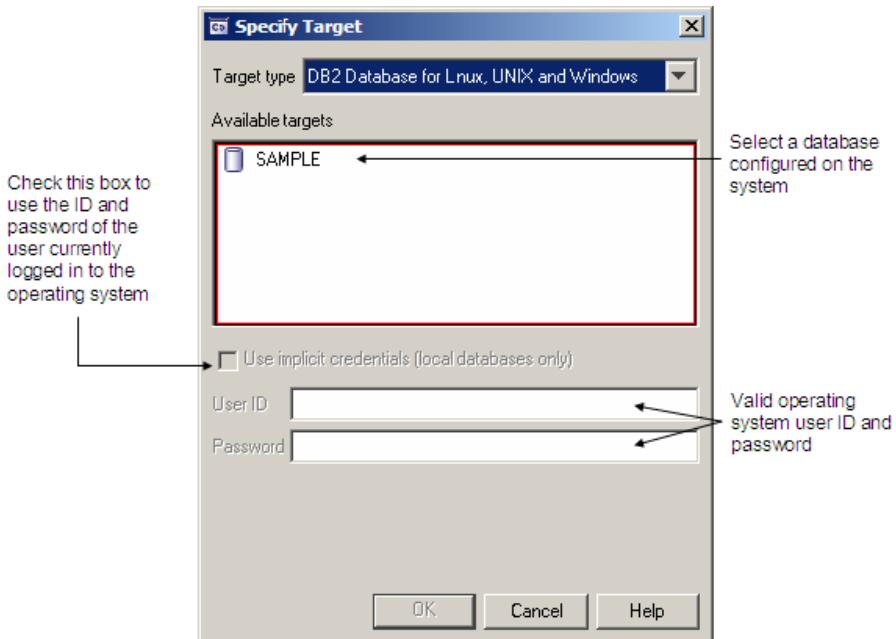


Рисунок 5.11. Добавление соединения с базой данных

5.4 Мастер SQL Assist (устарело)

Если вы не знакомы с языком SQL и хотите, чтобы помощник или мастер сгенерировал SQL-код, вам поможет мастер SQL Assist (мастер построения SQL), доступный в Редакторе команд. Как показано на *рис. 5.12*, он запускается через Редактор команд нажатием на последнюю пиктограмму с надписью SQL. Эта пиктограмма отображается только после подключения к базе данных.

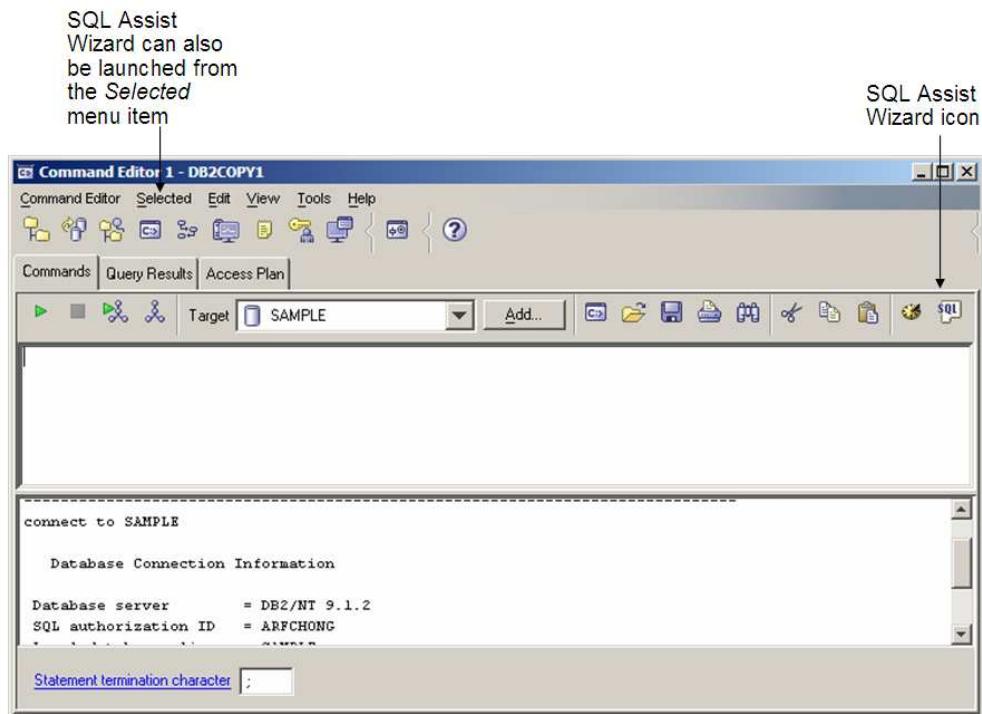


Рисунок 5.12. Запуск мастера SQL Assist

На *рис. 5.13* показан мастер SQL Assist. Он довольно прост в использовании. Сначала необходимо указать тип оператора SQL, для которого требуется помощь (SELECT, INSERT, UPDATE, DELETE). В зависимости от выбранного оператора, будут показаны разные параметры. В нижней части окна можно проследить за построением оператора SQL по мере выбора различных вариантов в мастере.

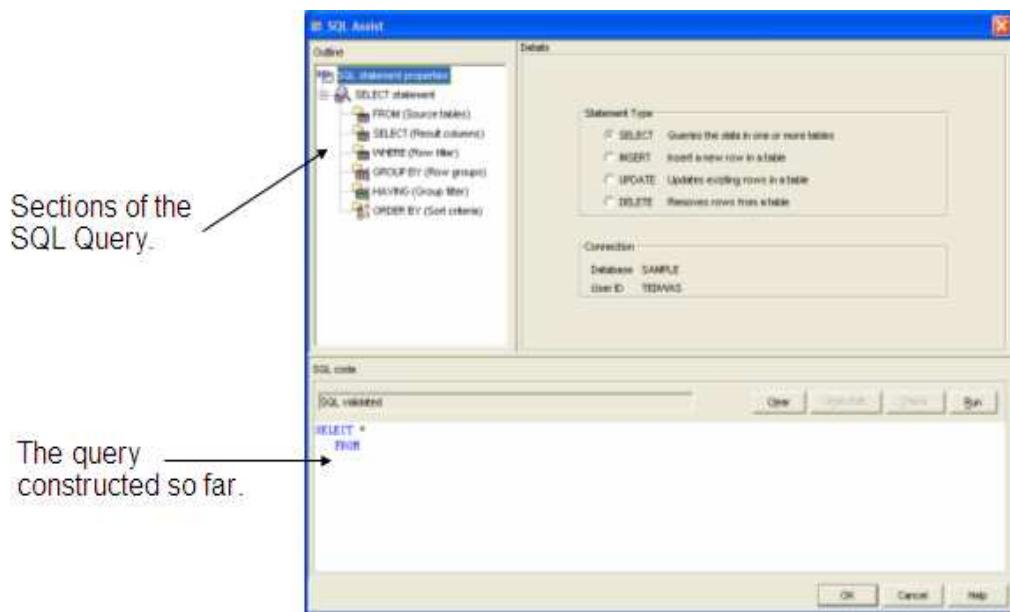


Рисунок 5.13. Мастер SQL Assist

5.5 Кнопка «Показать SQL» (устарело)

Большинство GUI-инструментов и мастеров в DB2 позволяют просмотреть фактическую команду или оператор SQL, создаваемый в результате их использования для выполнения определенного действия. Для этого достаточно нажать кнопку **Показать SQL** в используемом инструменте, как показано на *рис. 5.14* и *рис. 5.15*.



Рисунок 5.14. Кнопка «Показать SQL»



Рисунок 5.15 — Результат нажатия кнопки «Показать SQL»

Возможность просмотреть операторы и команды SQL очень полезна для изучения синтаксиса SQL, а также для сохранения команд или операторов в файл для использования в дальнейшем. Также можно построить сценарии, повторно воспользовавшись такими сгенерированными командами и операторами.

5.6 Центр задач (устарело)

GUI-инструмент Центр задач позволяет создавать задачи — набор операций, таких как выполнение команд DB2, команд операционной системы или сценариев. Последующие действия можно выполнить после успешного или неудачного выполнения задачи. К примеру, если задача создания резервной копии важной базы данных в 3:00 утра пройдет успешно, можно отправить электронное сообщение администратору базы данных с уведомлением об этом. С другой стороны, если задача создания резервной копии не будет выполнена, Центр задач может отправить сообщение на пейджер администратора базы данных. На *рис. 5.16* показан Центр задач.

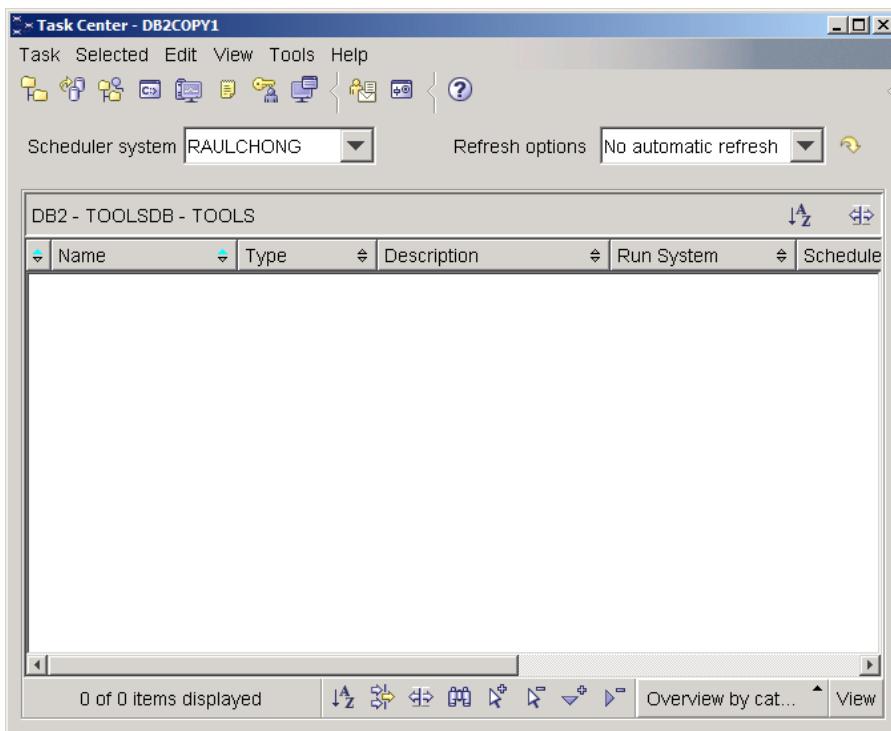


Рисунок 5.16. Центр задач

5.6.1 База данных каталога инструментов (устарело)

Вся информация о задачах и их расписании хранится в отдельной базе данных DB2 — базе данных каталога инструментов. Эта база данных должна существовать до назначения расписания задач. Для создания базы данных каталога инструментов можно воспользоваться следующей командой:

```
CREATE TOOLS CATALOG systools CREATE NEW DATABASE toolsdb
```

В приведенном выше примере *systools* — это имя схемы всех таблиц базы данных, а база данных имеет имя *toolsdb*. Схемы рассмотрены более подробно в Главе 8 «Работа с объектами базы данных».

5.6.1.1 Запуск Центра задач

Центр задач можно запустить из Центра управления, нажав *Инструменты* -> *Центр задач*, как показано на рис. 5.17. Этот инструмент можно также запустить из меню Пуск в Windows: Пуск -> Все программы -> IBM DB2 -> DB2COPY1 (Default) -> General Administration Tools -> Task Center (DB2COPY1 (по умолчанию) -> Общие инструменты управления -> Центр задач).

92 Начало работы с DB2 Express-C

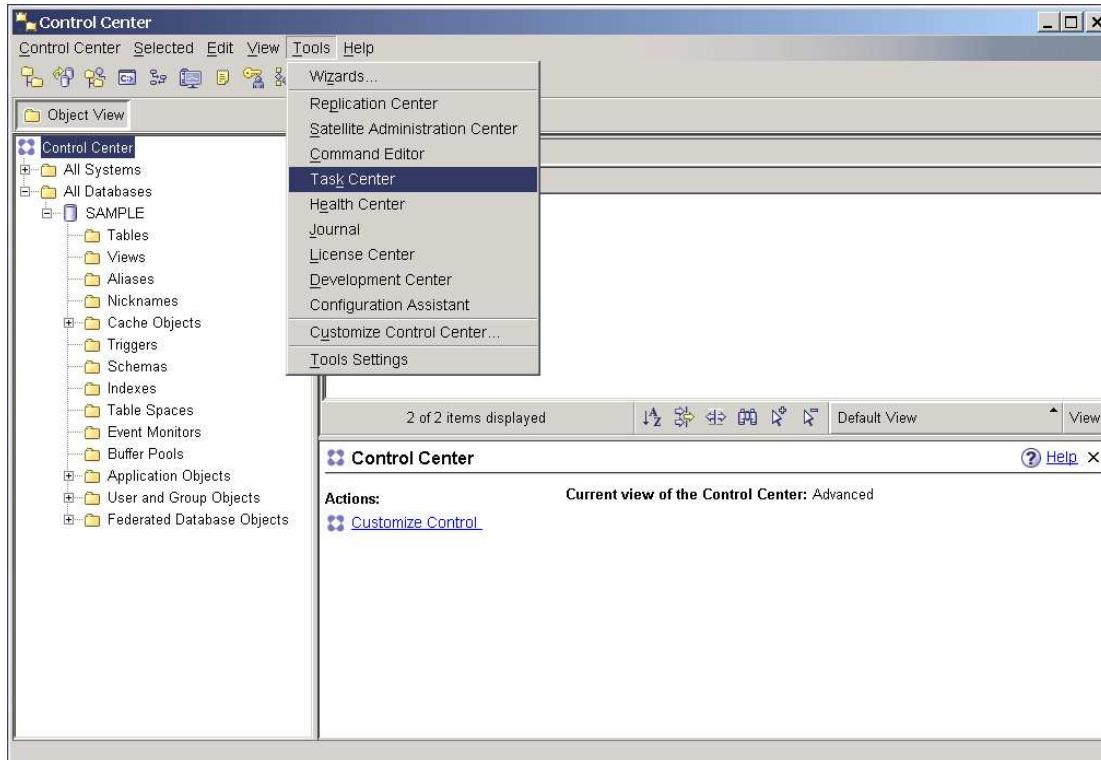


Рисунок 5.17. Запуск Центра задач

5.6.1.2 Планирование в Центре задач

Любой сценарий, созданный с помощью GUI-инструментов DB2 или другими средствами, можно запланировать в Центре задач. Задачи запускаются в назначенное время в системе, в которой создана база данных каталога инструментов. Советуем самостоятельно изучить Центр задач. Процесс создания задачи очень простой.

5.7 Журнал (устарело)

GUI-инструмент Журнал DB2 предоставляет администраторам базы данных онлайновый журнал операций. На рис. 5.18 показан Журнал, а в табл. 5.2 представлена доступная в Журнале информация.

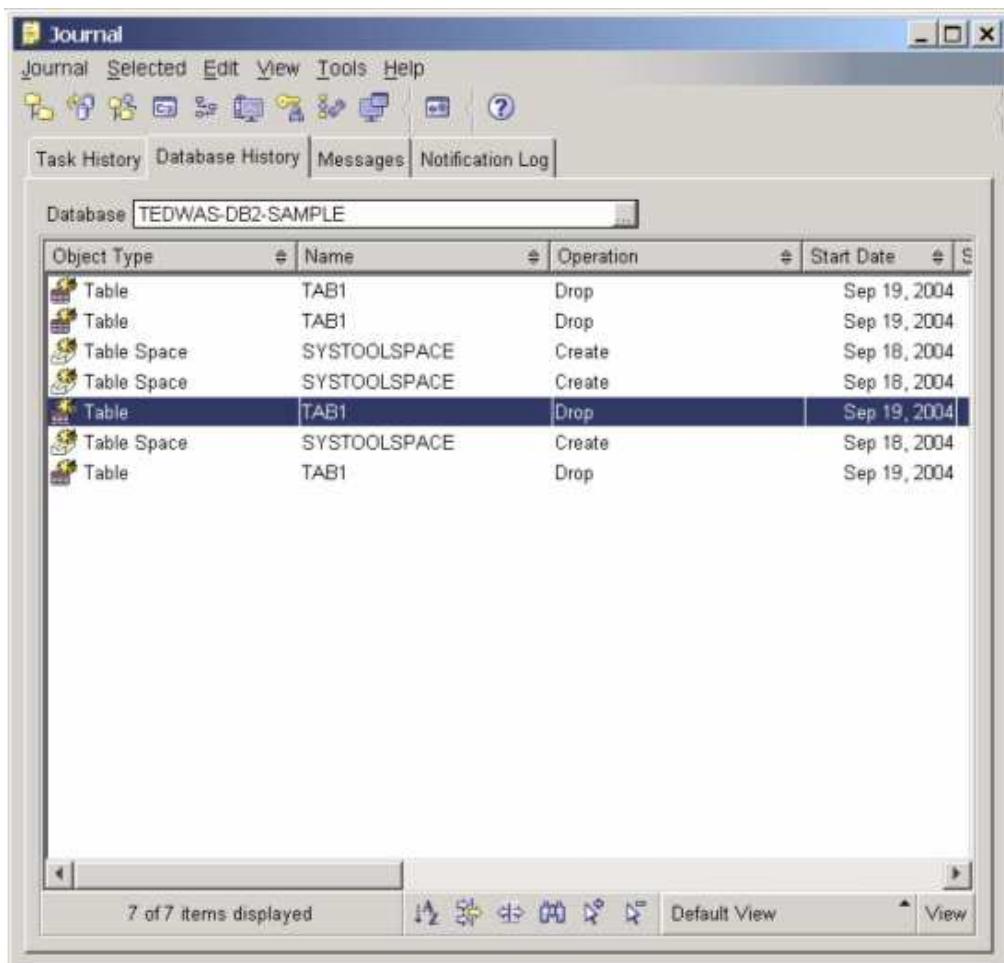


Рисунок 5.18. Журнал

| Тип информации | Описание |
|---------------------|--|
| История задач | Все запускающиеся по расписанию задачи и статус их выполнения |
| История базы данных | Запись всех операций с базой данных (резервное копирование, восстановление, REORG и пр.) |
| Сообщение | История сообщений инструментов DB2. Это полезно, если необходимо вспомнить и сравнить старые сообщения об ошибках, а также в случае преждевременного или случайного закрытия диалогового окна. |
| Журнал уведомлений | Содержит сообщения системного уровня. Здесь записываются критические ошибки. |

Таблица 5.2. Информация, представленная в Журнале

5.7.1 Запуск Журнала

Журнал можно запустить из Центра управления, нажав *Инструменты -> Журнал*, как показано на *рис. 5.19*. Этот инструмент можно также запустить из меню Пуск в Windows: *Пуск -> Все программы -> IBM DB2 -> DB2COPY1(Default) -> General Administration Tools -> Journal* (DB2COPY1(по умолчанию) -> Общие инструменты управления -> Журнал).

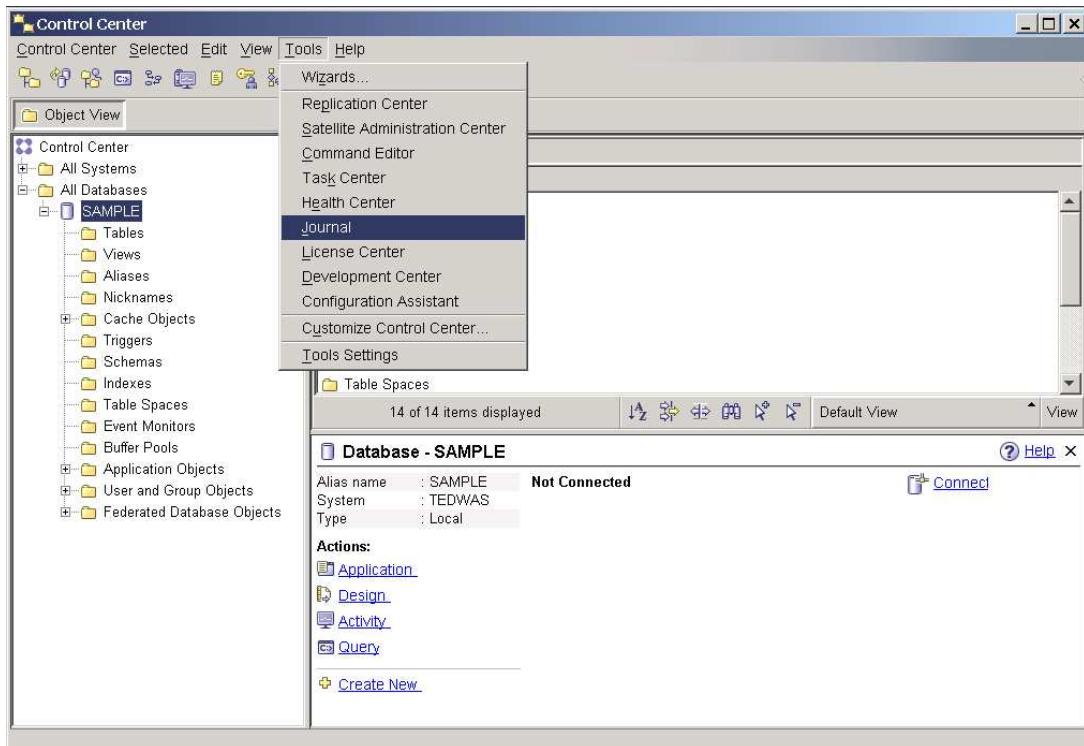


Рисунок 5.19. Запуск Журнала

5.8 Монитор работоспособности (устарело)

Монитор работоспособности — это используемый по умолчанию агент, работающий в ядре DB2 и контролирующий все аспекты работоспособности базы данных (память, управление пространством, предварительно заданные автоматизированные операции и пр.). Если какая-либо часть DB2 работает вне заданных параметров, возникает исключение, о котором уведомляется администратор базы данных. Существует три типа оповещений:

- Внимание: ненормальное состояние
- Предупреждение: некритическое состояние, не требующее немедленных действий, но, возможно, указывающее на неоптимальную работу системы
- Тревога: критическое состояние, требующее немедленного действия

Монитор работоспособности можно включить или выключить с помощью параметра конфигурации менеджера базы данных HEALTH_MON.

5.8.1 Центр работоспособности (устарело)

Центр работоспособности — это графический инструмент для взаимодействия с Монитором работоспособности. Центр работоспособности анализирует оповещения в системе на уровне экземпляра, базы данных и табличного пространства. На рис. 5.20 показан Центр работоспособности.

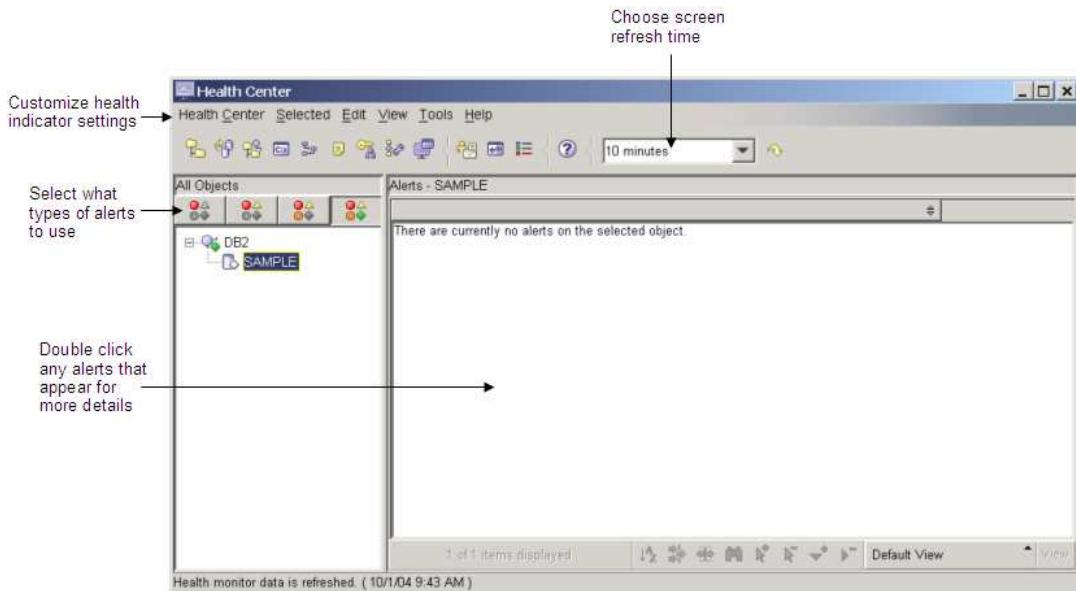


Рисунок 5.20. Центр работоспособности

5.8.1.1 Запуск Центра работоспособности

Центр работоспособности можно запустить через Центр управления, выбрав *Инструменты -> Центр работоспособности*. Это показано на рис. 5.21. Этот инструмент можно также запустить из меню *Пуск -> Все программы -> IBM DB2 -> DB2COPY1(Default) -> Monitoring Tools -> Health Center* (DB2COPY1(по умолчанию) -> Инструменты мониторинга -> Центр работоспособности).

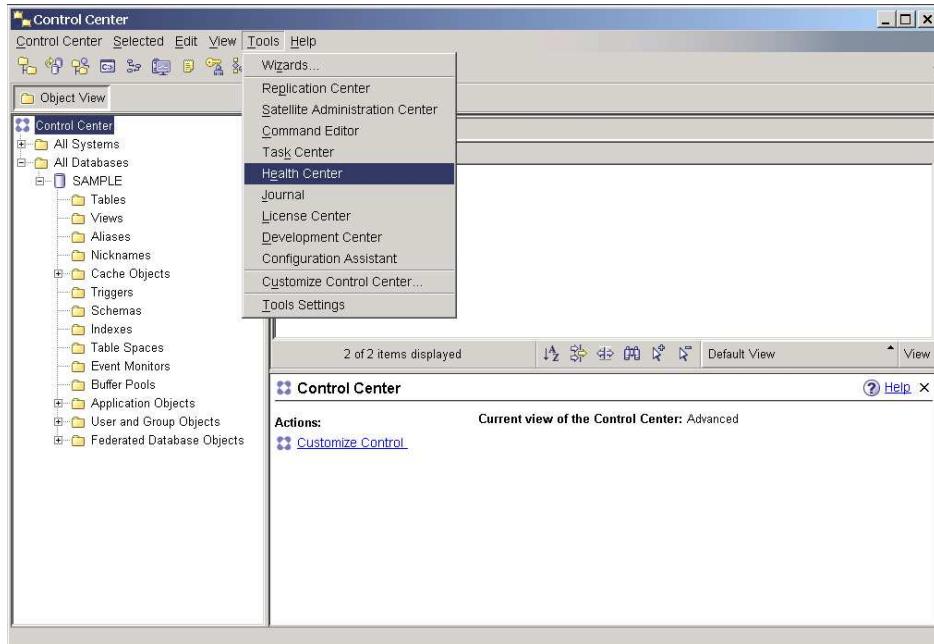


Рисунок 5.21. Запуск Центра работоспособности

5.8.1.2 Конфигурирование уведомления о работоспособности

Запустив Центр работоспособности, можно настроить уведомление, нажав меню *Центр работоспособности -> Конфигурирование -> Уведомление*, как показано на рис. 5.22. В уведомлении можно задать имена, адреса электронной почты и номера пейджеров контактных лиц, которые будут оповещены о появлении уведомлений.

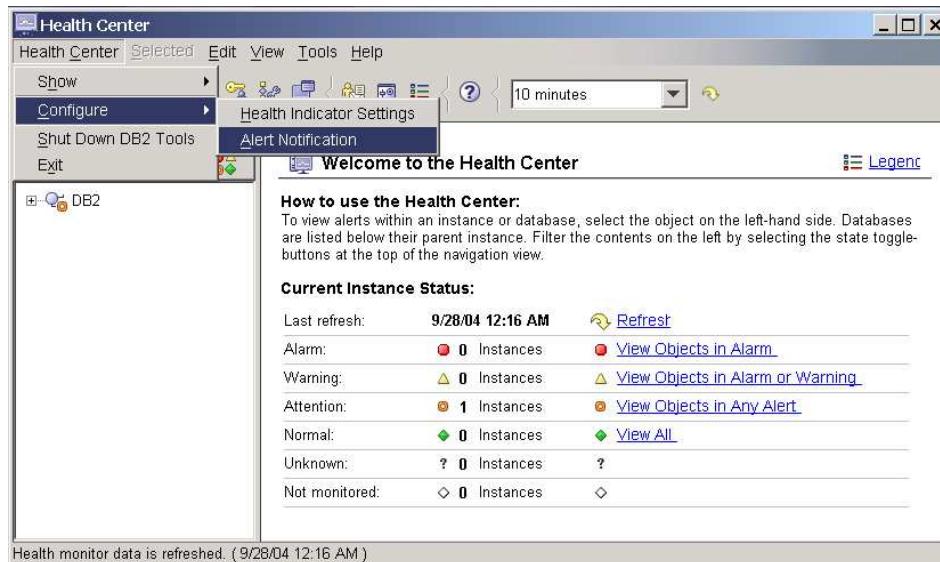


Рисунок 5.22. Уведомление

5.9 Самонастраивающийся менеджер памяти

Представленный в DB2 9 самонастраивающийся менеджер памяти (STMM) — это одна из нескольких автономных вычислительных функций, которые упрощают задачу конфигурирования памяти, автоматически устанавливая значения нескольких параметров конфигурации памяти. Если эта функция включена, автоматический настройщик динамически распределяет доступные ресурсы памяти между несколькими потребителями памяти в базе данных. Настройщик памяти реагирует на изменения характеристик рабочей нагрузки, регулируя значения параметров конфигурирования памяти и размер буферных пулов для оптимизации производительности. Чтобы включить STMM, установите для параметра db cfg SELF_TUNING_MEM значение ON.

Прочие автономные вычислительные функции, такие как автоматическое обслуживание и автоматическое хранение, рассматриваются в других разделах этой книги.

5.10 Использование сценариев

Всегда полезно иметь возможность создать файлы сценариев для повторного выполнения нескольких команд или SQL-операторов DB2. К примеру, администратору базы данных может понадобиться ежедневно выполнять определенный сценарий для проверки количества строк в важных таблицах. Существует два основных вида сценариев:

- сценарии SQL;
- сценарии операционной системы (командного процессора).

5.10.1 Сценарии SQL

Сценарии SQL включают операторы запросов и команды баз данных. Эти сценарии довольно просты и не зависят от платформы. Однако в них не поддерживаются переменные или параметры. К примеру, показанные в *Листинге 5.1* ниже команды хранятся в файле `script1.db2`.

```
CONNECT TO EXPRESS;
CREATE TABLE user1.mytable
  ( col1 INTEGER NOT NULL,
    col2 VARCHAR(40),
    col3 DECIMAL(9,2));
SELECT * FROM user1.mytable FETCH FIRST 10 ROWS ONLY;
COMMIT;
```

Листинг 5.1. Образец сценария SQL, хранящегося в файле script1.db2

В приведенном выше сценарии все операторы являются операторами SQL, и каждый из них отделен разделителем, в данном случае — точкой с запятой. Имя файла необязательно должно иметь расширение «`db2`». Можно использовать любое расширение.

5.9.1.1 Выполнение сценариев SQL

Сценарий SQL можно выполнить через Редактор команд или командное окно DB2 в Windows, или же через командный процессор Linux. Для выполнения сценария, показанного в *Листинге 5.1*, через командное окно DB2 или командный процессор Linux можно воспользоваться следующей командой:

```
db2 -t -v -f script1.db2 -z script1.log
```

либо эквивалентной командой:

```
db2 -tvf script1.db2 -z script1.log
```

В этой команде:

- t означает, что операторы используют символ завершения оператора по умолчанию (точку с запятой);
- v означает режим расширенного вывода; заставляет db2 отображать ход выполнения текущей команды;
- f означает, что имя файла, указанное после этой метки, является файлом сценария;
- z означает, что имя файла, указанное после этой метки, используется для добавления в него вывода для дальнейшего анализа (данний вариант является рекомендуемым, хотя и не обязательным).

Когда применяется метка `-t` и не указывается разделитель строки, разделителем операторов считается точка с запятой. Иногда необходимо использовать другой разделитель. К примеру, в сценарии с программным кодом SQL PL потребуется другой символ завершения оператора вместо точки с запятой (по умолчанию),

поскольку точки с запятой используются в определениях объектов SQL PL для завершения процедурных операторов.

К примеру, файл сценария functions.db2, показанный в *Листинге 5.2* ниже, содержит оператор для создания функции, и точка с запятой используется в конце оператора SELECT как часть синтаксической структуры функции. В качестве завершающего символа оператора CREATE FUNCTION используется восклицательный знак (!). Если в качестве завершающего символа оператора использовалась бы точка с запятой, в сценарии возник бы конфликт с символом, используемым для оператора SELECT, в результате чего возникла бы ошибка DB2.

```
CREATE FUNCTION f1()
  SELECT ... ;
...
END!
```

Листинг 5.2. Содержимое файла сценария functions.db2

Чтобы сообщить DB2 об использовании другого символа завершения оператора, воспользуйтесь меткой -d с указанием соответствующего символа, как показано ниже:

```
db2 -td! -v -f functions.db2 -z functions.log
```

Чтобы получить более подробную информацию о прочих метках, которые можно использовать в командном окне или командном процессоре Linux, выполните следующую команду:

```
db2 list command options
```

5.10.2 Сценарии операционной системы (командного процессора)

Сценарии операционной системы предоставляют больший уровень гибкости и мощности, чем сценарии SQL, поскольку дают возможность добавлять дополнительную программную логику. Они зависят от платформы, но поддерживают параметры и переменные. В *Листинге 5.3* показан пример простого сценария операционной системы Windows (командного процессора).

```
set DBPATH=C:
set DBNAME=PRODEXPR
set MEMORY=25
db2 CREATE DATABASE %DBNAME% ON %DBPATH% AUTOCONFIGURE USING
      MEM_PERCENT %MEMORY% APPLY DB AND DBM
db2 CONNECT TO %DBNAME% USER %1 USING %2
del schema.log triggers.log app_objects.log
db2 set schema user1
db2 -t -v -f schema.db2 -z schema.log
db2 -td@ -v -f triggers.db2 -z triggers.log
db2 -td@ -v -f functions.db2 -z functions.log
```

Листинг 5.3. Содержимое файла сценария операционной системы create_database.bat

Чтобы выполнить этот сценарий операционной системы из командной строки, в Windows необходимо выполнить следующую команду:

```
create_database.bat db2admin ibmdb2
```

где *db2admin* — это идентификатор пользователя и первый параметр сценария, а *ibmdb2* — пароль и второй параметр сценария.

В Windows расширение «*bat*» показывает операционной системе, что это выполняемый пакетный файл. В Linux необходимо изменить режим для файла, например с помощью команды *chmod +x*, чтобы указать, что файл выполняемый. После этого файл можно запустить, как описано выше.

5.11 Особенности применения в Windows Vista

Из-за функции управления доступом на уровне пользователей (UAC) в Windows Vista приложения запускаются со стандартными правами, даже если используемый идентификатор пользователя принадлежит локальному администратору. Иными словами, запускаемые в Vista инструменты или команды DB2 будут работать, но могут возникать проблемы, связанные с доступом. Этой проблемы можно избежать, воспользовавшись ярлыком «Command window — Administrator» (Командное окно — Администратор), специально создаваемым при установке для пользователей Vista. В открывшемся окне можно выполнять прочие команды и запускать другие инструменты (используя команды, приведенные в Таблице 5.1 в начале этой главы). Также, воспользовавшись меню Пуск в Windows Vista или любым ярлыком DB2, можно найти инструмент DB2, который требуется запустить, щелкнуть на нем правой кнопкой мыши и выбрать вариант Запуск от имени администратора.

Если по умолчанию включена расширенная безопасность DB2 (подробную информацию см. в Главе 10 «Безопасность базы данных»), для запуска таких графических инструментов, как Центр управления, необходимо также убедиться в том, что идентификатор пользователя входит в группу DB2ADMNS.

5.12 Краткий обзор

В этой главе мы рассмотрели широкий набор инструментов, доступных для администрирования, конфигурирования и управления сервером данных DB2.

Появление IBM Data Studio в DB2 9.7 в качестве основного инструмента администрирования открывает новые возможности для администрирования и разработки баз данных.

Также рассматривался ряд устаревших GUI-инструментов: Центр управления, мастер SQL Assist, Центр задач, Журнал, Агент и Монитор работоспособности. Однако инструменты процессора командной строки и командного окна и в дальнейшем будут входить в состав наших продуктов, даже в версиях после DB2 9.7. Также самонастраивающийся диспетчер памяти остается весомой составляющей процесса оптимизации баз данных.

Ключевым элементом инструментария каждого администратора баз данных является использование сценариев для выполнения команд и функций DB2. В этой главе мы

подробно рассмотрели SQL-сценарии и сценарии операционной системы (командного процессора), в частности процессы их создания, хранения и выполнения.

Мы подытожили главу информацией о том, как обеспечить беспрепятственную работу инструментов DB2 в Windows Vista.

5.13 Упражнения

Упражнения в этой главе дадут возможность поупражняться в работе со сценариями в DB2.

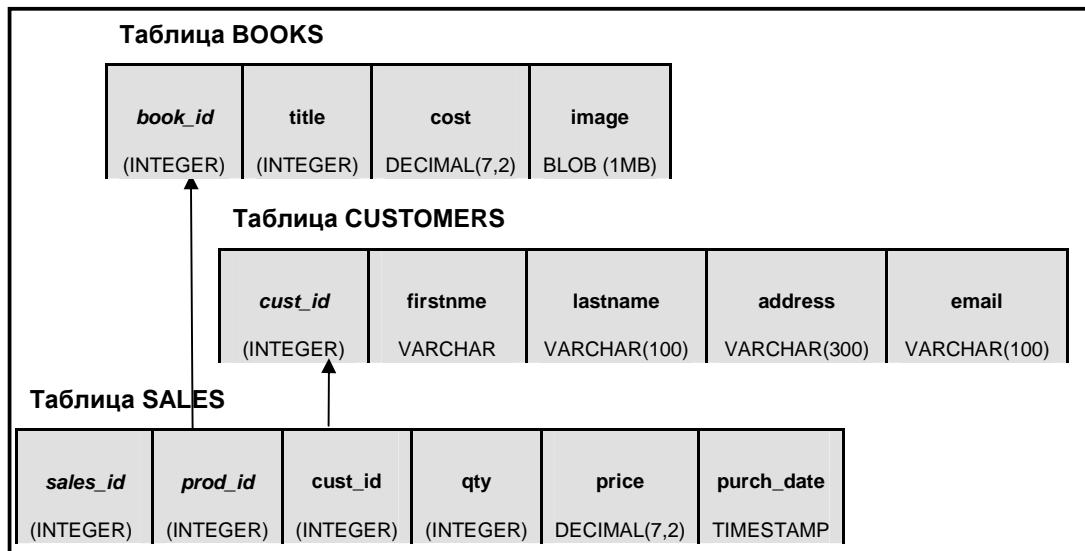
Часть 1. Заполнение базы данных EXPRESS с помощью сценариев

В этой части упражнений необходимо заполнить базу данных EXPRESS (созданную ранее), используя Редактор команд и два предоставленных сценария.

Процедура

1. Заполните базу данных EXPRESS несколькими таблицами и данными. Для удобства, эту задачу можно выполнить с помощью двух предварительно созданных сценариев, `Lab_Chpt5.db2` и `Lab_Chpt5.dat`. Сценарий `Lab_Chpt5.db2` содержит команды, используемые для создания таблиц, а потому его необходимо выполнить первым. Сценарий `Lab_Chpt5.dat` содержит операторы, вставляющие данные в таблицы. Оба сценария можно найти в файле `expressc_book_exercises_9.7.zip`, прилагаемом к этой книге. Для выполнения сценариев откройте Редактор команд. Убедитесь, что в раскрывающемся списке на панели инструментов выбрана только что созданная база данных. Если этой базы данных нет в списке, добавьте подключение к ней с помощью кнопки *Добавить*.
2. В Редакторе команд выберите пункт меню *Выделенное → Открыть* и найдите папку, в которой хранятся сценарии. Выберите файл `Lab_Chpt5.db2` и нажмите кнопку *OK*. Теперь содержимое файла должно отображаться в области ввода Редактора команд. Нажмите кнопку *Выполнить*, чтобы выполнить сценарий. Убедитесь в том, что сценарий выполнен без ошибок.
3. Повторите шаг (2) для файла `Lab_Chpt5.dat`.

Созданная база данных — это очень простая база данных книжного интернет-магазина. В таблице `BOOKS` (книги) содержится вся информация о предлагаемых магазином книгах. В таблице `CUSTOMERS` (клиенты) содержится информация о каждом из клиентов магазина. Наконец, в таблице `SALES` (продажи) содержатся данные продаж. Каждый раз, когда клиент совершает покупку, в таблицу `SALES` вносится запись. На диаграмме ниже показана структура таблиц и отношения между ними.



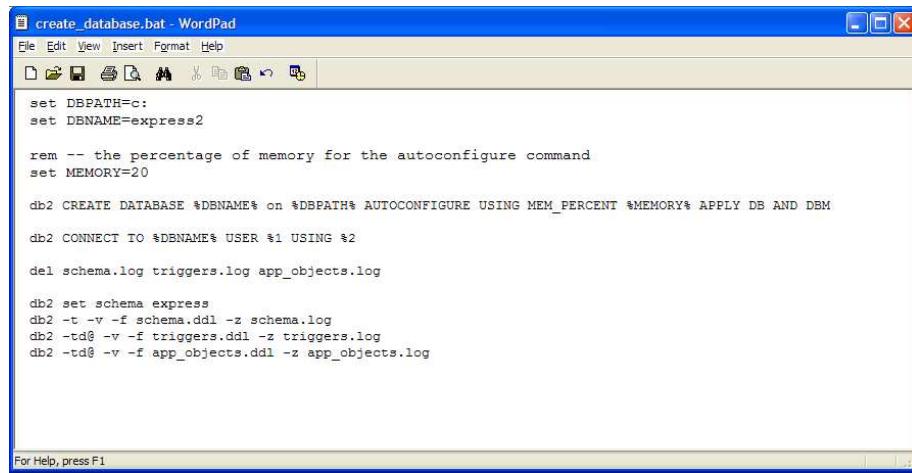
Часть 2. Создание сценария установки базы данных EXPRESS

Сценарии — это мощный механизм выполнения повторяющихся задач, таких как сбор статистики, резервное копирование и развертывание баз данных. Преимуществом сценариев операционной системы является поддержка параметров сценариев, благодаря чему они имеют большую гибкость. В этой части будет создан сценарий операционной системы для повторного развертывания базы данных **EXPRESS** как базы данных **EXPRESS2**. Данный сценарий будет обращаться к предварительно созданным SQL-сценариям за объектами базы данных. Для краткости в этом упражнении описаны сценарии и команды только для платформы Windows. Если вы предпочитаете Linux, внесите соответствующие корректизы в приведенные ниже инструкции.

Процедура

1. Откройте текстовый редактор и введите информацию, как показано ниже. Печатая показанные ниже строки, большинство пользователей допускает ошибки. Мы преднамеренно не предоставляем этот текст в отдельном файле, чтобы вы научились самостоятельно исправлять допущенные ошибки.

Также обратите внимание, что необходимо указать правильный путь к файлам `schema.ddl`, `triggers.ddl` и `app_objects.ddl`, которые содержатся в файле `expressc_book_exercises_9.7.zip`, прилагаемом к этой книге.



```

create_database.bat - WordPad
File Edit View Insert Format Help
New Open Save Print Find Replace Undo Redo
set DBPATH=c:
set DBNAME=express2

rem -- the percentage of memory for the autoconfigure command
set MEMORY=20

db2 CREATE DATABASE $DBNAME$ on $DBPATH$ AUTOCONFIGURE USING MEM_PERCENT $MEMORY% APPLY DB AND DBM

db2 CONNECT TO $DBNAME$ USER %1 USING %2

del schema.log triggers.log app_objects.log

db2 set schema express
db2 -t -v -f schema.ddl -z schema.log
db2 -td@ -v -f triggers.ddl -z triggers.log
db2 -td@ -v -f app_objects.ddl -z app_objects.log

For Help, press F1

```

2. Сохраните файл сценария в каталоге, например C:\express, под именем create_database.bat.

Примечание. Если используется Wordpad, в диалоговом окне *Сохранить как* обязательно выберите вариант *Формат MS-DOS*. Если сохранить файл в другом формате, Wordpad может вставить невидимые символы, которые помешают выполнению сценария. Кроме того, возмите имя файла в кавычки, как показано на рисунке ниже, иначе Windows может присоединить к нему расширение .TXT.

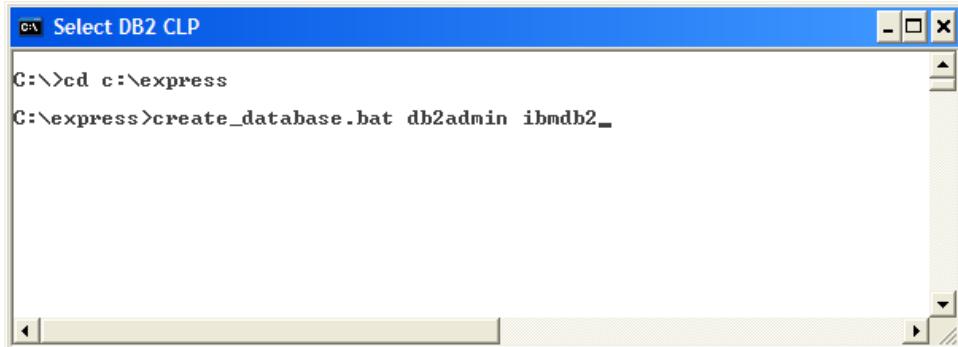


3. Для выполнения сценариев, взаимодействующих с DB2, понадобится среда командной строки DB2. Откройте командное окно DB2, выбрав *Пуск -> Все программы -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Window (DB2COPY1 (по умолчанию) -> Инструменты командной строки -> Командное окно). Также можно открыть меню Пуск -> Выполнить, ввести db2cmd и нажать кнопку OK, как показано ниже.*



4. Затем, чтобы выполнить сценарий, введите следующие команды в командном окне:

```
cd C:\express
create_database.bat db2admin ibmdb2
```



5. Внимательно ознакомьтесь со сценарием, который мы только что создали. Понимаете ли вы, для чего предназначена каждая из строк?
6. Ответьте на такие вопросы:
- Какая строка устанавливает соединение с базой данных?
 - Что означает %1 и %2?
 - Что делает следующая строка кода? Где она используется? Для чего?
SET DBPATH=C:
 - Что делает следующая строка кода?
del schema.log, triggers.log, app_objects.log
 - Что произойдет, если вызвать сценарий без параметров?
 - Почему в сценариях SQL нет команды CONNECT TO? Как такие сценарии подключаются к базе данных?

ЧАСТЬ II. ИЗУЧЕНИЕ DB2: АДМИНИСТРИРОВАНИЕ БАЗЫ ДАННЫХ

6

Глава 6. Архитектура DB2

В этой главе кратко описана архитектура DB2. Будут рассмотрены:

- Модель процессов DB2
- Модель памяти DB2
- Модель хранения данных DB2

Примечание.

Чтобы получить более подробную информацию об архитектуре DB2, просмотрите видео: <http://www.channeldb2.com/video/video/show?id=807741:Video:4482>

6.1 Модель процессов DB2

На *рис. 6.1* изображена модель процессов DB2. На этом рисунке прямоугольники обозначают процессы, а овалы — порождаемые подпроцессы. Главный процесс DB2 называется db2sysc. Этот процесс имеет несколько подпроцессов, главный из которых также называется db2sysc. Это основной подпроцесс, порождающий остальные подпроцессы. Когда удаленное приложение пытается подключиться к серверу с помощью SQL-оператора CONNECT, удаленные средства отслеживания протокола обмена данными получают соответствующий запрос и связываются с координирующим агентом DB2 (db2agent). Агент DB2 является своего рода маленьким работником, выполняющим операции от имени DB2. В случае локального приложения, т. е. приложения, работающего на одном сервере с DB2, процесс проходит очень похожим образом, но запрос обрабатывается агентом db2ipccm, а не подпроцессом db2tcpccm. В некоторых случаях, к примеру если включен параллелизм, db2agent может порождать других агентов, которые отображаются как подпроцессы db2agntp. Прочие показанные на рисунке агенты, например db2pfchr, db2loggr, db2dlock, также могут использоваться в разных целях. Наиболее распространенные процессы описаны в *Таблице 6.1*, а подпроцессы — в *Таблице 6.2*.

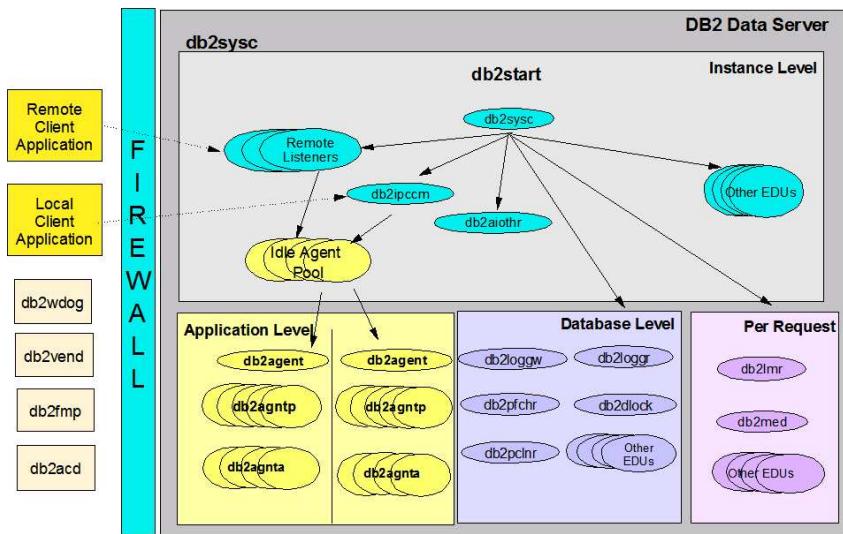


Рисунок 6.1. Модель процессов DB2

| Имя процесса | Описание |
|-----------------------------------|--|
| db2sysc (Linux) db2syscs (Win) | Главный системный контроллер или ядро DB2. Начиная с DB2 9.5, во всем разделе выполняется только один главный процесс ядра с несколькими подпроцессами. Все управляемые единицы ядра (Engine Dispatchable Units, EDU) являются подпроцессами этого процесса. Без этого процесса сервер данных функционировать не может. |
| db2acd | Автономная вычислительная служба-демон. Используется для выполнения автоматических задач со стороны клиента, например, системы мониторинга состояния, служб автоматического обслуживания и планировщика администрирования. Раньше этот процесс назывался db2hmon. |
| db2wdog | Схема безопасности DB2. Схема безопасности является родительским элементом для главного процесса ядра — db2sysc. Она освобождает ресурсы при неправильном завершении процесса db2sysc. |
| db2vend | Защищенный сторонний процесс, появившийся в DB2 9.5. Все программные коды сторонних разработчиков выполняются в рамках этого процесса вне ядра. Приложения сторонних разработчиков — это программы, разработанные не компанией IBM, которые могут взаимодействовать с DB2; к примеру, программный код стороннего разработчика сможет управлять архивированием журнала, если настроить параметр операции выхода пользователя на использование этого кода. |
| db2fmp | Защищенные процессы, выполняющие программный код пользователя на сервере вне брандмауэра для хранящих процедур и функций, определенных пользователем. Этот процесс заменяет процессы db2udf и db2dari, которые использовались в предыдущих версиях DB2. |

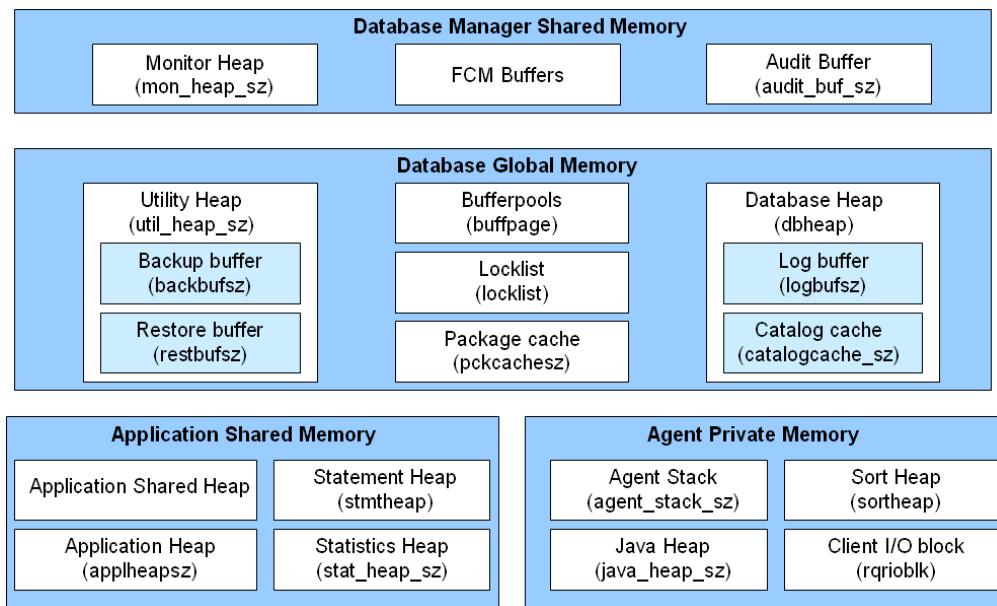
Таблица 6.1. Распространенные процессы DB2

| Имя подпроцесса | Описание |
|-----------------|--|
| db2sysc | Подпроцесс системного контроллера. Этот подпроцесс отвечает за запуск и завершение текущего экземпляра, а также управление им во время работы. |
| db2tcpcm | Средство отслеживания связи TCP/IP. |
| db2agent | Координирующий агент, выполняющий операции базы данных от имени приложений (не меньше 1 для каждого подключения, в зависимости от того, включен ли концентратор соединений). |
| db2agntp | Активный субагент, появляющийся при установке для INTRA_PARALLEL значения YES. Этот подпроцесс выполняет операции базы данных для приложения. db2agent координирует совместную работу различных субагентов db2agntp. |
| db2pfchr | Средство предварительной выборки асинхронного ввода/вывода данных DB2 (NUM_IOSERVERS) |
| db2pcInr | Средство записи асинхронного ввода/вывода данных DB2 (NUM_IOCLEANER) |

Таблица 6.2. Распространенные подпроцессы DB2

6.2 Модель памяти DB2

Модель памяти DB2 состоит из различных областей памяти на уровне экземпляра, базы данных, приложения и агента (см. рис. 6.2). В этой книге мы не будем подробно рассматривать каждую из областей памяти, а ограничимся их кратким обзором.

**Рисунок 6.2. Модель памяти DB2**

При запуске экземпляра распределяется общая память менеджера базы данных. Каждому экземпляру присваивается один участок такой памяти. Обычно это не требует большого пространства — в среднем требуется менее 100 МБ. При первом подключении к базе данных распределяется глобальная память базы данных. Каждой базе данных, к которой выполняется подключение, присваивается один участок такой памяти. В этом блоке буферный пул является одной из наиболее важных частей, особенно для повышения производительности запросов. Размер буферных пулов определяет размер целой глобальной памяти базы данных.

Общая память приложений — это область, в которой приложения и агенты DB2 могут обмениваться такой информацией, как разделы планов доступа, компиляция SQL или XQuery, сбор статистических данных и пр.

Частная память агента — это память, используемая каждым агентом DB2. Если не используется концентратор соединений, для каждого соединения требуется один агент. Обычно агент использует приблизительно 3–5 МБ. С помощью концентратора соединений для нескольких соединений может использоваться один агент, благодаря чему уменьшается потребность в физической памяти.

6.3 Модель хранения данных DB2

В этом разделе описаны следующие понятия:

- страницы и экстенты;
- буферный пул;
- табличное пространство.

6.3.1 Страницы и экстенты

Страница — это наименьшая единица хранения в DB2. Допустимые размеры страниц: 4 КБ, 8 КБ, 16 КБ и 32 КБ. Экстент — это группа страниц. Работа с одной страницей за раз в DB2 негативно сказывалась на производительности; поэтому DB2 работает с экстентами. Размер страницы и экстента определяется при работе с буферными пулами и табличными пространствами, как описано в последующих разделах.

6.3.2 Буферные пулы

Буферные пулы — это реальная кэш-память для данных таблиц и индексов. Буферный пул повышает производительность, сокращая непосредственный последовательный ввод/вывод, а также способствует асинхронному считыванию (предварительному отбору) и записи. Иными словами, DB2 предвидит, какие потребуются страницы, и предварительно отбирает их с диска в буферный пул, чтобы они были готовы к использованию.

Буферные пулы распределяются в единицах памяти со страницами 4 КБ, 8 КБ, 16 КБ и 32 КБ. Каждая база данных должна иметь хотя бы один буферный пул, а также хотя бы один совпадающий буферный пул для табличного пространства с заданным размером страницы.

6.3.2.1 Создание буферного пула

Для создания буферного пула можно воспользоваться оператором `CREATE BUFFERPOOL`. Или же можно в Центре управления щелкнуть правой кнопкой мыши папку «Буферные пулы» в соответствующей базе данных и выбрать пункт *Создать*, как показано на рис. 6.3.

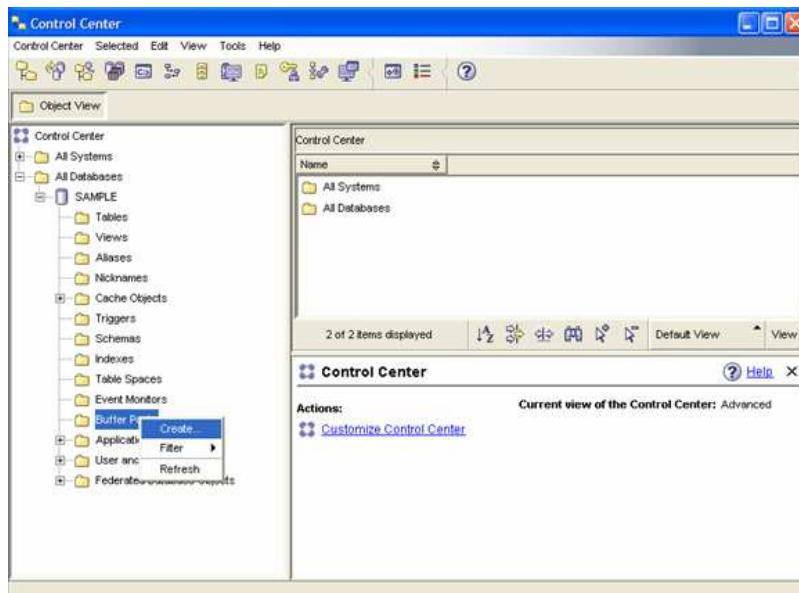


Рисунок 6.3. Создание буферного пула

После выбора пункта Создать откроется диалоговое окно «Создать буферный пул» (см. рис. 6.4).

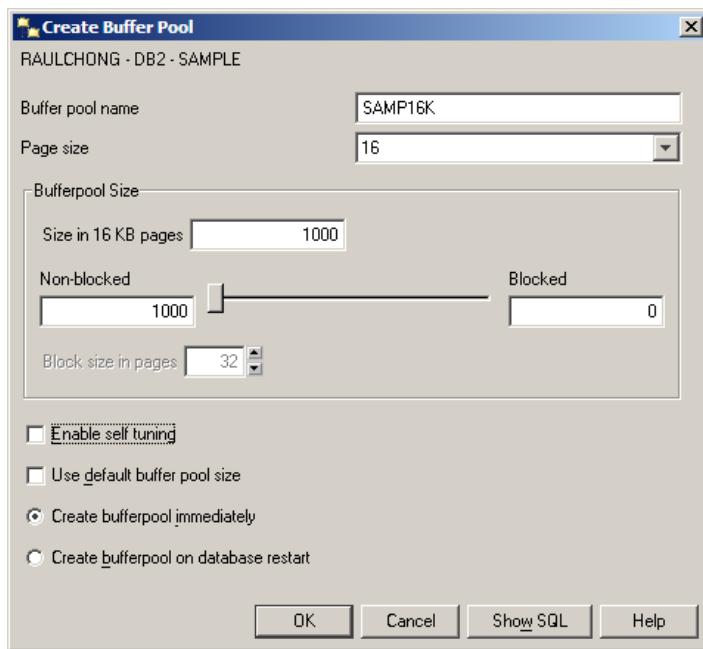


Рисунок 6.4. Диалоговое окно «Создать буферный пул»

Большинство записей на рис. 6.4 не требуют дополнительных объяснений. Поля **Неблокировано** и **Блокировано** относятся к количеству неблокированных и блокированных страниц. Буферный пул на основе блокированных страниц обеспечивает перемещение в буферный пул страниц, смежных на диске, в таком же смежном порядке в блочной области; это может повысить производительность. Количество таких страниц не должно превышать 98% общего количества страниц буферного пула. Значение 0 отключает блочный ввод/вывод.

Созданный буферный пул отображается в Центре управления (см. рис. 6.5).

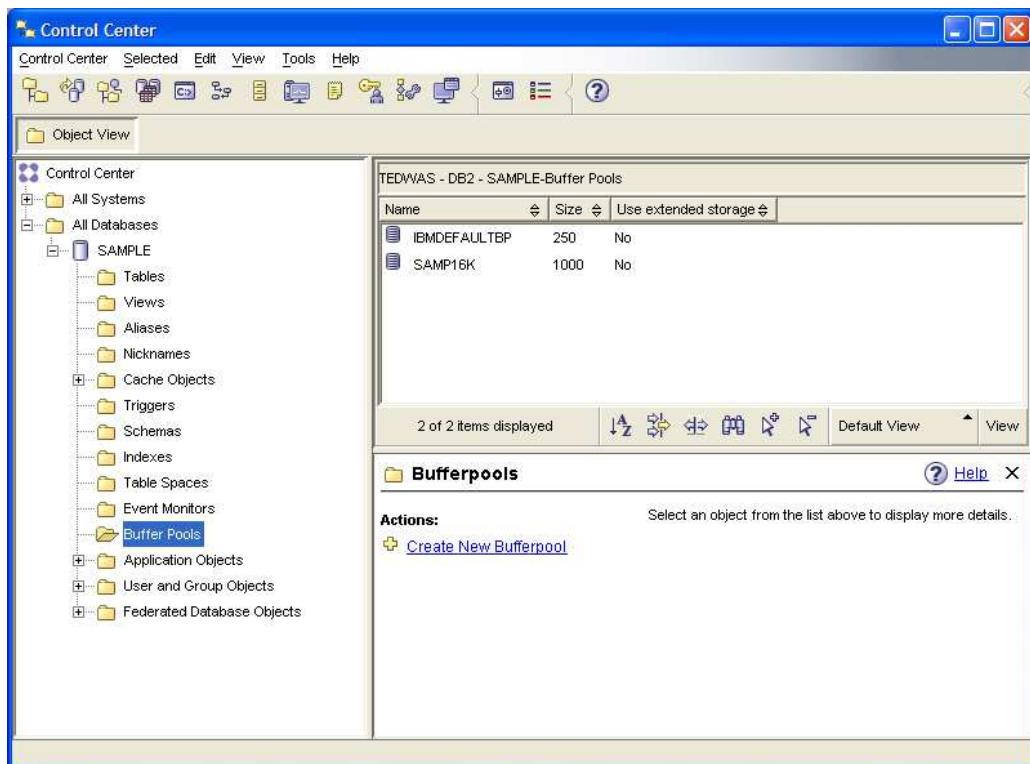


Рисунок 6.5. Центр управления после создания буферного пула SAMP16K

6.3.3 Табличные пространства

Табличные пространства — это логический интерфейс между логическими таблицами, физической памятью (буферным пулом) и контейнерами (дисками) системы. Воспользуйтесь оператором `CREATE TABLESPACE`, чтобы создать табличное пространство, для которого можно указать:

- размер страницы табличного пространства (4 КБ, 8 КБ, 16 КБ или 32 КБ). Размер страницы должен соответствовать размеру страницы буферного пула;
- имя буферного пула, привязанного к данному табличному пространству;
- размер экстента;
- размер предварительной выборки.

6.3.3.1 Типы табличных пространств

Существует три типа табличных пространств:

- Обычные

Предназначены для пользовательских таблиц. К примеру, созданное по умолчанию табличное пространство USERSPACE1 является обычным табличным пространством.

- Крупные

Используются (опционально) для отделения данных LOB в собственные табличные пространства. Также применяются для хранения данных XML в базах данных, созданных с поддержкой pureXML и использующих тип данных XML в столбцах. Крупные табличные пространства используются по умолчанию.

- Временные

Существует два типа временных табличных пространств:

- Системные временные

Используются DB2 для внутренних операций, таких как сортировка. К примеру, созданное по умолчанию при создании базы данных табличное пространство TEMPSPACE1 является системным временным табличным пространством. Обязательно должно существовать хотя бы одно системное временное табличное пространство.

- Пользовательские временные

Используются для создания пользовательских объявленных глобальных временных таблиц (Declared Global Temporary Tables, DGTT) и созданных глобальных временных таблиц (Create Global Temporary Tables, CGTT), которые являются временными таблицами, хранящимися в оперативной памяти. Их часто путают с системными временными табличными пространствами. Пользователям необходимо создать пользовательское временное табличное пространство, чтобы иметь возможность использовать DGTT или CGTT.

6.3.3.2 Управление табличными пространствами

Табличные пространства можно классифицировать по типу управления. Это можно указать в операторе **CREATE TABLESPACE**.

Табличные пространства под управлением системы

Этот тип табличных пространств известен как System Managed Storage (SMS). Это означает, что хранилищем управляет операционная система. Такими табличными пространствами легко управлять, и контейнерами для них служат каталоги файловой системы. Пространство не распределяется предварительно, файлы увеличиваются динамически. При определении контейнеры фиксируются на момент создания, и новые контейнеры можно добавить, только если воспользоваться перенаправленным восстановлением. При использовании табличных пространств типа SMS табличные данные, индексы и данные LOB не могут распределяться в разных табличных пространствах.

Табличные пространства под управлением базы данных

Этот тип табличных пространств известен как Database Managed Storage (DMS). Это означает, что хранилищем управляет DB2. Управление пространством требует большего вмешательства со стороны администратора баз данных. В качестве контейнеров могут использоваться предварительно распределенные файлы или устройства непосредственного доступа. При использовании устройств непосредственного доступа данные записываются, не попадая в кэш-память операционной системы.

Контейнеры можно добавлять, удалять или менять с помощью оператора ALTER TABLESPACE. Табличные пространства типа DMS являются лучшим выбором с точки зрения производительности; табличные данные, индексы и данные LOB можно разбивать между несколькими табличными пространствами, что повышает производительность.

Табличные пространства под управлением автоматического хранилища

Этот тип табличных пространств работает под управлением автоматического хранилища и характеризуется простотой использования (как в табличных пространствах типа SMS), но с наивысшей производительностью и гибкостью (как в табличных пространствах типа DMS). Соответственно, начиная с DB2 9, именно этот тип табличных пространств используется по умолчанию. Для таких табличных пространств пользователь сначала указывает путь хранилища и логическую группу устройств хранения данных, которые будут использоваться DB2 в процессе управления. Четкие определения контейнеров не задаются. Контейнеры автоматически создаются на пути хранилища. Увеличением существующих и добавлением новых контейнеров полностью управляет DB2. Если путь хранилища не указан в команде CREATE DATABASE, в качестве пути хранилища используется путь базы данных. Путь базы данных — это место хранения основных определений базы данных. Если путь базы данных не указан, он определяется на основе параметра конфигурации менеджера базы данных DFTDBPATH. В Windows в этой роли может использоваться только диск, а не путь.

Чтобы разрешить применение автоматического хранилища, для начала нужно создать базу данных с включенным автоматическим хранилищем (используется по умолчанию) и привязать к ней набор путей хранилища. Позже, при необходимости, можно переопределить пути хранилища с помощью операции RESTORE для базы данных. После этого можно создать табличные пространства с автоматическим хранилищем (опять же, этот вариант используется по умолчанию).

Табличные пространства под управлением автоматического хранилища очень похожи на табличные пространства типа DMS, но операции автоматизированы, а потому выполняются под управлением DB2; это относится, в частности, к назначению и распределению контейнеров, а также к автоматическому изменению размера.

Рассмотрим пример табличного пространства под управлением автоматического хранилища. Сначала создайте базу данных с включенным автоматическим хранилищем, как в примерах ниже:

Автоматическое хранилище включено по умолчанию:

```
CREATE DATABASE DB1
```

Автоматическое хранилище четко определено:

```
CREATE DATABASE DB1 AUTOMATIC STORAGE YES
```

Автоматическое хранилище включено по умолчанию, но указаны пути хранилища. Если путь хранилища является каталогом, его необходимо создать заранее:

Пример для Windows:

```
CREATE DATABASE DB1 ON C:/, C:/storagepath1, D:/storagepath2
```

Обратите внимание, что первым элементом в списке является диск, поскольку он соответствует пути базы данных, который в Windows может быть только диском, а не путём. Этот элемент также будет использоваться в качестве одного из путей хранилища. Таким образом, путь базы данных — это C:, а пути хранилища состоят из C:, C:\storagepath1 и D:\storagepath2, при этом два последних каталога необходимо создать заранее.

Пример для Linux:

```
CREATE DATABASE DB1 ON /data/path1, /data/path2
```

Автоматическое хранение явно отключено:

```
CREATE DATABASE DB1 AUTOMATIC STORAGE NO
```

Теперь создайте табличное пространство со включенным автоматическим хранением, как в примерах ниже:

Автоматическое хранение для табличных пространств также включено по умолчанию:

```
CREATE TEMPORARY TABLESPACE TEMPTS
```

Автоматическое хранение четко определено для табличного пространства:

```
CREATE TABLESPACE TS2 MANAGED BY AUTOMATIC STORAGE
```

Автоматическое хранение четко определено, определен начальный размер, а также степень его увеличения и максимальный размер.

```
CREATE TABLESPACE TS1
  INITIALSIZE 500 K
  INCREASESIZE 100 K
  MAXSIZE 100 M
```

6.3.3.3 Хранение данных в табличных пространствах

По умолчанию DB2 выполняет последовательную запись в экстенты диска с «расположением» между контейнерами. К примеру, если имеется табличное пространство на 4 КБ с размером экстента 8 страниц и используется 3 непосредственных контейнера в табличном пространстве типа DMS, это означает, что 32 КБ данных (4 КБ x 8 страниц = 32 КБ) будет записано на один диск, прежде чем начнется запись на следующий. Это показано на рис. 6.6. Обратите внимание на то, что таблицы не используют экстенты совместно.

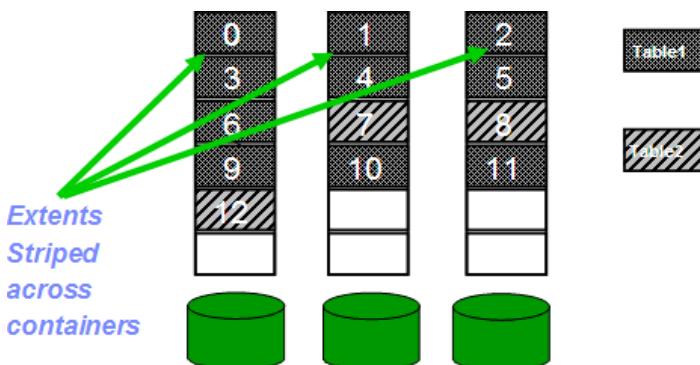


Рисунок 6.6. Запись данных в табличные пространства

6.3.3.4 Создание табличного пространства с помощью Центра управления

Чтобы создать табличное пространство в Центре управления, щелкните правой кнопкой мыши на папке *Табличные пространства* для соответствующей базы данных и выберите пункт *Создать*, как показано на рис. 6.7. Откроется *Мастер по созданию табличных пространств* (см. рис. 6.8).

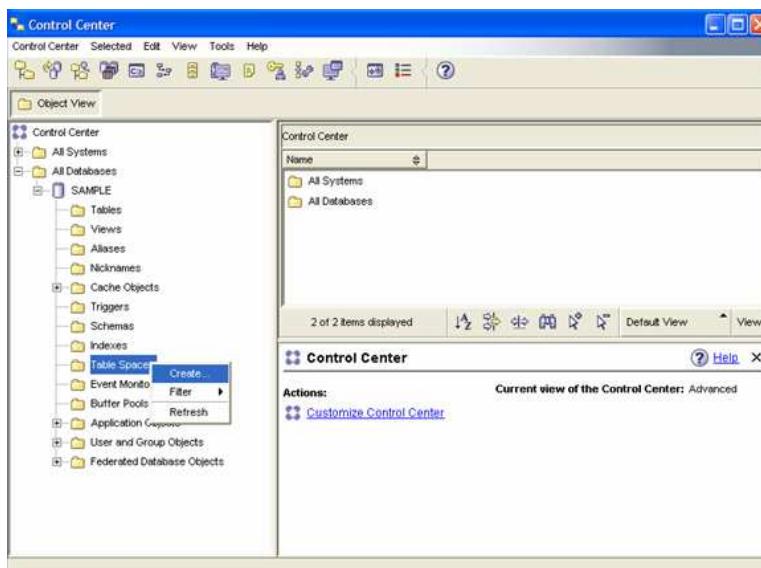


Рисунок 6.7. Создание табличного пространства в Центре управления



Рисунок 6.8. Мастер по созданию табличных пространств

Изображенный на рис. 6.8 мастер поможет выполнить шаги по созданию табличного пространства.

6.4 Краткий обзор

В этой главе рассматривались три ключевых аспекта архитектуры DB2: модель процесса, модель памяти и модель хранилища. Изучая модель процесса, мы рассмотрели распространенные процессы и подпроцессы, в частности процесс db2sysc, без которого работа DB2 была бы невозможна.

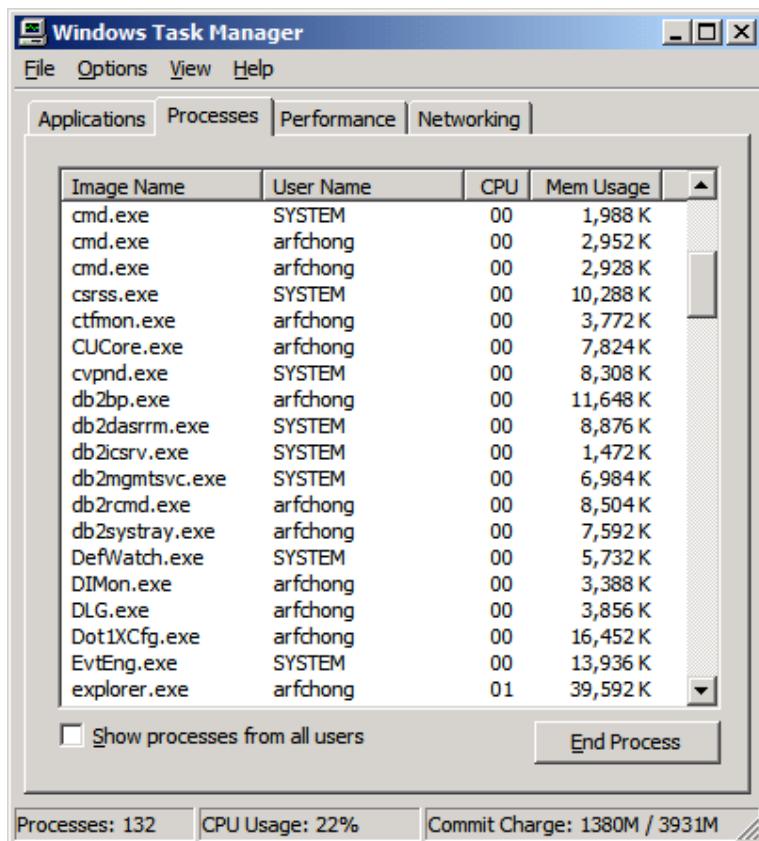
Мы подробно обсудили модель хранилища, охватив три наиболее важных аспекта: страницы и экстенты, буферные пулы (в том числе информацию об их создании) и табличные пространства. В конце главы мы рассмотрели разные типы табличных пространств и способы управления ими (SMS, DMS, автоматические), а также научились создавать табличное пространство с помощью Центра управления.

6.5 Упражнения

Это упражнение поможет понять модель процессов, модель памяти и модель хранилища DB2 в Windows. Мы рассмотрим различные процессы и подпроцессы, использование мониторинга памяти, а также поупражняемся в создании базы данных, использующей автоматические хранилища и пути хранилищ в Windows. В идеале пути хранилища создаются на разных дисках (накопителях), но поскольку не на всех компьютерах есть несколько дисков, в этом упражнении мы будем использовать только диск C:\.

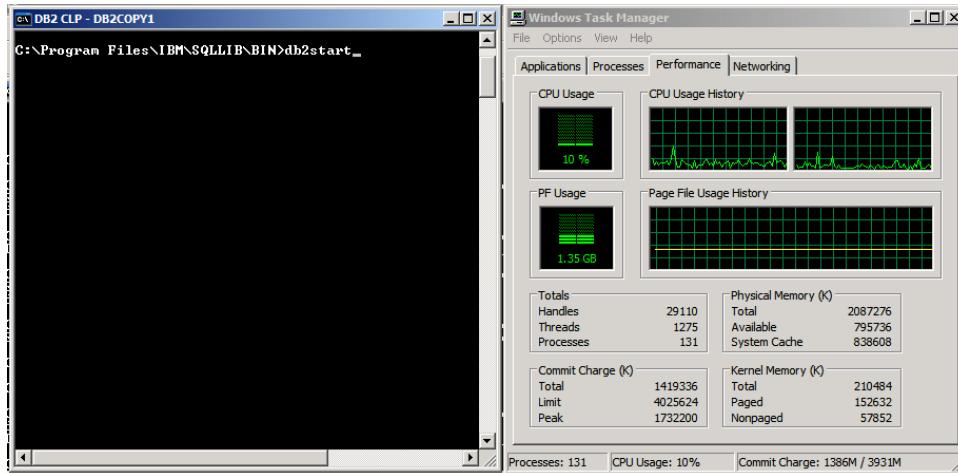
Процедура:

1. Рассмотрим несколько процессов в Windows. Для начала откройте командное окно DB2 (Пуск -> Выполнить -> db2cmd) и убедитесь, что экземпляр остановлен, выполнив следующую команду: **db2stop force**
2. Откройте диспетчер задач Windows и выберите вкладку *Процессы*; поставьте флажок возле пункта *Отображать процессы всех пользователей*. Щелкните столбец «Имя образа», чтобы выполнить по нему сортировку; затем попытайтесь найти процесс db2sysc.exe, как показано на рисунке ниже.

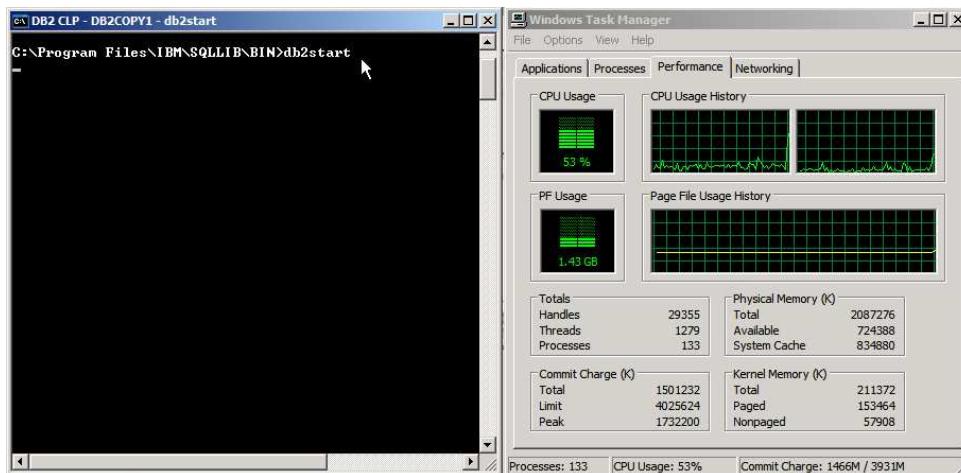


3. Процесс db2syscs.exe не должен отображаться в списке, поскольку мы остановили экземпляр DB2, выполняя шаг 1.
4. Запустите экземпляр DB2 с помощью команды **db2start** и повторите предыдущий шаг. Удалось ли найти в списке процесс db2syscs.exe?
5. Теперь проверим потребление ресурсов ЦП и памяти. Выполните следующие действия:
 - G. Убедитесь в том, что закрыты все прочие приложения в системе
 - H. Откройте новое командное окно DB2 и выполните команду **db2stop force**

- I. В диспетчере задач переключитесь на вкладку «Быстродействие»
- J. Не закрывайте диспетчер задач и командное окно DB2, расположите их рядом, как показано на рисунке ниже. Выпишите объем доступной физической памяти и уровень загрузки ЦП.



- K. Выполните команду `db2start`, при этом наблюдая за загрузкой ЦП и использованием памяти. Вы заметите небольшой пик на графике загрузки ЦП и уменьшение объема доступной памяти на 50—70 МБ. Такой объем памяти потребляет экземпляр DB2. Если снова выполнить команду `db2stop force`, восстановится прежнее значение объема памяти.



- 6. Повторите предыдущий шаг, но в этот раз следите за тем, что происходит после подключения к базе данных SAMPLE. Выполните следующие команды в командном окне DB2:

```
db2start
db2 connect to sample
```

Сразу после подключения к базе данных SAMPLE вы заметите сокращение объема доступной физической памяти. Это объясняется тем, что сразу после подключения к базе данных происходит распределение глобальной памяти базы данных (буферных пулов, кэш-памяти каталогов и пр.).

7. Повторите предыдущий шаг, но в этот раз следите за тем, что произойдет после создания буферного пула любого размера. Следите за тем, чтобы не превысить доступный уровень физической памяти компьютера. Если это все же случится, DB2 не будет сразу распределять буферный пул, а отложит его создание до отключения базы данных. Кроме того, взамен DB2 будет использовать небольшой системный буферный пул, пока не освободится требуемый объем памяти. К примеру, чтобы создать буферный пул размером приблизительно 160 МБ, выполните следующую команду после подключения к базе данных SAMPLE:

```
db2 create bufferpool mybp immediate size 5000 pagesize 32k
```

8. Создайте базу данных *mydb1*, которая использует автоматическое хранилище; путь базы данных — диск C:, пути хранилища — C:, C:\mystorage1, C:\mystorage2. Выполните следующую команду в командном окне DB2:

```
db2 create database mydb1 on C:\, C:\mystorage1, C:\mystorage2
```

Выполнение команды выше, вероятно, привело к ошибке, поскольку сначала нужно было создать каталоги C:\mystorage1, C:\mystorage2. Создайте их и повторите попытку.

7

Глава 7. Подключаемость клиента DB2

В этой главе рассмотрены настройки, необходимые для подключения клиента DB2 к серверу DB2 по TCP/IP. Обратите внимание, что сервер DB2 поставляется с компонентом клиента и может служить клиентом для подключения к другому серверу DB2. Подключаемость клиента DB2 можно настроить несколькими способами, однако в этой главе рассматривается только наиболее простой из них, т. е. настройка с помощью Ассистента конфигурирования.

Примечание.

Чтобы получить более подробную информацию о подключаемости клиента DB2, просмотрите видео: <http://www.channeldb2.com/video/video/show?id=807741&Video:4222>

Начиная с DB2 9.7, Ассистент конфигурирования считается устаревшим, однако он все так же входит в состав продукта и может использоваться.

7.1 Каталоги DB2

Каталоги DB2 — это двоичные файлы, в которых хранится информация о том, к каким базам данных можно подключиться с компьютера. Существует четыре каталога:

- Каталог системной базы данных
- Каталог локальной базы данных
- Каталог узла
- Каталог DCS

Проверку и обновление содержимого всех этих каталогов можно выполнить с помощью инструмента GUI «Ассистент конфигурирования».

7.1.1 Каталог системной базы данных

Этот каталог похож на содержание книги. В нем показаны все базы данных (как локальные, так и удаленные), к которым можно подключиться. Для локальных баз данных отображается указатель на *Каталог локальной базы данных*. Для удаленных баз данных отображается указатель на запись в *Каталоге узла*. Чтобы просмотреть содержимое этого каталога, выполните команду:

```
list db directory
```

7.1.2 Каталог локальной базы данных

Этот каталог содержит информацию о базах данных, хранящихся на локальном компьютере, к которым можно подключиться. Чтобы просмотреть содержимое этого каталога, выполните команду:

```
list db directory on <drive/path>
```

7.1.3 Каталог узла

В этом каталоге содержится информация о способах подключения к соответствующей удаленной базе данных. К примеру, если используется протокол TCP/IP, в записи узла TCP/IP будет указан IP-адрес сервера, на котором хранится база данных DB2, к которой устанавливается подключение, а также порт экземпляра, который содержит эту базу данных. Чтобы просмотреть содержимое этого каталога, выполните команду:

```
list node directory
```

7.1.4 Каталог DCS

Этот каталог отображается, только если установлено программное обеспечение DB2 Connect для подключения к DB2 для z/OS (майнфрейм) или DB2 для i5/OS. Чтобы просмотреть содержимое этого каталога, выполните команду:

```
list dcs directory
```

7.2 Ассистент конфигурирования (устарело)

С помощью инструмента GUI «Ассистент конфигурирования» можно легко настроить подключаемость между клиентом и сервером DB2. Чтобы запустить инструмент «Ассистент конфигурирования» в Windows, выберите: *Пуск -> Программы -> IBM DB2 -> DB2COPY1 -> Инструменты настройки -> Ассистент конфигурирования*.

Чтобы запустить этот инструмент из командной строки, выполните команду `db2ca`. На рис. 7.1 показан Ассистент конфигурирования.

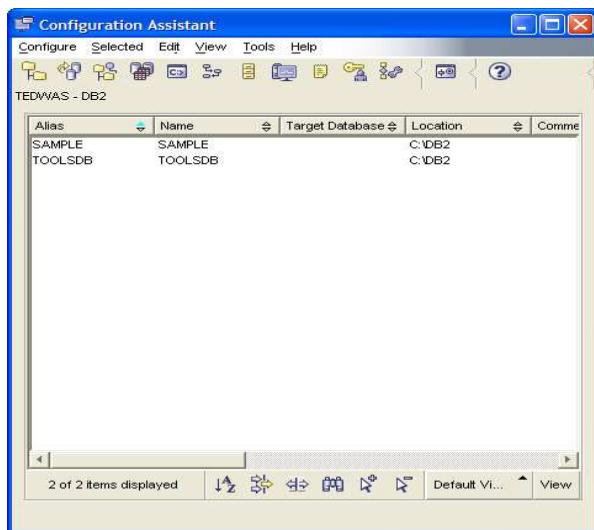


Рисунок 7.1. Ассистент конфигурирования

7.2.1 Необходимая настройка со стороны сервера

Со стороны сервера необходимо настроить два параметра:

1) DB2COMM

Эта переменная реестра DB2 определяет, для какого протокола обмена данными средства отслеживания будут выполнять мониторинг запросов от клиентов. Обычно, наиболее часто используется протокол обмена данными TCP/IP. Чтобы изменить значение этого параметра, потребуется перезагрузка экземпляра. Чтобы просмотреть и изменить значение DB2COMM в Ассистенте конфигурирования, выберите *Configure -> DB2 Registry* (Настроить -> Реестр DB2), как показано на *рис. 7.2 и рис. 7.3*.



Рисунок 7.2. Получение доступа к реестру DB2

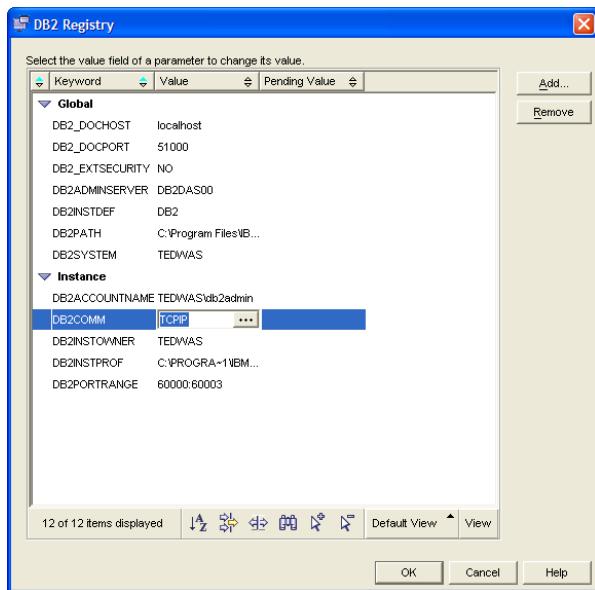


Рисунок 7.3. Проверка значения переменной реестра DB2 DB2COMM

2) SVCENAME

Для этого параметра конфигурации менеджера базы данных необходимо задать имя службы (определенное в файле служб TCP/IP) или номер порта для использования при доступе к базам данных этого экземпляра. В Ассистенте конфигурирования выберите Конфигурирование -> Конфигурация DBM, как показано на рис. 7.4

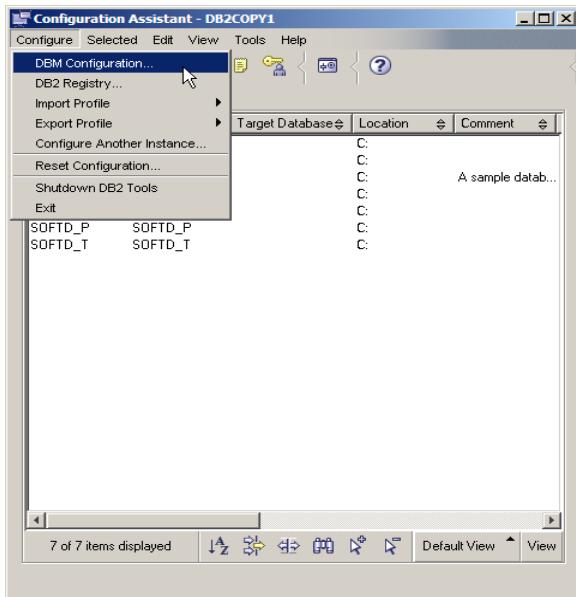


Рисунок 7.4. Проверка dbm cfg с помощью Ассистента конфигурирования

В окне конфигурации менеджера баз данных найдите раздел «Связь» и пункт SVCENAME. При необходимости можно задать строковую переменную или даже указать номер порта. Это показано на рис. 7.5.

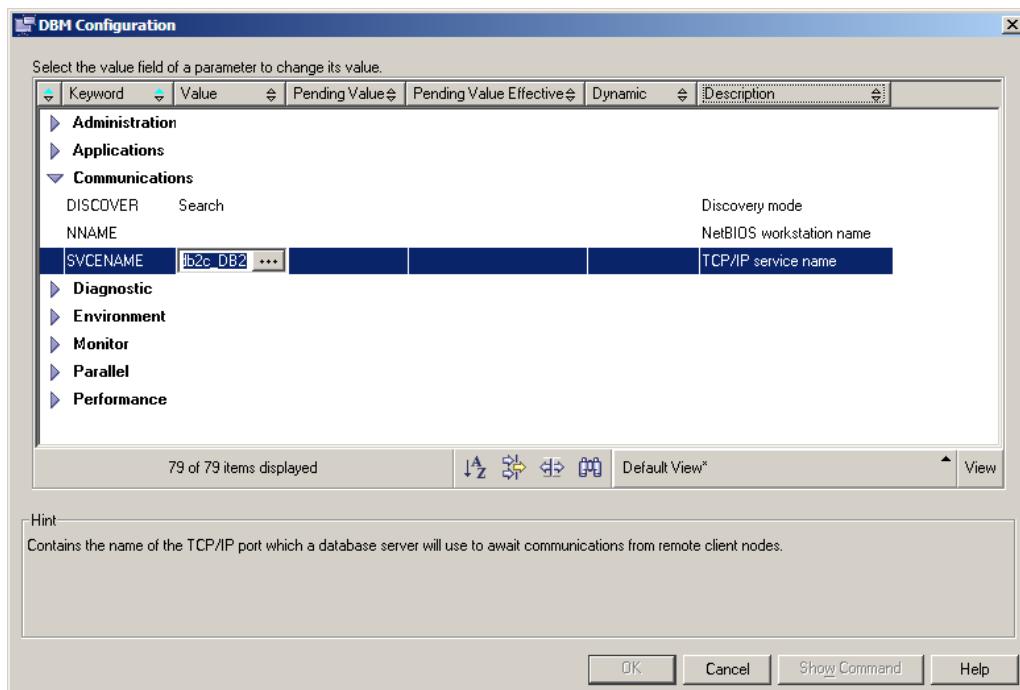


Рисунок 7.5. Проверка параметра SVCENAME для dbm cfg

7.2.2 Необходимая настройка со стороны клиента

Необходимо предварительно собрать следующую информацию со стороны клиента:

- Имя базы данных, с которой необходимо установить связь.
- Номер порта экземпляра DB2 со стороны сервера, на котором хранится база данных. Также можно использовать имя службы, если в файле служб TCP/IP существует совпадающая запись.
- Идентификатор пользователя операционной системы и пароль для подключения к базе данных. Этот идентификатор пользователя должен был определяться на сервере раньше.

Вышеперечисленную информацию можно ввести с клиента DB2, воспользовавшись Ассистентом конфигурирования. Для начала запустите *Мастер по добавлению баз данных*, нажав *Выбранное -> Добавить базу данных при помощи мастера*, как показано на рис. 7.6.

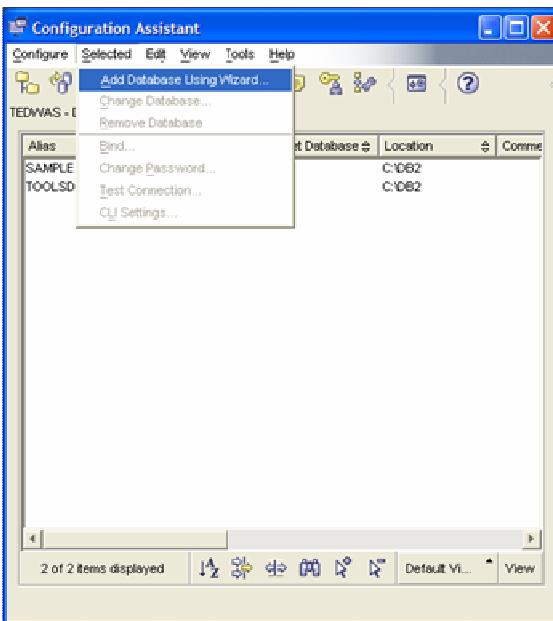


Рисунок 7.6. Запуск мастера по добавлению баз данных

Этот мастер также можно открыть, щелкнув правой кнопкой мыши на пустом пространстве Ассистента конфигурирования и выбрав пункт *Добавить базу данных при помощи мастера*. На рис. 7.7 показан мастер по добавлению баз данных.

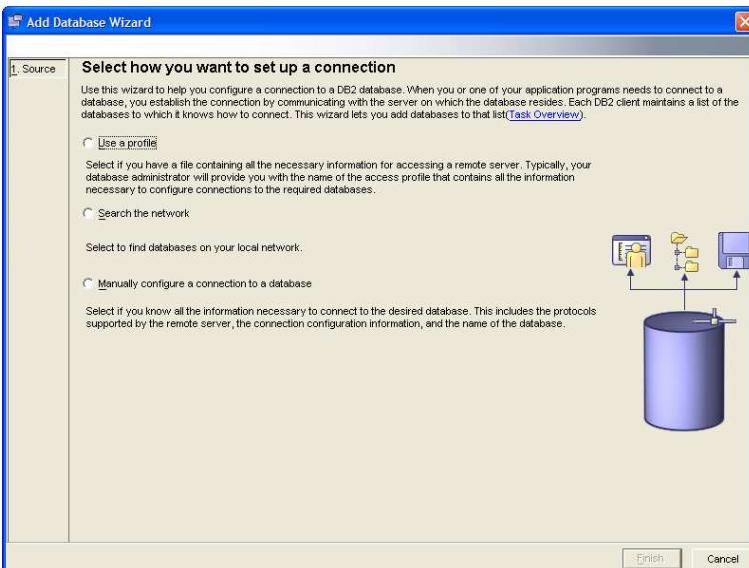


Рисунок 7.7. Мастер по добавлению баз данных

Мастер по добавлению баз данных предлагает три варианта.

1. Использовать профиль

Иногда необходимо конфигурировать несколько клиентов для соединения с одним сервером DB2. В таком случае удобно выполнить все настройки для одного клиента и сохранить их в файле «профиля». С помощью такого файла можно загрузить всю информацию непосредственно в остальные клиенты. Как показано на *рис. 7.7*, если выбрать вариант *Использовать профиль*, будет загружена информация из существующего «профиля». Процесс создания профилей клиента и сервера более подробно описан далее в этой главе.

2. Искать в сети

Этот способ, также известный как *Обнаружение*, дает DB2 команду искать заданные сервер, экземпляр и базу данных в сети. Для правильной работы этого способа на каждом сервере DB2, где могут быть обнаружены базы данных, должен работать DAS. Данный метод предусматривает два способа выполнения поиска:

- **SEARCH:** Другие системы (искать в сети):

Поиск во всей сети. Не рекомендуется для крупных сетей с множеством концентраторов, поскольку извлечение данных с каждой системы займет много времени.

- **KNOWN:** Известные системы:

Поиск в сети известного сервера по предоставленному пользователем адресу.

Оба способа проиллюстрированы на *рис. 7.8*.

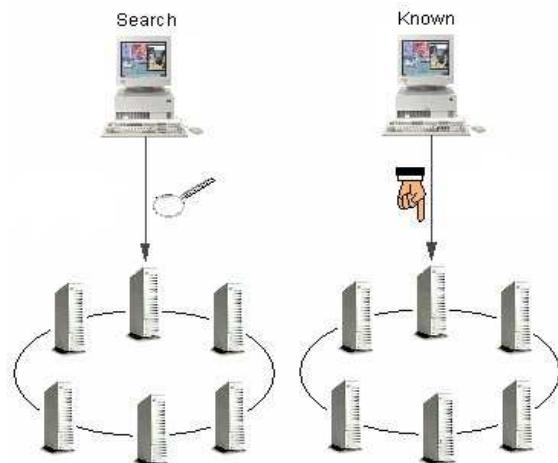


Рисунок 7.8. Способы поиска в сети и поиска (или обнаружения) известных систем

Бывают случаи, когда администратор не хочет разрешать клиентам поиск баз данных с конфиденциальной информацией в сети. Это можно предотвратить на уровне DAS, экземпляра или базы данных. Подробности проиллюстрированы на *рис. 7.9*.

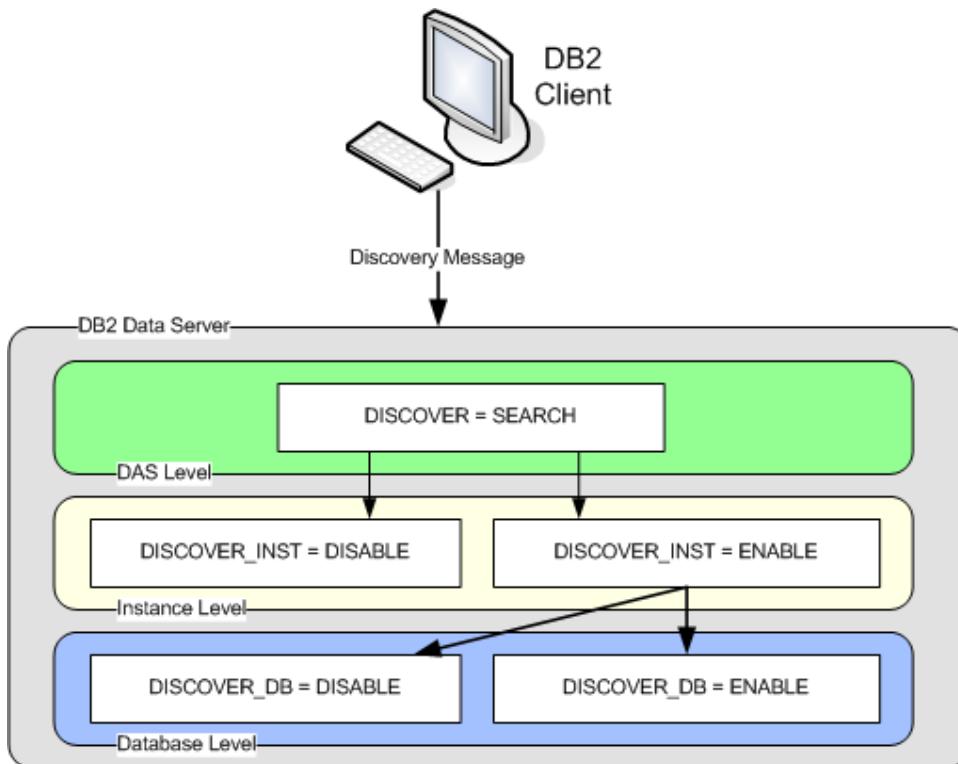


Рисунок 7.9. Конфигурирование параметров для разрешения обнаружения

На рис. 7.9 показаны разные уровни, на которых можно включить или отключить обнаружение. На уровне DAS можно задать для параметра DISCOVER значение SEARCH или KNOWN. На уровне экземпляра для параметра конфигурации менеджера базы данных DISCOVER_INST можно задать значение DISABLE или ENABLE. Наконец, на уровне базы данных для параметра DISCOVER_DB также можно задать значение ENABLE или DISABLE. Правильная установка этих параметров обеспечивает нужную степень структурированности обнаружения баз данных.

3. Конфигурировать соединение с базой данных вручную

Используя этот способ, пользователь вручную указывает имя хоста, номера портов и информацию о базе данных в Ассистенте конфигурирования, который затем генерирует команды каталога для конфигурирования подключаемости. Ассистент конфигурирования не проверяет правильность указанной информации. О неправильности такой информации будет свидетельствовать отсутствие соединения с сервером. Также убедитесь в правильности идентификатора пользователя и пароля для соединения с удаленной базой данных. По умолчанию аутентификация выполняется на сервере DB2, с которым устанавливается связь. Соответственно, необходимо указать идентификатор пользователя и пароль для этого сервера.

7.2.3 Создание профилей клиента и сервера

Если необходимо конфигурировать большое число серверов или клиентов, вместо того чтобы выполнять настройку каждого из них отдельно, можно сконфигурировать один, а затем экспорттировать профиль, который в дальнейшем будет применяться к остальным клиентам/серверам. Это позволит сэкономить немало времени, затрачиваемого на настройку среды.

Чтобы создать пользовательский профиль в Ассистенте конфигурирования, откройте меню *Конфигурирование* и выберите *Экспортировать профиль -> Настроить*, как показано на рис. 7.10.

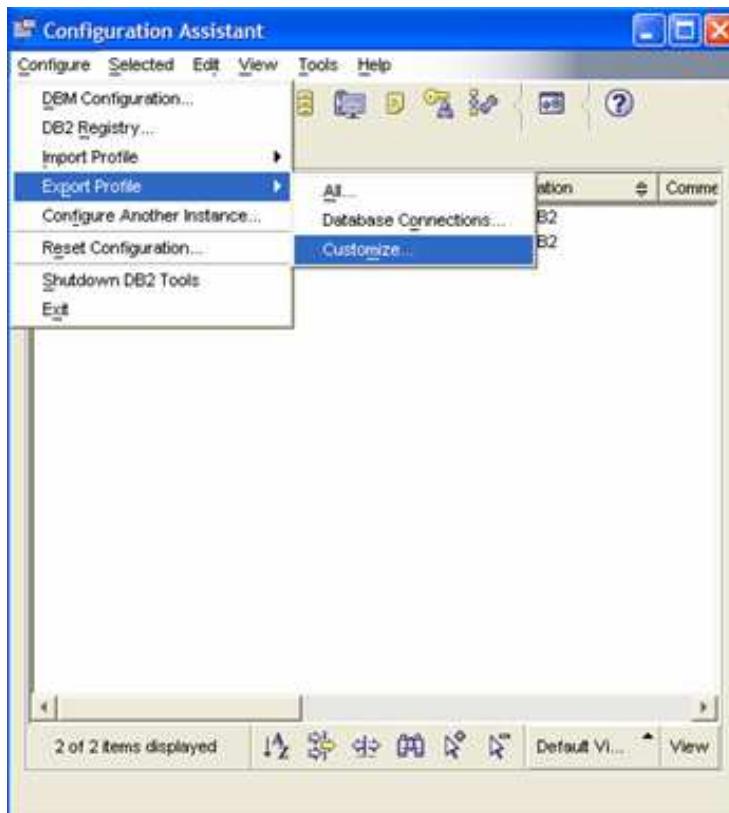


Рисунок 7.10. Экспортирование профиля

На рис. 7.11 показаны поля, которые необходимо заполнить для экспорттирования профиля.

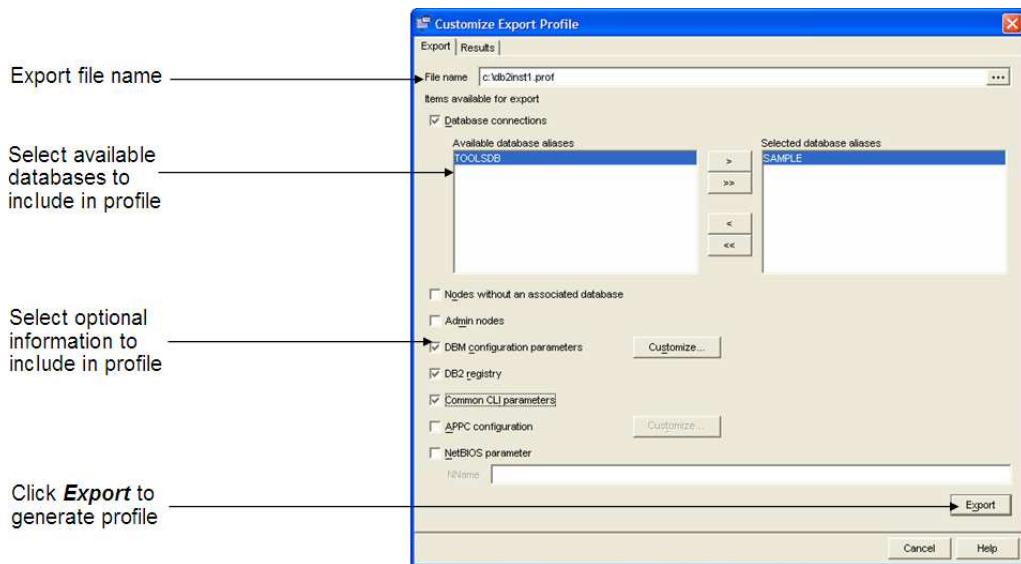


Рисунок 7.11. Диалоговое окно «Настройте профиль экспорта»

На рис. 7.12 проиллюстрирован результат нажатия кнопки «Экспорт» в диалоговом окне «Настройте профиль экспорта».

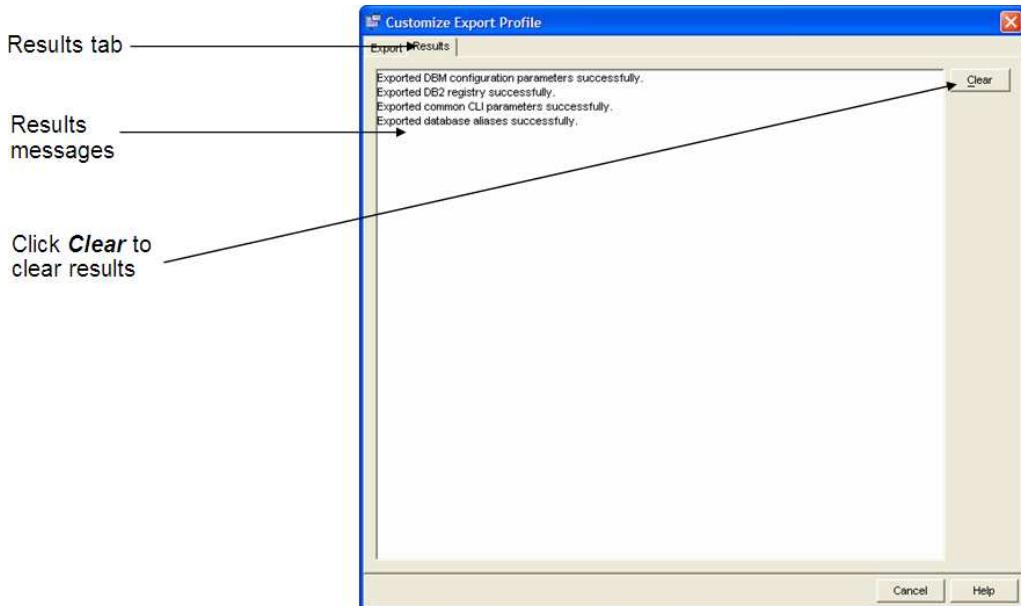


Рисунок 7.12. Результаты экспортации профиля

Чтобы импортировать пользовательский профиль с помощью Ассистента конфигурирования, откройте меню Конфигурирование и выберите *Импортировать профиль* -> *Настройте*, как показано на рис. 7.13.

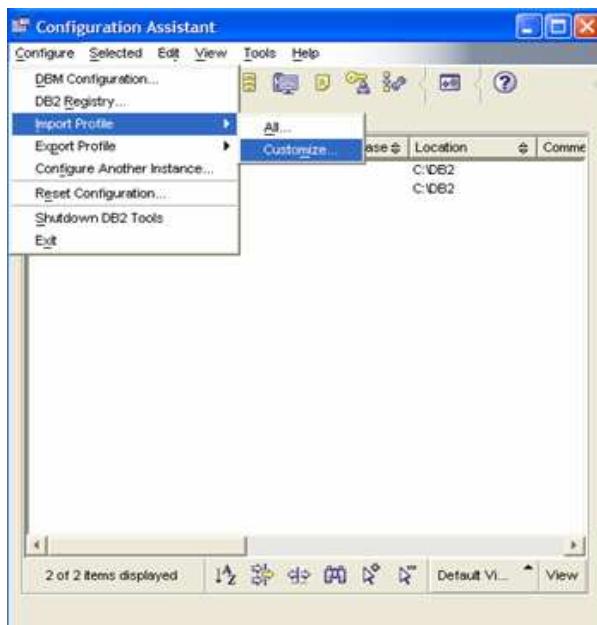


Рисунок 7.13. Импортирование профиля

На рис. 7.14 показаны поля, которые необходимо заполнить для импортирования профиля.

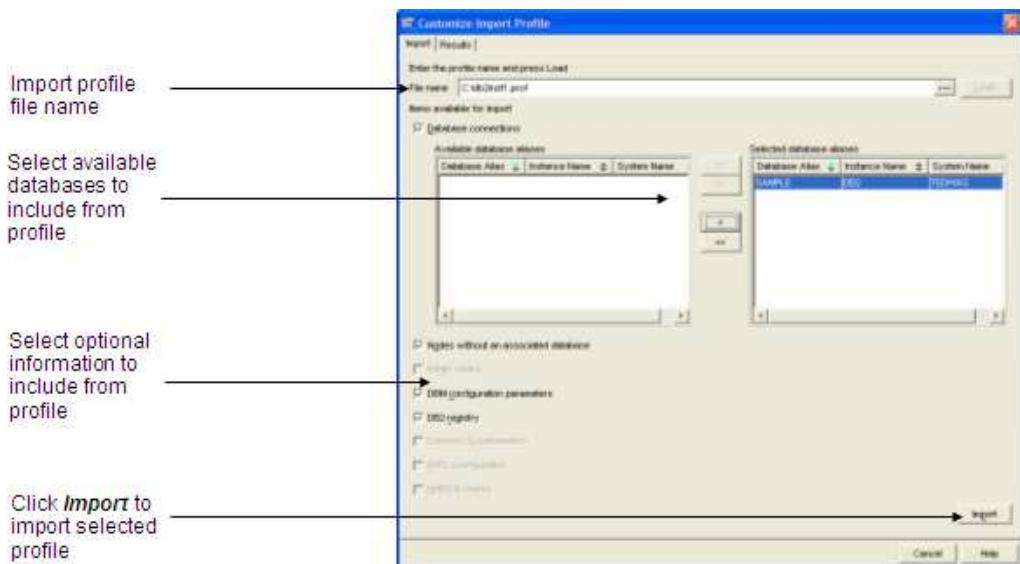


Рисунок 7.14. Настройте профиль импорта

7.3 Краткий обзор

Подключение клиента базы данных к серверу является ключевым аспектом управления реляционными базами данных. В этой главе мы рассмотрели возможности подключения клиента, начиная от целей и содержимого каталогов баз данных и узлов, привязанных к DB2.

Затем мы обсудили использование графического интерфейса пользователя «Ассистент конфигурирования» для установки соединения между клиентом и сервером, в том числе требуемые настройки двух сторон подключения.

Мы также рассмотрели использование мастера по добавлению баз данных для подключения к серверу с помощью одного из трех способов: с использованием сохраненного профиля, посредством поиска в сети (или обнаружения) или вручную, с указанием данных сервера. Затем мы более подробно изучили процесс создания профилей клиента и сервера.

7.4 Упражнения

С помощью Ассистента конфигурирования можно быстро и просто настроить подключения к удаленной базе данных. В этом упражнении мы создадим каталог базы данных, расположенный на удаленном сервере DB2 (удаленным сервером будем считать соседнюю рабочую станцию), воспользовавшись режимами поиска в сети и обнаружения. Каталогизировав базу данных, мы сможем получить к ней доступ точно так же, как и к базам данных в локальной системе. DB2 выполняет процесс подключения «внутри системы».

Для этого упражнения предположим, что вы работаете в сети. Если это не так, можно использовать один компьютер в качестве и клиента, и сервера, и выполнить приведенные ниже инструкции по конфигурированию, чтобы подключиться к собственной системе.

Процедура

1. Узнайте у напарника (или инструктора) следующую информацию:
2. Информация об удаленной базе данных:

| | | |
|------|------------------------|---------------|
| (PR) | Протокол | <u>TCPIP</u> |
| (IP) | IP-адрес или имя хоста | <u> </u> |
| (PN) | Номер порта экземпляра | <u> </u> |
| (DB) | Имя базы данных | <u>SAMPLE</u> |

Подсказки:

Чтобы узнать имя хоста в Windows, введите `hostname` в командном окне.

Чтобы узнать IP-адрес в Windows, введите `ipconfig` в командном окне.

3. Откройте Ассистент конфигурирования. (Подсказка: доступ к нему можно получить через меню Пуск.)
4. Откройте меню *Выбранное* и выберите *Добавить базу данных при помощи мастера*.
5. На странице мастера *Источник* выберите вариант *Конфигурировать соединение с базой данных вручную*. Нажмите кнопку *Далее*, чтобы перейти к следующей странице мастера.
6. На странице мастера *Протокол* выберите вариант TCP/IP. Нажмите кнопку *Далее*, чтобы перейти к следующей странице мастера.
7. На странице мастера TCP/IP укажите полное имя хоста или IP-адрес, полученный при выполнении шага (1). Укажите номер порта, полученный при выполнении шага (1). Нажмите кнопку *Далее*, чтобы перейти к следующей странице мастера.
8. Примечание. Вариантом *Имя службы* можно воспользоваться при наличии записи в файле служб *local* с указанием номера порта, определенного в соответствии с портом, который отслеживается экземпляром удаленного сервера. При использовании этого варианта DB2 анализирует файл служб на локальном компьютере, а не на сервере. Чтобы воспользоваться этим вариантом, необходимо добавить в файл соответствующую запись.
9. На странице мастера *База данных* введите в поле *Имя базы данных* определенное на удаленном сервере имя базы данных, полученное при выполнении шага (1). Обратите внимание на то, что в поле *Алиас базы данных* автоматически появляется такое же значение. Алиас базы данных — это имя, которое будет использоваться локальными приложениями для подключения к этой базе данных. Поскольку уже определена локальная база данных с именем *SAMPLE*, DB2 не разрешит каталогизировать ещё одну базу данных с таким же именем. Поэтому необходимо воспользоваться другим именем алиаса. Для этого примера измените алиас базы данных на *SAMPLE1*. При желании можно внести комментарии об этой базе данных. Нажмите кнопку *Далее*, чтобы перейти к следующей странице мастера.
10. На странице мастера *Источник данных* при необходимости можно зарегистрировать новую базу данных (источник данных) в качестве источника данных ODBC. В результате будет автоматически выполнена регистрация в диспетчере ODBC Windows. Для этого примера снимите флажок возле пункта *Зарегистрировать эту базу данных для ODBC*, поскольку ODBC использовать не будет. Нажмите кнопку *Далее*, чтобы перейти к следующей странице мастера.
11. На странице мастера *Опции узла* укажите операционную систему сервера, на котором расположена удаленная база данных. Поскольку в этом упражнении на всех компьютерах используется Microsoft Windows, убедитесь, что из раскрывающегося меню выбран элемент *Windows*. Поле «*Имя экземпляра*»

должно иметь значение *DB2*. Если указано другое значение, задайте значение *DB2*. Нажмите кнопку *Далее*, чтобы перейти к следующей странице мастера.

12. На странице мастера *Опции системы* можно убедиться в правильности системы и имени хоста, а также проверить параметры операционной системы. Нажмите кнопку *Далее*, чтобы перейти к следующей странице мастера.
13. На странице мастера *Опции безопасности* можно указать, где и каким методом следует выполнять аутентификацию пользователя. Выберите вариант *Использовать значение аутентификации из конфигурации DBM сервера*. В результате будет использоваться метод, определенный параметром AUTHENTICATION в файле конфигурации удаленного экземпляра. Нажмите кнопку *Готово*, чтобы каталогизировать удаленную базу данных и закрыть мастер. Должно открыться окно подтверждения. Нажмите кнопку *Проверить соединение*, чтобы убедиться в возможности успешного соединения с базой данных. Также убедитесь в том, что предоставлены действительные имя пользователя и пароль, *определенные на удаленном сервере* (поскольку для параметра AUTHENTICATION сервера вероятно задано значение SERVER). Успешное выполнение проверки соединения указывает на успешность каталогизации удаленной базы данных. Если проверку выполнить не удалось, перейдите назад по страницам мастера и убедитесь в правильности всех указанных значений. (Нажмите кнопку *Изменить*, чтобы вернуться к настройкам мастера).
14. Откройте Центр управления и попробуйте просмотреть разные таблицы в только что каталогизированной удаленной базе данных.
15. Вернитесь к Ассистенту конфигурирования и попробуйте каталогизировать другую базу данных, на этот раз выбрав вариант *Искать в сети*. Выполните предлагаемые мастером шаги так же, как при конфигурировании соединения вручную. Обратите внимание, что выполнение поиска и обнаружения в крупных сетях может занять некоторое время.

8

Глава 8. Работа с объектами базы данных

В этой главе рассмотрены такие объекты базы данных, как схемы, таблицы, представления, индексы, последовательности и пр. Некоторые сложные объекты приложений базы данных, в частности триггеры, пользовательские функции (UDF) и хранимые процедуры, рассмотрены в Главе 14 «*Введение в разработку приложений DB2*».

Примечание.

Чтобы получить более подробную информацию о работе с объектами базы данных, просмотрите видео: <http://www.channeldb2.com/video/video/show?id=807741:Video:4242>

8.1 Схемы

Схемы — это пространства имён для сбора объектов базы данных. Они преимущественно используются для:

- обеспечения индикации монопольного использования объектов или связей с приложением;
- логической группировки связанных объектов.

Все объекты базы данных DB2 (за исключением общих синонимов) имеют полностью классифицированные имена, состоящие из двух частей; схема является первой частью такого имени, как показано ниже:

<имя_схемы>. <имя_объекта>

Полностью классифицированное имя объекта должно быть уникальным. Если подключиться к базе данных и создать или обратиться к объекту, не указав схему, DB2 будет использовать идентификатор пользователя, с помощью которого установлено подключение к базе данных, в качестве имени схемы. К примеру, если подключиться к базе данных `SAMPLE` как пользователь `arfchong` и создать таблицу `artists` с помощью следующего оператора `CREATE TABLE`:

```
CREATE TABLE artists ...
```

созданная таблица будет иметь полностью классифицированное имя `arfchong.artists`.

Чтобы задать схему для сеанса, можно воспользоваться оператором `set schema`. В листинге 8.1 представлен пример.

```
connect to sample user arfchong using mypsw
select * from staff ## This looks for arfchong.staff
set schema db2admin
select * from staff ## This looks for db2admin.staff
```

Листинг 8.1. Пример использования оператора `set schema`

Чтобы проиллюстрировать использование схем, можно воспользоваться «конкурсной системой». Допустим, компания проводит конкурс, участники которого должны создать собственные таблицы и выполнить некоторые SQL-операции. Все участники получают одинаковый идентификатор пользователя для подключения к базе данных и одинаковый сценарий для создания таблиц, все объекты которых имеют неполные имена, т. е. не имеют имени схемы. Когда участники входят в систему конкурса, она генерирует имена схем на основании меток времени после подключения. Таким образом, участник А и участник В будут работать в таблицах с одинаковыми именами, но разными схемами, что позволит избежать конфликтов в их работе.

new in
V9.7

8.2 Общие синонимы (или алиасы)

В DB2 9.7 впервые представлена концепция общих синонимов, также известных как общие алиасы. Общие синонимы дают возможность ссылаться на объекты, не указывая схемы. В листинге 8.2 представлен пример.

```
connect to sample user arfchong using mypsw
create public synonym raul for table arfchong.staff
select * from raul
select * from arfchong.raul ## Error
connect to sample user db2admin using psw
select * from raul
```

Листинг 8.2. Пример общего синонима

В Листинге 8.2 сначала устанавливается подключение пользователя `arfchong` и создается общий синоним `raul`, который ссылается на таблицу `arfchong.staff`. Сам синоним не использует схему. При попытке использовать схему возникнет ошибка. Другие пользователи, такие как `db2admin`, в примере в листинге 8.2, также могут использовать синоним `raul`, который является общим.

Если в нашем примере не используется ключевое слово `public` (общий), создаваемый синоним будет частным. В листинге 8.3 рассмотрен тот же пример, но с использованием частного синонима.

```
connect to sample user arfchong using mypsw
create synonym raul for table arfchong.staff
select * from raul
select * from arfchong.raul ## OK, it also works
connect to sample user db2admin using psw
select * from raul ## Error, cannot find db2admin.raul
select * from arfchong.raul ## OK, this works
```

Листинг 8.3. Пример частного синонима

Из примера в листинге 8.3 видно, что поскольку синоним является частным, его нельзя использовать при входе под другим именем пользователя, не указав схему.

8.3 Таблицы

Таблица — это собрание связанных данных, логически упорядоченных в столбцах и строках. В листинге 8.4 ниже представлен пример создания таблицы с помощью оператора `CREATE TABLE`.

```

CREATE TABLE artists
  (artno      SMALLINT not null,
   name       VARCHAR(50) with default 'abc',
   classification CHAR(1) not null,
   bio        CLOB(100K) logged,
   picture    BLOB(2M) not logged compact
  )
IN mytbls1
  
```

Листинг 8.4. Пример оператора CREATE TABLE

В последующих разделах рассмотрены основные составляющие оператора `CREATE TABLE`.

8.3.1 Типы данных

На рис. 8.1, полученном в информационном центре DB2, перечислены типы данных, поддерживаемые в DB2.

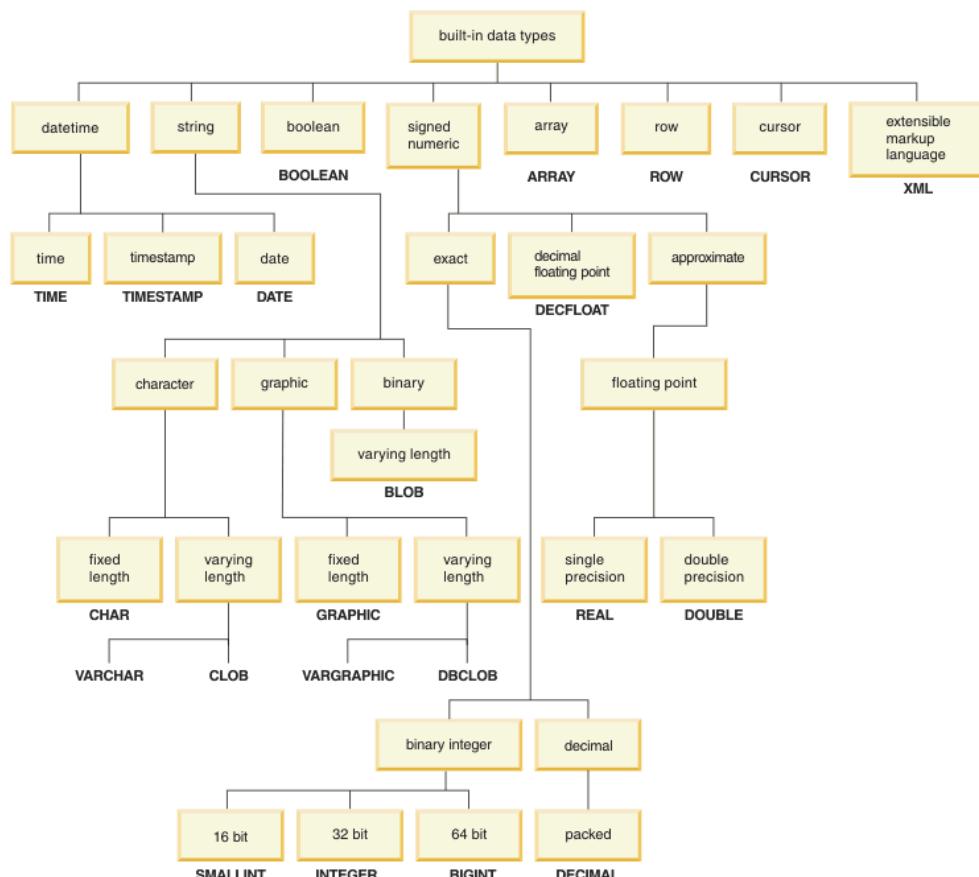


Рисунок 8.1. Встроенные в DB2 типы данных

Показанные на *рис. 8.1* типы данных подробно описаны в документации DB2; большинство этих типов широко используются в других реляционных СУБД, поэтому мы не будем рассматривать их в этой книге. С другой стороны, некоторые типы данных, например большие объекты (LOB), могут быть не столь интуитивно понятны для новых пользователей.

Типы данных для работы с большими объектами используются для хранения длинных символьных строк, длинных двоичных строк или файлов, как показано на *рис. 8.2*.

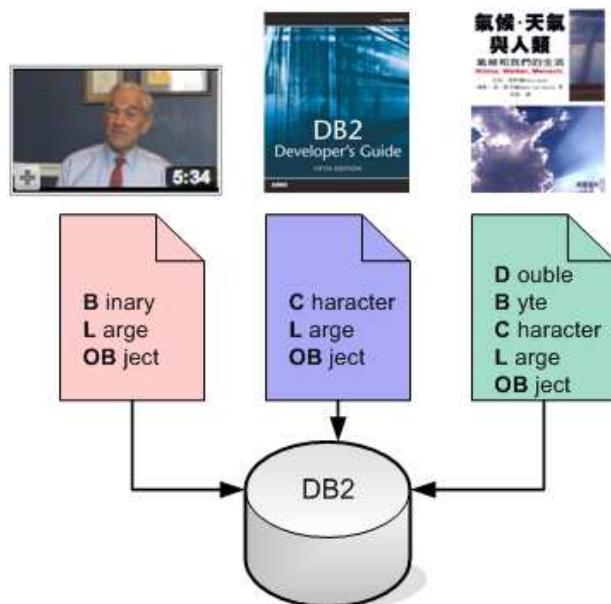


Рисунок 8.2. Типы данных LOB

Названия двоичных элементов таких больших объектов обычно сокращают для большей четкости: большой двоичный объект — BLOB (binary large object), большой символьный объект — CLOB (character large object), большой двухбайтовый символьный объект — DBCLOB (double byte character large object).

На *рис. 8.1* также перечислены новые типы данных, представленные в DB2 9.7:

new in V9.7

- BOOLEAN (логический тип данных);
- ARRAY (массив данных);
- ROW (строковый тип данных);
- CURSOR (курсорный тип данных).

Эти типы данных используются серверами данных Oracle и теперь поддерживаются в DB2. Типы данных сервера данных Oracle рассмотрены более подробно в последующих разделах этой главы.

8.3.1.1 Пользовательские типы данных

DB2 дает пользователям возможность самостоятельно определить типы данных, используя пользовательские типы (UDT). UDT имеют следующую классификацию:

- особый тип (Distinct);
- структурированный тип (Structured);
- ссылочный тип (Reference);
- тип массива (Array);
- строковый тип (Row);
- курсорный тип (Cursor).

**new in
V9.7**

Ссылочный, строковый, курсорный типы данных и тип массива впервые представлены в DB2 9.7 и используются в стандартных операциях SQL PL. Особые пользовательские типы данных основаны на встроенных типах данных. Такие UDT полезны в следующих случаях:

- необходимо определить контекст значений;
- необходимо в обязательном порядке применять в DB2 строгий контроль типов данных.

SQL-операторы в *Листинге 8.5* показывают, когда и каким образом применяются UDT:

```
CREATE DISTINCT TYPE POUND AS INTEGER WITH COMPARISONS
CREATE DISTINCT TYPE KILOGRAM AS INTEGER WITH COMPARISONS
CREATE TABLE person
  (f_name  VARCHAR(30),
   weight_p POUND NOT NULL,
   weight_k KILOGRAM NOT NULL )
```

Листинг 8.5. Пример особых типов данных

В этом примере создаются два особых UDT: POUND (фунт) и KILOGRAM (килограмм). Оба типа основаны на встроенном типе данных INTEGER (целые числа). Выражения WITH COMPARISONS, определенные как часть синтаксиса, обозначают, что также будут созданы функции приведения типов с именем, соответствующим типам данных.

Таблица person использует два новых UDT в столбцах weight_p и weight_k, соответственно. Теперь, если задать следующий оператор:

```
SELECT F_NAME FROM PERSON
  WHERE weight_p > weight_k
```

появится сообщение об ошибке, поскольку сравниваются два столбца с разными типами данных. Хотя weight_p и weight_k используют, соответственно, типы данных POUND и KILOGRAM, созданные на основе типа данных INTEGER, в результате создания UDT такой тип сравнения становится невозможным. Требовалось достичь

именно такого результата, поскольку в реальной жизни сравнение значений в фунтах и килограммах не имело бы смысла.

В следующем примере попробуем сравнить значения столбца `weight_p` с целочисленными значениями; однако эти два типа данных являются разными, поэтому появится сообщение об ошибке, если не использовать функцию приведения типов.

Как видно в операторе ниже, мы используем функцию приведения типов `POUND()`, чтобы сделать такое сравнение возможным. Как уже упоминалось, функция приведения типов `POUND()` была создана с UDT при использовании выражения `WITH COMPARISONS` в операторе `CREATE DISTINCT TYPE`.

```
SELECT F_NAME FROM PERSON
WHERE weight_p > POUND(30)
```

8.3.1.2 Типы данных сервера данных Oracle

Следующие типы данных, используемые в сервере данных Oracle, теперь поддерживаются сервером данных DB2: `NUMBER`, `VARCHAR2`, `TIMESTAMP(n)`, «`DATE`», `BOOLEAN`, `INDEX BY`, `VARRAY`, `REF CURSOR` и строчный тип. Чтобы они работали, необходимо включить переменную реестра `DB2_COMPATIBILITY_VECTOR`, как описано ниже:

```
db2set DB2_COMPATIBILITY_VECTOR=FF
db2stop
db2start
```

После включения этой переменной реестра новые базы данных смогут поддерживать такие типы данных. Некоторые из этих типов могут применяться только в контексте SQL PL.

Примечание.

Если используется редакция DB2, поддерживающая функцию совместимости SQL (см. Главу 2), также можно использовать эти типы данных при применении PL/SQL. В таком случае для переменной реестра `DB2_COMPATIBILITY_VECTOR` нужно задать значение «`FFF`», а не «`FF`», как в примере выше.

8.3.1.3 Неявное приведение типов или слабый контроль типов

Многие динамические языки программирования, такие как Ruby on Rails или PHP, допускают неявное приведение типов. Это было проблемой для DB2 в связи с использованием строгого контроля типов. В DB2 9.7 правила упростились, и теперь допускается неявное приведение типов или слабый контроль типов. Иными словами, к примеру, теперь можно назначить или сравнить строчный тип данных с числовыми типами, как показано ниже:

```
create table t1 (coll int)
```

```
select * from t1 where col1 = '42'
```

В этом примере строку «42» теперь можно сравнить с целочисленным столбцом *col1*.

Кроме того, в DB2 9.7 теперь можно указывать нетипизированные маркеры параметров и нетипизированные пустые значения (NULL) в большем количестве случаев. Раньше необходимо было явно отнести их к определенному типу данных. К примеру, теперь может работать следующий оператор:

```
select ?, NULL, myUDF(?, NULL) from t1
```

8.3.1.4 Пустые значения

Пустое значение указывает на неизвестное состояние. Оператор CREATE TABLE может определить столбец с помощью выражения NOT NULL, чтобы обеспечить содержание в столбце данных известного типа. Также можно указать значение по умолчанию для столбцов, объявленных как NOT NULL. Следующий оператор иллюстрирует вышеописанное:

```
CREATE TABLE staff (
    ID      SMALLINT NOT NULL,
    NAME    VARCHAR(9),
    DEPT   SMALLINT NOT NULL with default 10,
    JOB    CHAR(5),
    YEARS   SMALLINT,
    SALARY  DECIMAL(7,2),
    COMM    DECIMAL(7,2) with default 15
)
```

В этом примере столбцы *ID* и *DEPT* определены как NOT NULL. Столбец *DEPT* также по умолчанию включает значение 10, если не указано другое значение.

8.3.2 Столбцы идентификации

Столбец идентификации — это числовой столбец, автоматически генерирующий уникальное числовое значение для каждой вставляемой строки. В таблице может быть только один столбец идентификации.

Существует два способа генерирования значений столбца идентификации, в зависимости от его определения:

- **Генерируются всегда:** DB2 всегда генерирует значения. Приложения не могут задавать явные значения.
- **Генерируются по умолчанию:** приложения могут задавать явные значения, а если значение не указано, его генерирует DB2. DB2 не может гарантировать уникальность таких значений. Этот способ предназначен для распространения данных, а также для разгрузки и перезагрузки таблицы.

Рассмотрим следующий пример:

```
CREATE TABLE subscriber(
    subscriberID INTEGER GENERATED ALWAYS AS
        IDENTITY (START WITH 100 INCREMENT BY 100),
    firstname VARCHAR(50),
    lastname VARCHAR(50) )
```

В этом примере столбец *subscriberID* имеет тип INTEGER и определен как столбец идентификации, значения для которого генерируются всегда. Генерируемые значения будут начинаться со 100 и повышаться с шагом 100.

8.3.3 Объекты последовательностей

Хотя объекты последовательностей не зависят от таблиц, они рассматриваются в этой главе, поскольку функционируют так же, как столбцы идентификации. Разница между ними заключается в том, что объекты последовательностей генерируют уникальные числа в базе данных, а столбцы идентификации — в таблице. Операторы ниже иллюстрируют пример:

```
CREATE TABLE t1 (salary int)

CREATE SEQUENCE myseq
    START WITH 10
    INCREMENT BY 1
    NO CYCLE

INSERT INTO t1 VALUES (nextval for myseq)

INSERT INTO t1 VALUES (nextval for myseq)

INSERT INTO t1 VALUES (nextval for myseq)

SELECT * FROM t1

SALARY
-----
10
11
12
3 record(s) selected.

SELECT prevval for myseq FROM sysibm.sysdummy1

1
-----
12
1 record(s) selected
```

Листинг 8.6. Пример последовательностей

PREVVAL предоставляет текущее значение последовательности, а NEXTVAL — следующее значение. В примере выше также используется SYSIBM.SYSDUMMY1. Это таблица системного каталога, содержащая один столбец и одну строку. Она применяется в случаях, когда для запроса необходимо вывести одно значение. Таблицы системных каталогов описаны в следующем разделе.

8.3.4 Таблицы системного каталога

Каждая база данных имеет собственные таблицы и представления системного каталога. В них хранятся *методанные* об объектах базы данных. При повреждении таких системных таблиц база данных выйдет из строя. В этих таблицах можно выполнять запросы, как и в обычных таблицах базы данных. Для идентификации таблиц системного каталога используются три схемы:

- SYSIBM: основные таблицы, оптимизированные для использования в DB2
- SYSCAT: представления на основе таблиц SYSIBM, оптимизированные для простоты использования
- SYSTAT: статистика базы данных

Ниже представлены примеры представлений каталога:

- SYSCAT.TABLES
- SYSCAT.INDEXES
- SYSCAT.COLUMNS
- SYSCAT.FUNCTIONS
- SYSCAT.PROCEDURES.

8.3.5 Объявленные глобальные временные таблицы (Declared global temporary tables, DGTT)

Объявленные временные таблицы — это созданные в памяти таблицы, используемые приложением и автоматически отбрасываемые при завершении его работы. Доступ к таким таблицам может получить только то приложение, которое их создало, и они не хранятся ни в одной из таблиц каталога DB2. Доступ к таким таблицам значительно повышает производительность, поскольку не возникают конфликты каталога, а также не используется блокировка строк, ведение журнала (не обязательно) и проверка полномочий. Такие временные таблицы также поддерживают индексы, т. е. во временной таблице можно создать любой стандартный индекс. Для таких таблиц можно также выполнить RUNSTATS.

Объявленные временные таблицы располагаются в пользовательском временном табличном пространстве, которое должно определяться до их создания. Операторы в листинге 8.7 иллюстрируют процесс создания трех объявленных временных таблиц:

```
CREATE USER TEMPORARY TABLESPACE apptemps
  MANAGED BY SYSTEM USING ('apptemps');

DECLARE GLOBAL TEMPORARY TABLE temployees
```

```

    LIKE employee NOT LOGGED;

DECLARE GLOBAL TEMPORARY TABLE tempdept
(deptid CHAR(6), deptname CHAR(20))
ON COMMIT DELETE ROWS NOT LOGGED;

DECLARE GLOBAL TEMPORARY TABLE tempprojects
AS ( fullselect ) DEFINITION ONLY
ON COMMIT PRESERVE ROWS NOT LOGGED
WITH REPLACE IN TABLESPACE apptemps;

```

Листинг 8.7. Работа с DGTT

При создании объявленная временная таблица имеет схему SESSION; эту схему необходимо указывать, ссылаясь на DGTT. Используемый для создания временных таблиц идентификатор пользователя будет иметь в этих таблицах все привилегии. Каждое приложение, создающее временную таблицу, будет иметь собственную копию такой таблицы, как показано на рис. 8.3.



Рисунок 8.3. Область определения объявленных глобальных временных таблиц

В листинге 8.8 показаны ограничения области определения объявленных глобальных временных таблиц. Подразумевается, что пользовательские временные табличные пространства были созданы ранее.

В командном окне DB2 #1:

```

db2 connect to sample
db2 declare global temporary table mydgtt (coll int, col2 varchar(10)) on
commit preserve rows
db2 insert into session.mydgtt values (1,'hello1'),(2,'hello2'),
(3,'hello3')
db2 select * from session.mydgtt

```

```
COL1      COL2
-----
1 hello1
2 hello2
3 hello3
3 record(s) selected.
```

В командном окне DB2 #2:

```
db2 connect to sample
db2 select * from session.mydgtt
SQL0204N "SESSION.MYDGTT" is an undefined name. SQLSTATE=42704
```

Листинг 8.8. Работа с областью определения DGTT

В Листинге 8.8 видно, что при попытке использовать `SESSION.MYDGTT` во втором сеансе (командное окно DB2 #2) возникает ошибка, поскольку в этом сеансе нет определения DGTT. Обратите внимание, что определение DGTT использует выражение `ON COMMIT PRESERVE ROWS`, поскольку командное окно DB2 по умолчанию задействуется при вводе каждого оператора.

new in
V9.7

8.3.6 Созданные глобальные временные таблицы (Create Global Temporary Tables, CGTT)

Хотя DGTT дают возможность создать временную таблицу, определение такой таблицы нельзя совместно использовать в разных подключениях или сессиях. При запуске каждого сеанса необходимо выполнять оператор `DECLARE GLOBAL TEMPORARY TABLE`. С другой стороны, при использовании созданных глобальных временных таблиц (Create Global Temporary Tables, CGTT) определение временной таблицы нужно создать всего один раз, поскольку оно навсегда сохраняется в каталоге DB2. Это означает, что другие подключения могут попросту воспользоваться такой таблицей, а не создавать её снова. Хотя структуру таблицы можно использовать сразу, данные разных подключений не зависят друг от друга и пропадают после закрытия подключения. К примеру, рассмотрим листинг 8.9. Подразумевается, что пользовательское временное табличное пространство было создано ранее.

В командном окне DB2 #1:

```
db2 connect to sample
db2 create global temporary table mycggtt (col1 int, col2 varchar(10)) on
commit preserve rows
db2 insert into mycggtt values (1,'hello1'),(2,'hello2'), (3,'hello3')
db2 select * from mycggtt
```

```
COL1      COL2
-----
1 hello1
```

```
2 hello2
3 hello3
3 record(s) selected.

В командном окне DB2 #2:
db2 connect to sample
db2 select * from mycgtt

COL1      COL2
-----
0 record(s) selected.
```

Листинг 8.9. Работа с областью определения CGTT

Из листинга 8.9 видно, что в командном окне DB2 #2 (другой сеанс или подключение) не нужно снова создавать CGTT — можно просто сослаться на неё; однако на выходе нет ни одной строки, поскольку данные относились к первому сеансу.

8.4 Представления

Представление — это отображение данных в таблицах. Данные для представлений не хранятся отдельно, они отбираются при запуске представления. Поддерживаются вложенные представления, т. е. представления, созданные на основе других представлений. Вся информация о представлениях хранится в следующих представлениях каталога DB2: SYSCAT.VIEWS, SYSCAT.VIEWDEP и SYSCAT.TABLES. В листинге 8.10 приведен пример создания и использования представления.

```
CONNECT TO MYDB1;

CREATE VIEW MYVIEW1
AS SELECT ARTNO, NAME, CLASSIFICATION
FROM ARTISTS;

SELECT * FROM MYVIEW1;
```

Результат:

| ARTNO | NAME | CLASSIFICATION |
|-------|-----------|----------------|
| 10 | HUMAN | A |
| 20 | MY PLANT | C |
| 30 | THE STORE | E |
| ... | | |

Листинг 8.10. Работа с представлениями

8.5 Индексы

Индекс — это упорядоченный набор ключей, каждый из которых указывает на строку таблицы. Индексы обеспечивают уникальность строк и повышают производительность. Ниже описаны некоторые характеристики, которые можно определить для индексов:

- Индексы могут указываться по возрастанию или по убыванию
- Ключи индексов могут быть уникальными или неуникальными
- Индексы могут записываться в нескольких столбцах (такие индексы называют комбинированными)
- Если индексные и физические данные сгруппированы в одинаковой последовательности индексов, они называются сгруппированными индексами.

Например:

```
CREATE UNIQUE INDEX artno_ix ON artists (artno)
```

8.5.1 Советчик по структуре (Design Advisor)

Советчик по структуре — это отличный инструмент, помогающий подобрать оптимальную структуру базы данных для заданной рабочей нагрузки SQL. Советчик по структуре может помочь создать структуру индексов, MQT, MDC и сегментирования базы данных. Советчик по структуре можно открыть из Центра управления; щелкните правой кнопкой мыши на базе данных и выберите пункт *Советчик по структуре*, как показано на рис. 8.4.

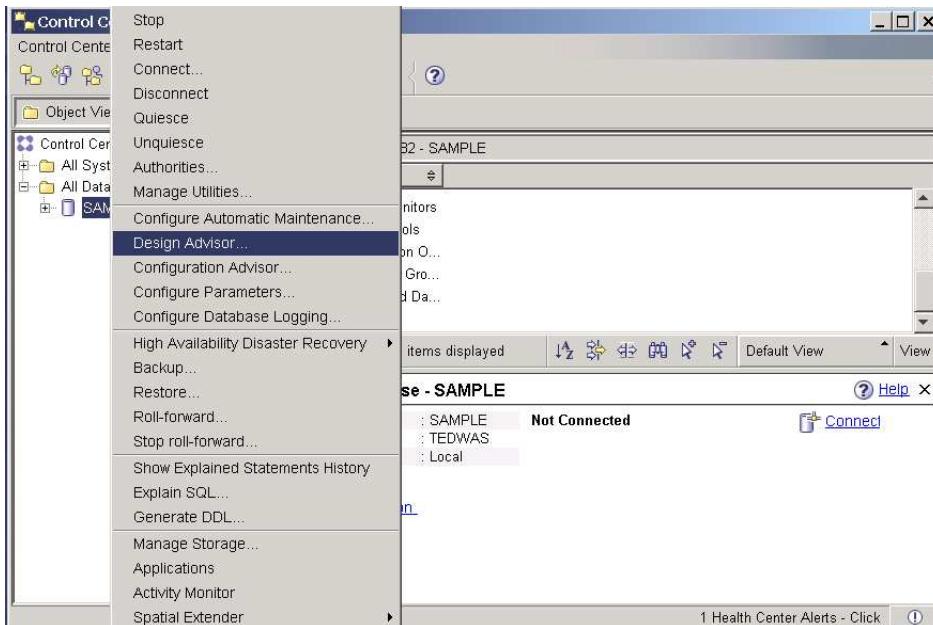


Рисунок 8.4. Запуск Советчика по структуре через Центр управления

На рис. 8.5 показан Советчик по структуре. Выполните предлагаемые мастером шаги, чтобы получить от DB2 рекомендации по структуре.

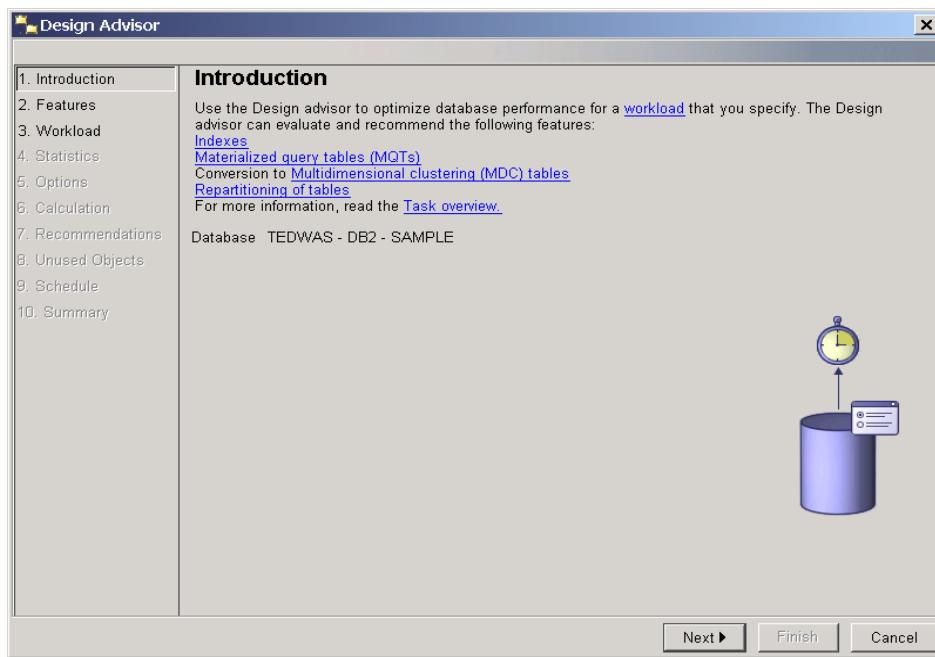


Рисунок 8.5. Советчик по структуре

8.6 Целостность на уровне ссылок

Благодаря целостности на уровне ссылок база данных может управлять отношениями между таблицами. Можно установить между таблицами тип отношений «родитель — потомок» (parent-child), как показано на рис. 8.6. На рисунке представлено две таблицы, DEPARTMENT (отдел) и EMPLOYEE (сотрудник), связанные за номером отдела. Столбец WORKDEPT таблицы EMPLOYEE может содержать только номера отделов, уже существующие в таблице DEPARTMENT. Это объясняется тем, что в нашем примере DEPARTMENT является таблицей-родителем, или главной таблицей, а EMPLOYEE — таблицей-потомком, или подчиненной таблицей. На рисунке также показан оператор CREATE TABLE для таблицы EMPLOYEE, необходимый для определения отношений.

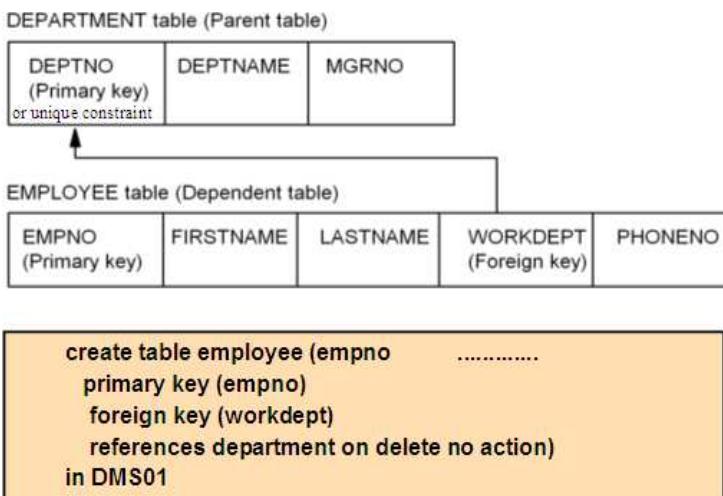


Рисунок 8.6. Пример целостности на уровне ссылок между таблицами

При рассмотрении целостности на уровне ссылок часто употребляются понятия, описанные в Таблице 8.1.

| Понятие | Описание |
|---------------------|---|
| Главная таблица | Управляющая таблица данных, в которой существует родительский ключ |
| Подчиненная таблица | Таблица, зависимая от данных главной таблицы. Она также содержит внешний ключ. Стока может существовать в подчиненной таблице, только если в главной таблице уже существует совпадающая строка. |
| Первичный ключ | Определяет родительский ключ главной таблицы. Не может содержать значения NULL; значения должны быть уникальными. Первичный ключ состоит из одного или нескольких столбцов таблицы. |
| Внешний ключ | Ссылается на первичный ключ главной таблицы |

Таблица 8.1. Ключевые понятия целостности на уровне ссылок

Данные таблиц могут привязываться к данным одной или нескольких таблиц при соблюдении целостности на уровне ссылок. На значения данных также можно наложить ограничения, чтобы они соответствовали определенным качествам или бизнес-правилам. К примеру, если в столбце таблицы указывается пол человека, можно ограничить допустимые значения: «М» — мужчина, «Ж» — женщина.

new in V9.7

8.7 Эволюция схем

По мере изменений потребностей бизнеса вспомогательные информационные технологии (ИТ) и системы также должны меняться. В отношении баз данных это означает, что необходимо создавать новые таблицы, отбрасывать или модифицировать существующие таблицы, должна меняться логика триггеров и т. п. Хотя на первый взгляд такие изменения довольно легко внести, на самом деле они могут быть комплексными и сложными в реализации. Потребуются длительные периоды обслуживания, а также выполнение сложных рискованных процедур. Одной из причин того, что раньше такие изменения было очень сложно воплотить, являлось то, что в DB2 обязательной была постоянная согласованность всех объектов.

Поэтому действия, которые влияли на объекты, зависимые от изменяемых объектов, либо запрещались, либо приводили к отбрасыванию подчиненных объектов. На рис. 8.7 представлен образец сценария.

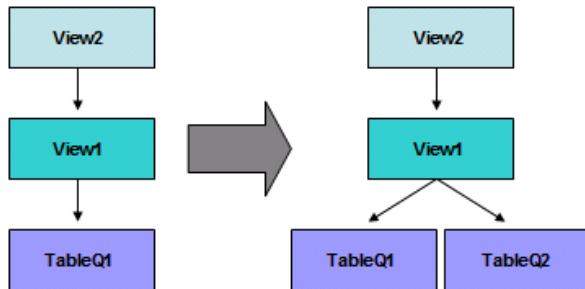


Рисунок 8.7. Образец сценария эволюции схем

В качестве примера для рис. 8.7 рассмотрим ситуацию, когда определение представления *View1* необходимо изменить так, чтобы оно основывалось не на таблице *TableQ1* с финансовыми показателями компании за первый квартал, а на таблицах *TableQ1* и *TableQ2* с данными за первый и второй квартал, соответственно. При стандартном подходе нужно было бы удалить представление *View1* и воссоздать его с новым определением; однако *View2* зависит от *View1*. До выхода DB2 9.7 сервер данных DB2 не разрешил бы удалить *View1* из-за наличия подчиненного *View2*. Сначала нужно было бы удалить *View2*, затем *View1*, а после этого — воссоздать оба представления. В DB2 9.7 правила упростились. Теперь допускается внесение изменений, влияющих на подчиненные объекты. Подчиненный объект (в нашем примере — *View2*) необходимо повторно проверить перед использованием, но система делает это автоматически. Этот процесс называется **автоматической перепроверкой**. Для включения или выключения автоматической перепроверки, а также определения времени её выполнения, используется параметр db cfg AUTO_REVAL. К примеру, если для этого параметра указать значение DEFERRED_FORCE, перепроверка будет отложена до обнаружения недействительного или подчиненного объекта, но при этом для создания новых подчиненных объектов можно будет воспользоваться оператором CREATE (появится предупреждение).

Среди прочих изменений, влияющий на модель подчинения, — внедрение таких особенностей, как синтаксис CREATE OR REPLACE для представлений, функций, процедур, триггеров, алиасов и пр. Например:

```
create or replace procedure p1 begin ... end
```

Использование такой синтаксической структуры подразумевает, что если объект (к примеру, процедура P1) не существовал, он будет создан. Если объект существовал, будет выполнена его замена. Большое значение для подчиненности объектов имеет именно последнее свойство. При замене P1 все подчиненные P1 объекты автоматически перепроверяются. По такому же принципу работают такие нововведения, как RENAME COLUMN, а также оператор ALTER COLUMN для изменения типов данных, который к тому же был расширен и теперь поддерживает больше вариантов изменений.

Такое смежное понятие, как **плавная инвалидация** (soft invalidation), позволяет пользователям отбрасывать объект, даже если он используется в других активных

транзакциях. Новые транзакции не смогут получить доступ к такому отброшенному объекту.

8.8 Краткий обзор

В этой главе рассматривались объекты баз данных в DB2: что они собой представляют, как создаются и каким образом используются. Мы ознакомились со схемами баз данных и сравнили их с новыми общими синонимами, а также противопоставили общие синонимы частным.

Затем мы подробно обсудили таблицы и их элементы: типы данных (встроенные и пользовательские), столбцы идентификации, объекты последовательностей и глобальные временные таблицы. После этого были рассмотрены представления и индексы, а также применение графического интерфейса пользователя Советчика по структуре для улучшения возможности доступа к данным в таблицах и их извлечения.

Наконец, мы изучили целостность на уровне ссылок для определения отношений между таблицами и новое понятие эволюции схем, которое позволяет менять объекты данных без лишних усложнений.

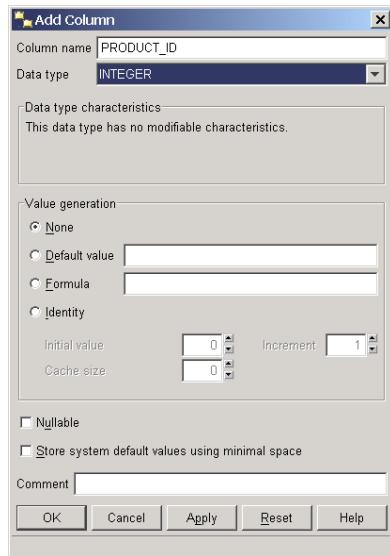
8.9 Упражнения

До сих пор для иллюстраирования различных понятий мы использовали существующие таблицы базы данных `SAMPLE`. Со временем вам придется создать собственные таблицы в базе данных. В этом упражнении мы воспользуемся *Мастером по созданию таблиц*, чтобы создать две новые таблицы в базе данных `SAMPLE`.

Процедура

1. Запустите *Мастер по созданию таблиц*, как описывалось ранее в этой главе. (*Центр управления* -> *Все базы данных* -> `SAMPLE` -> (правый щелчок) *Объекты таблиц* -> *Создать...*)
2. Укажите имя таблицы, определения столбцов и требуемые ограничения. Таблица будет использоваться для хранения информации о канцелярских товарах, используемых для проекта в базе данных `SAMPLE`. При каждой покупке канцтоваров в эту таблицу будет добавляться строка. В таблице будет шесть столбцов:
 - `product_id`: уникальный идентификатор покупаемого товара
 - `description`: описание товара
 - `quantity`: купленное количество
 - `cost`: стоимость товара
 - `image`: изображение товара (если доступно)
 - `project_num`: проект, для которого куплен товар

3. На первой странице мастера в поле имени схемы введите идентификатор пользователя, под которым вы вошли в систему, и воспользуйтесь именем таблицы **SUPPLIES**. Также при желании можно добавить комментарий. Нажмите кнопку **Далее**, чтобы перейти к следующей странице мастера.
4. На этой странице можно добавить в таблице столбцы. Для этого нажмите кнопку **Добавить**.



5. Введите имя столбца **product_id** и выберите тип данных **INTEGER**. Снимите флажок возле пункта **Допускаются пустые значения** и нажмите кнопку **Применить**, чтобы определить столбец.
6. Повторите этот шаг для остальных столбцов таблицы, используя параметры, указанные в таблице ниже. Добавив (применив) все столбцы, нажмите кнопку **OK**. В результате должен отобразиться итоговый список созданных столбцов. Нажмите кнопку **Далее**, чтобы перейти к следующей странице мастера.

| Имя столбца | Атрибуты |
|---------------------|---|
| product_id (готово) | INTEGER, пустые значения НЕ допускаются |
| description | VARCHAR, длина 40, пустые значения НЕ допускаются |
| quantity | INTEGER, пустые значения НЕ допускаются |
| cost | DECIMAL, точность 7, масштаб 2, пустые значения НЕ допускаются |
| image | BLOB, 1 МБ, допускаются пустые значения, данные не регистрируются |
| project_num | CHAR, длина 6, пустые значения НЕ допускаются |

Примечание. Опцию «Не регистрируются» можно задать при определении столбцов LOB. Она является обязательной для столбцов размером больше

чем 1 ГБ. Её также рекомендуют использовать для столбцов LOB с размером больше чем 10 МБ, поскольку изменения больших столбцов могут быстро заполнить файл журнала. Даже если используется вариант «Не регистрируются», изменения, внесенные в LOB-файлы во время транзакций, можно успешно откатить. Также обратите внимание на то, что столбец изображения — единственный столбец, для которого допускаются пустые значения. Как вы считаете, почему этот столбец определен именно так?

7. На этом этапе уже собрана вся необходимая информация для создания таблицы. Пропуская остальные окна мастера, вы выбираете для соответствующих параметров значения по умолчанию. Ключи и ограничения всегда можно добавить уже после создания таблицы.
8. Добавьте в таблицу ограничение для значений столбца *quantity*. На странице мастера *Ограничения* нажмите кнопку *Добавить*. В поле *Имя проверочного ограничения* введите *valid_quantities*. В поле *Условие проверки* введите *quantity > 0*

Нажмите кнопку *OK*. На странице *Ограничения* будет показана информация обо всех добавленных ограничениях. Нажмите кнопку *Далее*, чтобы перейти к следующей странице мастера.

9. Можно продолжить использование мастера и редактирование прочих параметров таблицы. В качестве альтернативы можно перейти к странице *Сводка* или просто нажать кнопку *Готово*, чтобы создать таблицу.
10. В Центре управления, в окне дерева объектов, нажмите на папку *Таблицы* в базе данных *SAMPLE*. Теперь в списке должна отображаться таблица, которую мы только что создали. Для отображения изменений может потребоваться обновление вида Центра управления.
11. Теперь проверим неявное приведение типов, воспользовавшись таблицей *STAFF* базы данных *SAMPLE*. Выполните следующее:

```
C:\>db2 describe table staff
```

Обратите внимание, что столбец идентификаторов определен как *SMALLINT*.

```
C:\>db2 select * from staff where id = '350' --> Примечание: '350' —
строковое значение
```

```
C:\>db2 select * from staff where id = 350 --> Примечание: '350' —
числовое значение
```

В обоих случаях на выходе получим такой результат:

| ID | NAME | DEPT | JOB | YEARS | SALARY | COMM |
|-----|--------|------|-------|-------|----------|--------|
| 350 | Gafney | 84 | Clerk | 5 | 43030.50 | 188.00 |

Используя строковое значение '350' в первом операторе *SELECT*, DB2 выполняет неявное приведение типов к числовому значению (*SMALLINT*).

9

Глава 9. Утилиты перемещения данных

Инструменты и команды, рассмотренные в этой главе, используются для перемещения данных в пределах одной базы данных или между разными базами данных на одинаковых или разных платформах. На рис. 9.1 представлен обзор утилит перемещения данных.

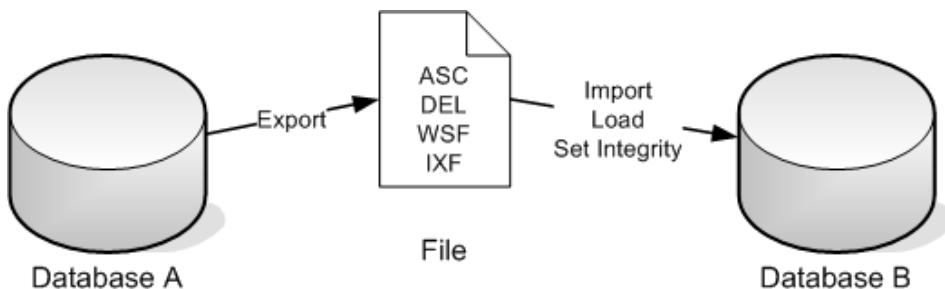


Рисунок 9.1. Утилиты перемещения данных

На рис. 9.1 показано две базы данных, А и В. С помощью утилиты EXPORT можно экспорттировать данные из таблицы в файл. Файл может иметь любой из перечисленных форматов:

- ASC = ASCII
- DEL = Delimited ASCII (ASCII с разделителями)
- WSF = Worksheet format (формат рабочей страницы)
- IXF = Integrated Exchange Format (формат интегрированного обмена)

Файлы ASC и DEL — это текстовые файлы, которые можно открыть и просмотреть в любом текстовом редакторе. WSF — это формат, используемый для перемещения данных в редактор электронных таблиц, например Excel или Lotus 1-2-3. IXF — это формат, который включает не только данные, но и язык описания данных (Data Definition Language, DDL) соответствующей таблицы. Формат IXF удобен, поскольку если необходимо восстановить таблицу, это можно сделать непосредственно из экспортированного файла с форматом IXF; при использовании других форматов этого сделать нельзя.

После экспорта данных в файл можно воспользоваться утилитой IMPORT, чтобы импортировать данные из файла в другую таблицу. Для форматов ASC, DEL и WSF таблицу необходимо создать предварительно, а для формата IXF этого делать не нужно. Еще один способ загрузки данных в таблицу — использование утилиты LOAD. Утилита LOAD работает быстрее, поскольку обращается непосредственно к страницам базы данных, не взаимодействуя с ядром DB2; этим способом не предусмотрена проверка ограничений и использование триггеров. Для обеспечения

согласованности загруженных утилитой LOAD данных после использования этой утилиты часто выполняют команду SET INTEGRITY.

В последующих разделах утилиты EXPORT, IMPORT и LOAD рассмотрены более подробно.

Примечание.

Чтобы узнать о работе с утилитами перемещения данных более подробно, просмотрите видео: <http://www.channeldb2.com/video/video/show?id=807741&Video:4262>

9.1 Утилита EXPORT

Утилита EXPORT используется для извлечения данных из таблицы в файл, как уже упоминалось выше. На самом деле при этом выполняется SQL-операция SELECT. В представленном ниже примере выполняется экспорт 10-ти строк таблицы `employee` в файл `employee.ixf` в формате IXF.

```
EXPORT TO employee.ixf OF IXF
  SELECT * FROM employee
    FETCH FIRST 10 ROWS ONLY
```

Попробуйте выполнить операцию, показанную в примере выше. Таблица «employee» является частью базы данных SAMPLE, поэтому сначала нужно выполнить подключение к этой базе данных, созданной в предыдущей главе.

Если вы предпочитаете работать с GUI-инструментами, можете запустить утилиту EXPORT из Центра управления, как показано на рис. 9.2.

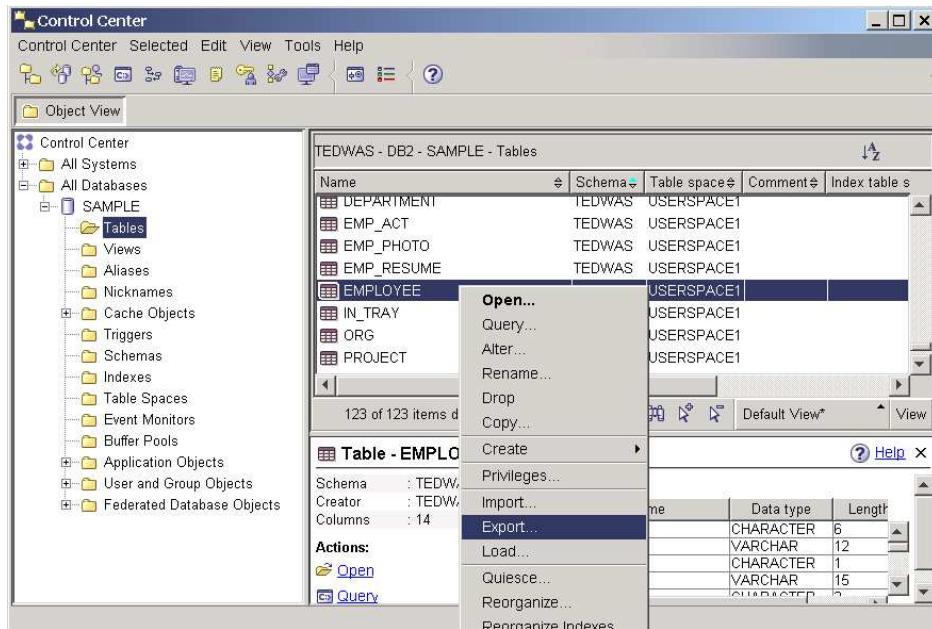


Рисунок 9.2. Запуск диалогового окна «Экспортировать таблицу»

Как показано на рисунке, сначала необходимо выбрать таблицу «employee», щелкнув на неё один раз, а затем щелкнуть на ней правой кнопкой мыши и во всплывающем меню выбрать пункт «Экспорт». В результате откроется окно мастера. Просто выполните предлагаемые мастером шаги, чтобы завершить операцию.

9.2 Утилита IMPORT

Утилита IMPORT используется для загрузки данных из файла в таблицу, как уже упоминалось выше. На самом деле при этом выполняется SQL-операция INSERT. При выполнении операции INSERT срабатывают все триггеры, незамедлительно вступают в действие все ограничения и используется буферный пул базы данных. В следующем примере все данные из файла employee.ixf в формате IXF загружаются в таблицу *employee_copy*. Рекомендуем попробовать выполнить представленную в примере операцию, но для этого необходимо было воспользоваться утилитой EXPORT в предыдущем разделе.

```
IMPORT FROM employee.ixf OF IXF
  REPLACE_CREATE
  INTO employee_copy
```

Опция REPLACE_CREATE — одна из многих опций, доступных в утилите IMPORT. Эта опция заменяет содержимое таблицы *employee_copy*, если она уже существовала до использования утилиты IMPORT. Если такой таблицы еще нет, эта опция создаст её и загрузит все данные.

Если вы предпочитаете работать в Центре управления, можете запустить утилиту IMPORT, выбрав любую таблицу, щелкнув на ней правой кнопкой мыши и выбрав пункт «Импорт», как показано на рис. 9.3.

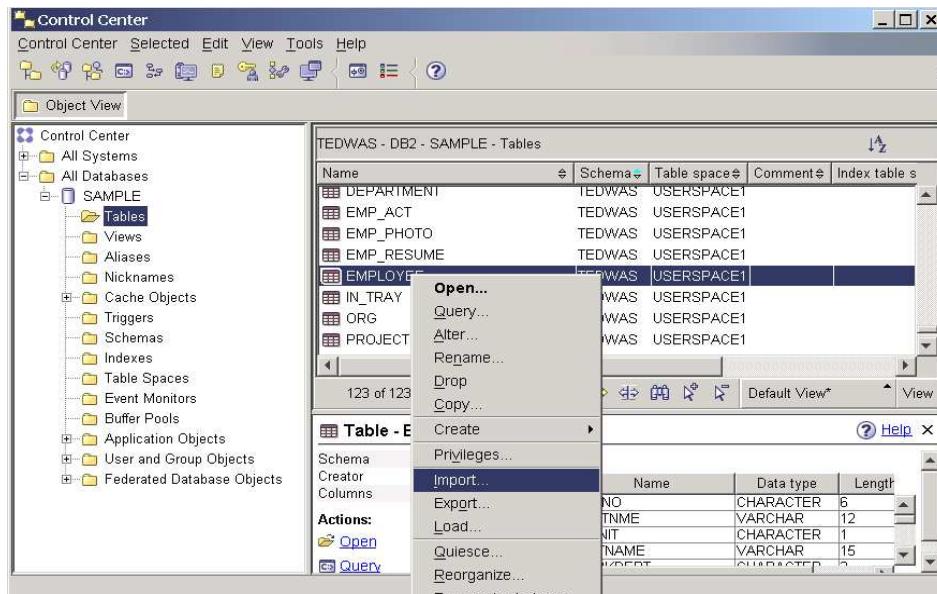


Рисунок 9.3. Запуск диалогового окна утилиты IMPORT

9.3 Утилита LOAD

Утилита LOAD — это более быстрый способ загрузки данных из файла в таблицу. Как уже упоминалось выше, утилита LOAD не проходит через ядро DB2, в связи с чем не запускаются триггеры, не используется буферный пул, а ограничения могут накладываться только отдельным действием. С другой стороны, операция LOAD выполняется быстрее, чем IMPORT, поскольку загрузчик данных низкого уровня получает доступ непосредственно к страницам данных на диске. Эта утилита работает в три этапа: LOAD, BUILD и DELETE.

В следующем примере все данные из файла `employee.ixf` в формате IXF загружаются в таблицу `employee_copy`. Одной из многих доступных в утилите LOAD опций является REPLACE. В данном случае она используется, чтобы заменить (REPLACE) всё содержимое таблицы `employee_copy`.

```
LOAD FROM employee.ixf OF IXF
    REPLACE INTO employee_copy
```

После выполнения представленной выше команды табличное пространство, в котором размещена таблица, могло перейти в состояние CHECK PENDING (ожидание проверки). Это означает, что необходимо выполнить команду SET INTEGRITY для проверки согласованности данных. В примере ниже показано, как это сделать:

```
SET INTEGRITY FOR employee_copy
    ALL IMMEDIATE UNCHECKED
```

Если вы предпочитаете работать в Центре управления, можете запустить утилиты LOAD и SET INTEGRITY, как показано на рис. 9.4 и 9.5, соответственно.

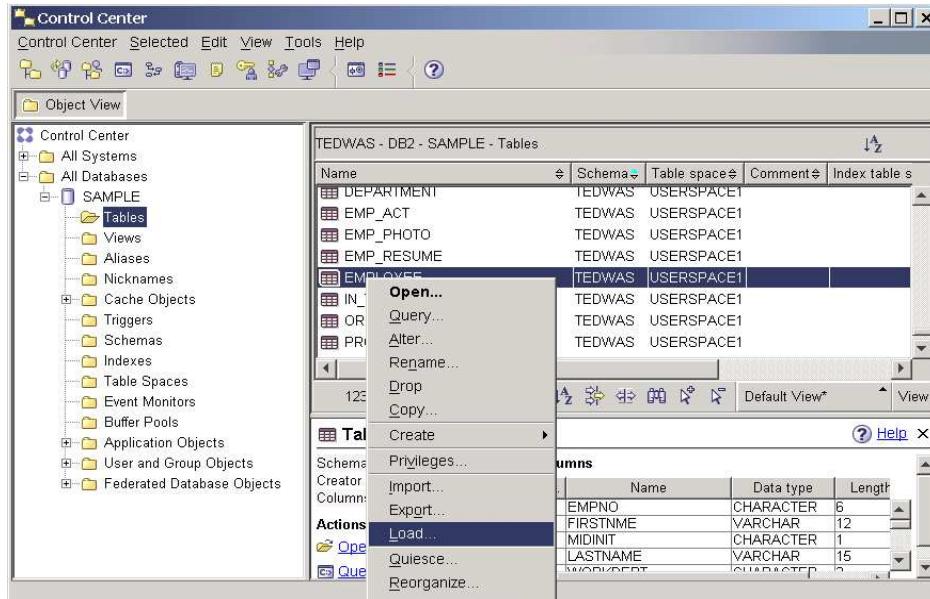


Рисунок 9.4. Запуск утилиты LOAD

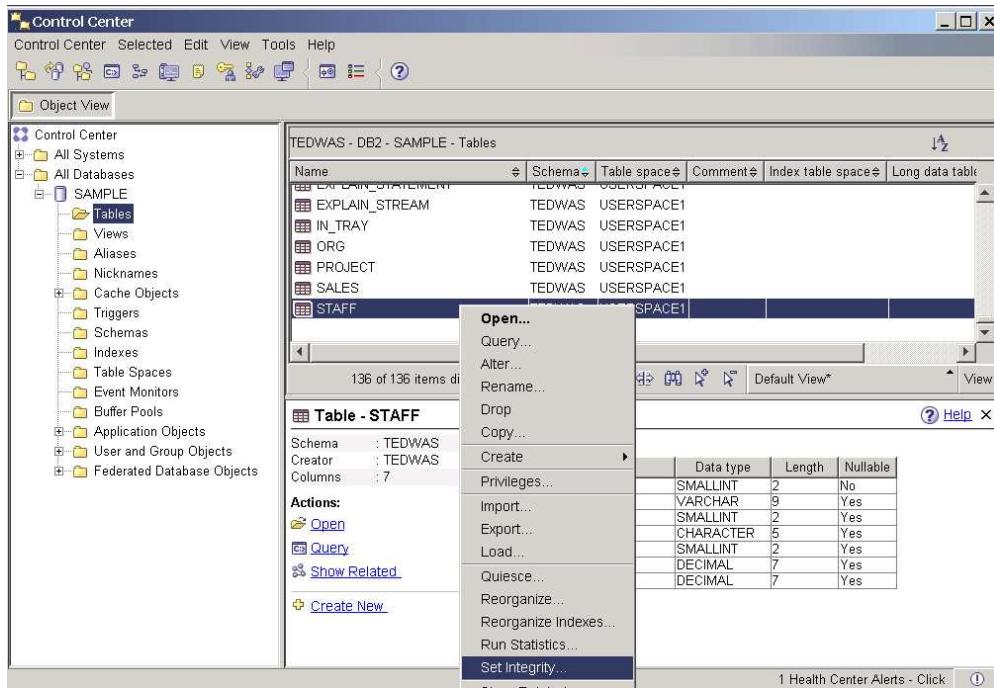


Рисунок 9.5. Запуск мастера SET INTEGRITY (Задать целостность)

9.4 Утилита db2move

Утилиты EXPORT, IMPORT и LOAD работают с одной таблицей за один раз. Хотя можно написать сценарий, генерирующий описанные выше команды для каждой таблицы базы данных, утилита db2move может сделать это автоматически. Утилита db2move может работать только с файлами IXF, при этом db2move автоматически генерирует имена файлов. В примерах ниже показано, как выполнить db2move с опциями экспорта и импорта, соответственно, для базы данных SAMPLE.

```
db2move sample export
db2move sample import
```

Утилитой db2move невозможно воспользоваться в Центре управления.

9.5 Утилита db2look

В то время как утилиты EXPORT, IMPORT, LOAD и db2move позволяют перемещать данные из одной таблицы в другую (в одной или нескольких базах данных), утилиту db2look можно использовать для извлечения операторов DDL, статистики базы данных и характеристик табличных пространств базы данных и их сохранения в файле сценария, который позже можно выполнить в другой системе. К примеру, если

необходимо клонировать базу данных с сервера DB2 с ОС Linux на сервер DB2 с ОС Windows, сначала следует запустить утилиту `db2look` на сервере DB2 Linux, чтобы получить структуру базы данных и сохранить её в файле сценария. Затем вы копируете этот файл сценария на сервер DB2 Windows и выполняете сценарий, чтобы начать построение клонированной базы данных. На этом этапе структура базы данных уже клонирована. Следующий шаг — запустить утилиту `db2move` с опцией экспорта на сервере DB2 Linux и скопировать все генерированные файлы на сервер DB2 Windows, а затем выполнить `db2move` с опцией импорта или загрузки. После этого база данных будет полностью клонирована с одного сервера на другой, работающий на иной платформе.

Описанный выше сценарий может понадобиться при работе с базами данных на разных платформах, например Linux и Windows. Если оба сервера работают на одной платформе, целесообразнее воспользоваться командами резервного копирования и восстановления, т. к. такой процесс будет более простым и прямолинейным. Команды `backup` и `restore` более подробно рассматриваются в одной из последующих глав этой книги.

В представленном ниже примере выполняется извлечение макета табличных пространств и пулов буферов, а также DDL-операторов из базы данных `SAMPLE` и их сохранение в файле `sample.ddl`. Рекомендуем выполнить показанную ниже команду и ознакомиться с выходным текстовым файлом `sample.ddl`.

```
db2look -d sample -l -e -o sample.ddl
```

The diagram shows the `db2look` command with four annotations pointing to specific options:

- `-d sample`: database name
- `-l`: layout (tablespaces & bufferpools)
- `-e`: extract DDL
- `-o sample.ddl`: output file

Мы не можем рассмотреть в этой книге все опции команды `db2look`, поскольку их слишком много; однако с помощью метки `-h` можно вывести краткое описание доступных опций:

```
db2look -h
```

Утилиту `db2look` можно также запустить из Центра управления (см. рис. 9.6).

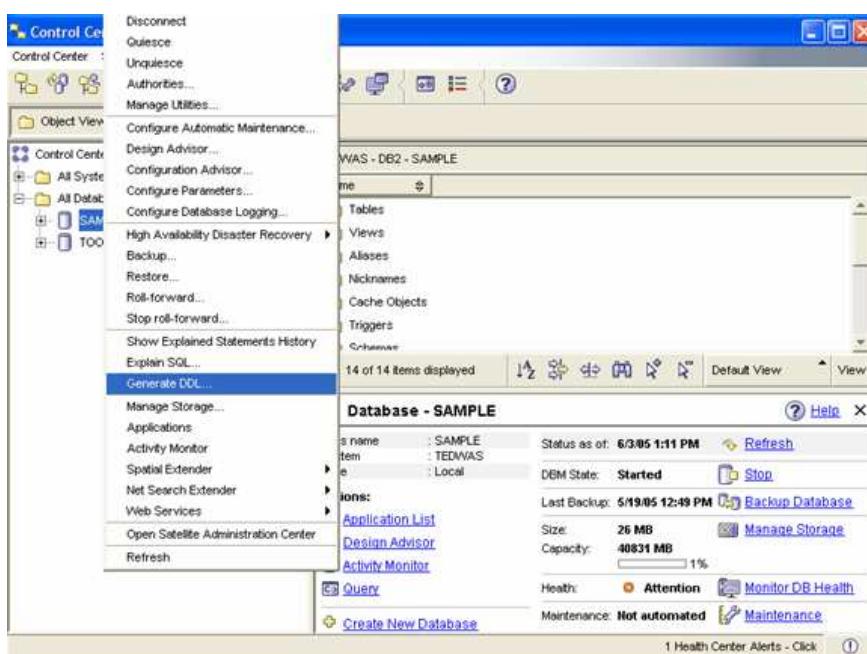


Рисунок 9.6. Извлечение DDL в Центре управления

Как показано на рис. 9.6, выберите базу данных, из которой необходимо получить DDL, щелкните на ней правой кнопкой мыши и выберите пункт «Генерировать DDL». Откроется окно «Генерировать DDL» с несколькими вариантами извлечения (см. рис. 9.7).

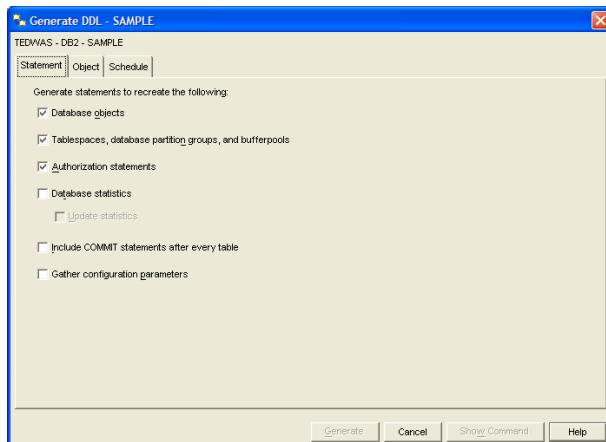


Рисунок 9.7. Извлечение DDL в Центре управления

9.6 Краткий обзор

В этой главе мы рассмотрели различные функции экспорта и импорта в DB2. Начав с разных форматов экспорта (ASC, DEL, WSF и IXF), мы перешли к подробному изучению утилиты EXPORT. Затем мы обсудили утилиты импорта IMPORT и LOAD, а также необходимость оператора SET INTEGRITY при использовании LOAD.

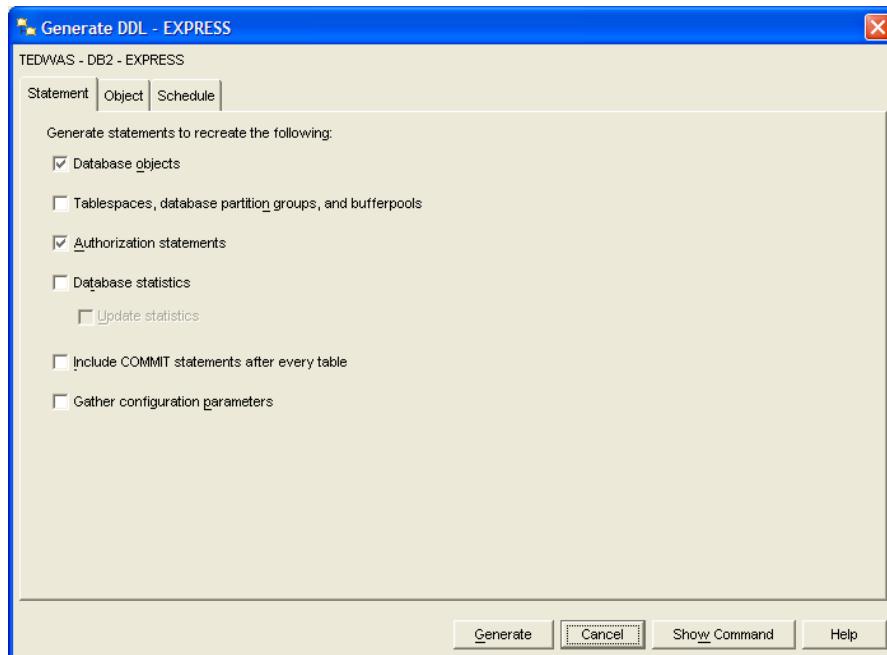
Команда db2move предоставляет средства для упрощения процесса передачи данных при их экспорте и импорте. Более сложная команда db2look дает возможность извлечь и сохранить все элементы базы данных, необходимые для её полного автономного воссоздания.

9.7 Упражнения

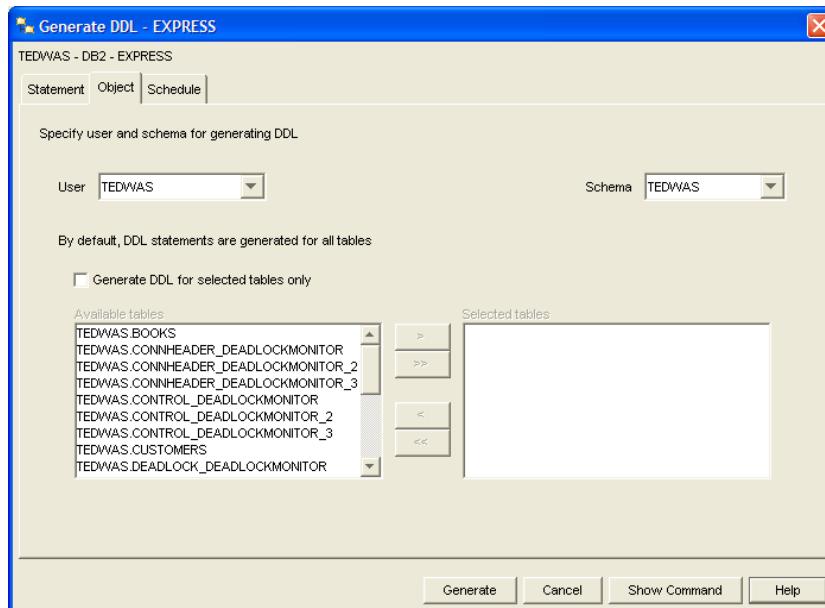
При клонировании данных основной целью является воссоздание базы данных наиболее прямолинейным и воспроизводимым способом. Обычно для этого используются сценарии SQL, которые можно выполнить сразу после установки DB2. В этом упражнении мы извлечем определения объектов из базы данных *EXPRESS* (созданной в предыдущих упражнениях) с помощью Центра управления.

Процедура

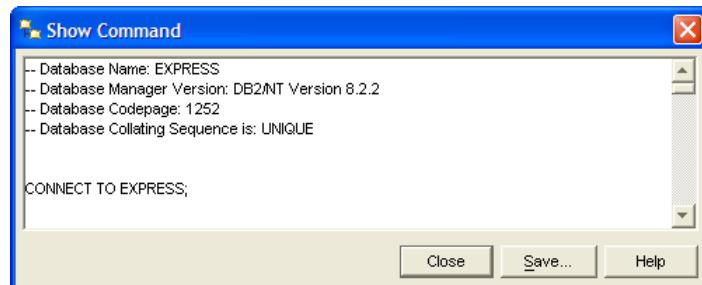
1. Откройте Центр управления.
2. Щелкните правой кнопкой мыши базу данных *EXPRESS* на панели дерева объектов и выберите в меню пункт *Генерировать DDL*. Откроется диалоговое окно *Генерировать DDL*.
3. В окне *Генерировать DDL* задайте параметры генерируемого DDL, как показано ниже. Если вы создавали в своей среде дополнительные объекты, например табличные пространства, буферные пулы и т. п., их необходимо выбрать в этом окне. Поскольку вы еще не создавали такие объекты, снимите флажок возле соответствующего пункта. Мы не включаем статистику базы данных, поскольку набор статистических данных в производственной среде, вероятно, будет отличаться от данных в среде разработки. Аналогично параметры конфигурирования также будут иными. Если в вашей среде всё настроено именно так, как будет выглядеть при развертывании, можете включить эти дополнительные параметры.



4. Перейдите на вкладку **Объект**. Можно выбрать, для каких именно объектов будет генерироваться DDL. В данном случае выберите пользователя и схему, которые использовались при создании всех объектов, и сгенерируйте DDL для всех объектов этой схемы. Нажмите кнопку **Генерировать**, чтобы запустить генерирование DDL.



5. Просмотрите полученный DDL. В результате выполнения предыдущего шага был получен единый сценарий со всеми SQL-операторами для выбранных объектов. Теперь организуем в этом сценарии логические группы.
6. Создайте каталог C:\express в файловой системе и сохраните сгенерированный файл DDL в этот каталог под именем schema.ddl. (Нажмите кнопку Сохранить)



7. Откройте только что сохраненный файл в Редакторе команд. (Подсказка: в Редакторе команд выберите Файл -> Открыть)
8. Хотя нам был нужен только DDL для таблиц, вы увидите, что также включен DDL для прочих объектов базы данных. Переместите все операторы CREATE TRIGGER в отдельный новый файл с именем triggers.ddl. Хотя мы создали всего один триггер, всегда рекомендуется разделять объекты по типу.
9. На данном этапе также рекомендуем удалить следующее:
 - операторы базы данных CONNECT TO
 - операторы DISCONNECT

Теперь у вас должно быть два сценария:

C:\express\schema.ddl, содержащий DDL для таблиц, представлений, индексов и ограничений;

C:\express\triggers.ddl, содержащий DDL для триггеров.

10. Очистите сценарий для развертывания:
 - Удалите ненужные комментарии (напр., -- CONNECT TO...)
 - Вынесите функции и процедуры в отдельные файлы (полезно при использовании большого количества функций и процедур). Также можно сгруппировать их по функции или приложению (например, billing.ddl, math.ddl, stringfunc.ddl и т. п.)
11. Вы могли заметить, что для обозначения конца триггеров, функций и процедур используется специальный символ (@). Это необходимо для обозначения завершения оператора CREATE <объект>, в отличие от завершения процедурного оператора в рамках объекта.

10

Глава 10. Безопасность базы данных

В этой главе рассматривается вопрос безопасности DB2. На рис. 10.1 схематически представлен общий обзор.

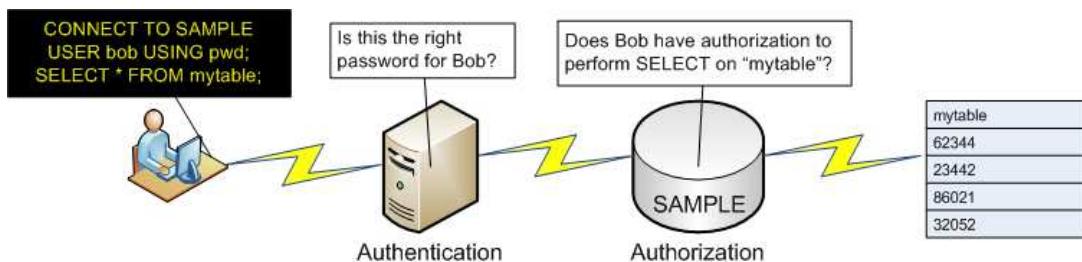


Рисунок 10.1. Обзор безопасности DB2

Как показано на рис. 10.1, безопасность DB2 состоит из двух частей:

- **Аутентификация**

Это процесс, с помощью которого проверяется личность пользователя. Аутентификацию выполняет средство обеспечения безопасности вне DB2 посредством подключаемого модуля безопасности. Используемый по умолчанию модуль безопасности зависит от безопасности операционной системы, но можно также использовать подключаемые модули для Kerberos, LDAP или же создать собственный пользовательский подключаемый модуль для аутентификации. При использовании подключаемого модуля аутентификации на базе ОС по умолчанию идентификатор пользователя и пароль высылаются на сервер данных (например, как часть оператора подключения). Затем сервер данных активизирует аутентификацию ОС для проверки идентификатора пользователя и пароля.

- **Авторизация**

На данном этапе DB2 проверяет, можно ли пользователю, прошедшему аутентификацию, выполнять запрашиваемую операцию. Авторизационная информация хранится в каталоге DB2 и файле конфигурации менеджера базы данных.

К примеру, на рис. 10.1 показано, что пользователь *bob* подключается к базе данных **SAMPLE** с помощью следующего оператора:

```
CONNECT TO sample USER bob USING pwd
```

И *bob*, и *pwd* передаются на системную или внешнюю службу аутентификации для получения подтверждения того, что пользователь с именем *bob* уже определен и предоставленный пароль соответствует этому имени пользователя. Если эта часть процесса проходит успешно, операционная система возвращает управление

безопасностью к DB2. Затем, когда пользователь *bob* выполняет оператор, например:

```
SELECT * FROM mytable
```

DB2 перенимает управление безопасностью для проверки авторизации и подтверждения того, что пользователь *bob* имеет привилегии для выполнения SELECT в таблице *mytable*. Если авторизация не проходит успешно, DB2 отображает сообщение об ошибке. В противном случае в таблице *mytable* выполняется указанный оператор.

Примечание.

Чтобы получить более подробную информацию о безопасности DB2, просмотрите видео: <http://www.channeldb2.com/video/video/show?id=807741:Video:4267>

10.1 Аутентификация

Хотя фактическая аутентификация выполняется операционной системой с помощью используемого по умолчанию подключаемого модуля безопасности (или другого внешнего средства обеспечения безопасности), DB2 все же определяет, на каком уровне проходит аутентификация.

Задаваемый на сервере DB2 параметр конфигурации базы данных AUTHENTICATION имеет несколько возможных значений. К примеру, если для этого параметра установлено значение SERVER (по умолчанию), аутентификация выполняется операционной системой или внешним средством обеспечения безопасности на сервере данных. Другой пример — если для параметра AUTHENTICATION установлено значение CLIENT, аутентификация выполняется операционной системой или внешним средством обеспечения безопасности на компьютере клиента. Это показано на рис. 10.2.

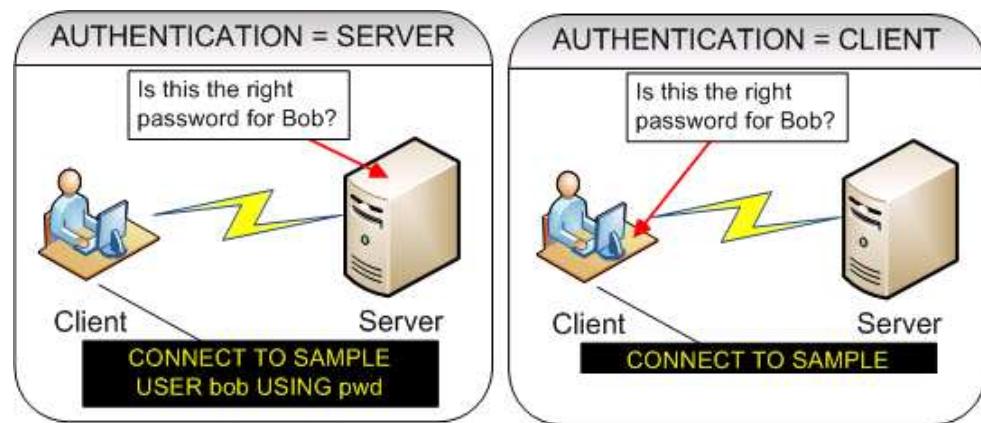


Рисунок 10.2. Место выполнения аутентификации

Для параметра AUTHENTICATION можно задать одно из значений, перечисленных в Таблице 10.1.

| Команда | Описание |
|--------------------------------|--|
| SERVER (по умолчанию) | Аутентификация выполняется на сервере данных |
| CLIENT | Аутентификация выполняется на компьютере клиента |
| SERVER_ENCRYPT | Аналогично команде SERVER, но идентификатор пользователя и пароль передаются по сети в зашифрованном виде |
| KERBEROS | Аутентификация выполняется с использованием механизма обеспечения безопасности Kerberos |
| SQL_AUTHENTICATION_DATAENC | Аутентификация на сервере данных, а все данные из результатов запросов к базе данных обязательно передаются в зашифрованном виде |
| SQL_AUTHENTICATION_DATAENC_CMP | Как и выше, но шифрование данных используется, только если доступно |
| GSSPLUGIN | Для аутентификации используется внешний механизм обеспечения безопасности подключаемого модуля GSS на базе API |

Таблица 10.1. Действительные значения параметра AUTHENTICATION

10.2 Авторизация

Авторизация состоит из привилегий, полномочий, ролей и учетных данных системы управления доступом на уровне меток (LBAC), которые хранятся в системных таблицах DB2 и находятся под управлением DB2.

Привилегия дает возможность пользователю выполнять один тип операций в базе данных, например CREATE, UPDATE, DELETE, INSERT и пр.

Роль позволяет группировать различные привилегии, предоставляемые пользователю, группе или другим ролям.

Полномочие — это предварительно определенная роль, состоящая из нескольких привилегий.

Учетные данные LBAC включают политики и метки, поддерживающие детализированный доступ определенных пользователей к заданным строкам и столбцам. LBAC не входит в состав DB2 Express-C, но в Главе 2 можно узнать об этой службе более подробно.

10.2.1 Привилегии

На рис. 10.3 показаны некоторые привилегии, используемые в DB2.

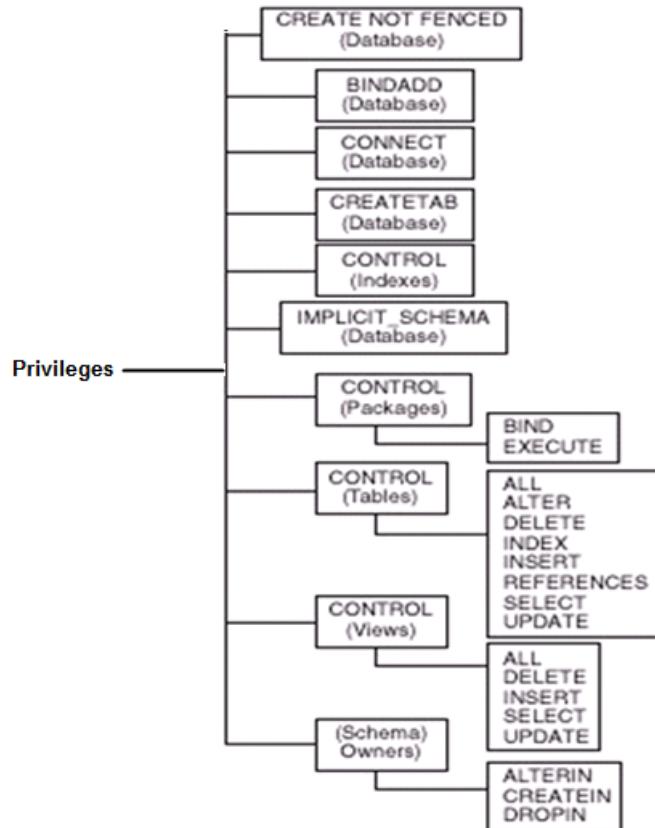


Рисунок 10.3. Перечень некоторых привилегий DB2

Предоставление пользователю или группе привилегий CONTROL означает, что они также смогут предоставлять привилегии другим пользователям или группам. С информацией о других привилегиях можно ознакомиться в информационном центре DB2.

10.2.2 Полномочия

Полномочия делятся на две группы:

- полномочия уровня экземпляра: эти полномочия могут работать на уровне экземпляра, к примеру, SYSADM;
- полномочия уровня базы данных: эти полномочия могут работать только на уровне базы данных, к примеру, DBADM.

10.2.2.1 Полномочия уровня экземпляра

В Таблице 10.2 перечислены полномочия уровня экземпляра.

| Полномочие | Описание |
|------------|--|
| SYSADM | Управление экземпляром как единственным целым |
| SYSCTRL | Администрирование экземпляра менеджера базы данных |
| SYSMAINT | Обслуживание баз данных в экземпляре |
| SYSMON | Мониторинг экземпляра и его баз данных |

Таблица 10.2. Полномочия уровня экземпляра

Чтобы предоставить полномочия SYSADM, SYSCTRL, SYSMAINT или SYSMON группе, такие параметры DBM CFG, как SYSADM_GROUP, SYSCTRL_GROUP, SYSMAINT_GROUP и SYSMON_GROUP, соответственно, можно назначить группе операционной системы.

К примеру, чтобы предоставить полномочие SYSADM группе операционной системы *myadmns*, можно выполнить следующую команду:

```
update dbm cfg using SYSADM_GROUP myadmns
```

Каждый экземпляр DB2 имеет собственные определения групп полномочий. В Windows эти параметры по умолчанию пусты, т. е. группа локальных администраторов будет иметь полномочие SYSADM. В DB2 9.7 группа DB2ADMNS (если включена расширенная безопасность) и локальная системная учетная запись также будут иметь полномочие SYSADM. Авторизационный идентификатор для локальной системной учетной записи — SYSTEM. В Linux группой владельцев экземпляра является используемая по умолчанию группа SYSADM.

На рис. 10.4, полученном из информационного центра DB2, проиллюстрированы различные полномочия уровня экземпляра и различные доступные для них функции. Как показано на рисунке, полномочие SYSADM включает все функции SYSCTRL и дополнительные функции, полномочие SYSCTRL — все функции SYSMAINT и дополнительные функции и т. д.

new in
V9.7

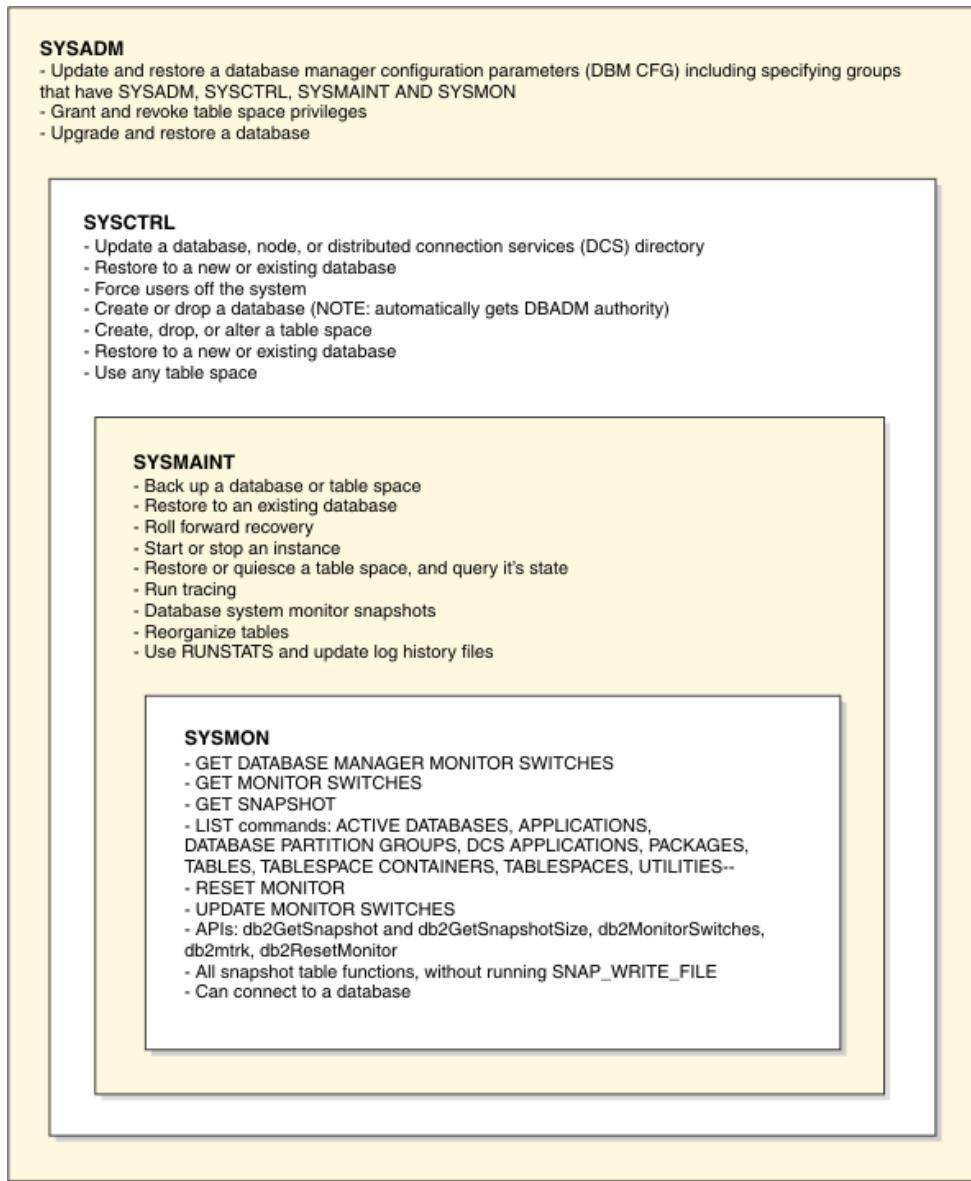


Рисунок 10.4. Полномочия уровня экземпляра и их функции

10.2.2.2 Полномочия уровня базы данных

В Таблице 10.3 перечислены полномочия уровня базы данных.

| Полномочие | Описание |
|----------------------------------|---|
| SECADM | Управление безопасностью в базе данных |
| DBADM | Администрирование базы данных |
| new in V9.7 ACCESSCTRL | Предоставление и отмена полномочий и привилегий (за исключением полномочий SECADM, DBADM, ACCESSCTRL и DATAACCESS. Обратите внимание, что для предоставления и отмены этих полномочий необходимо иметь полномочие SECADM) |
| new in V9.7 DATAACCESS | Предоставление возможности доступа к данным в базе данных |
| SQLADM | Мониторинг и регулировка SQL-запросов |
| WLMADM | Управление рабочими нагрузками |
| EXPLAIN | Пользователи, которые должны объяснять планы запросов (полномочие EXPLAIN не предоставляет доступ к самим данным) |

Таблица 10.3. Полномочия уровня базы данных

Чтобы предоставить полномочия уровня базы данных, воспользуйтесь оператором GRANT. К примеру, чтобы предоставить полномочие DBADM для базы данных **SAMPLE** пользователю **bob**, выполните следующее:

```
connect to sample
grant DBADM on database to user bob
```

В приведенном выше примере сначала нужно подключиться к базе данных, в данном случае — **SAMPLE**, а затем уже можно предоставить пользователю полномочие DBADM. Для предоставления полномочия DBADM и других полномочий уровня базы данных необходимо иметь полномочие SECADM.

Обратите внимание на то, что пользователи с полномочием DBADM не могут создавать табличные пространства, хотя они и являются объектами базы данных, поскольку табличное пространство работает с контейнерами (дисками) и буферными пулами (памятью), т. е. с физическими ресурсами системы. Создавать их могут только пользователи с полномочием SYSADM.

На рис. 10.5, полученном из информационного центра DB2, проиллюстрированы различные полномочия уровня базы данных и доступные для них функции.

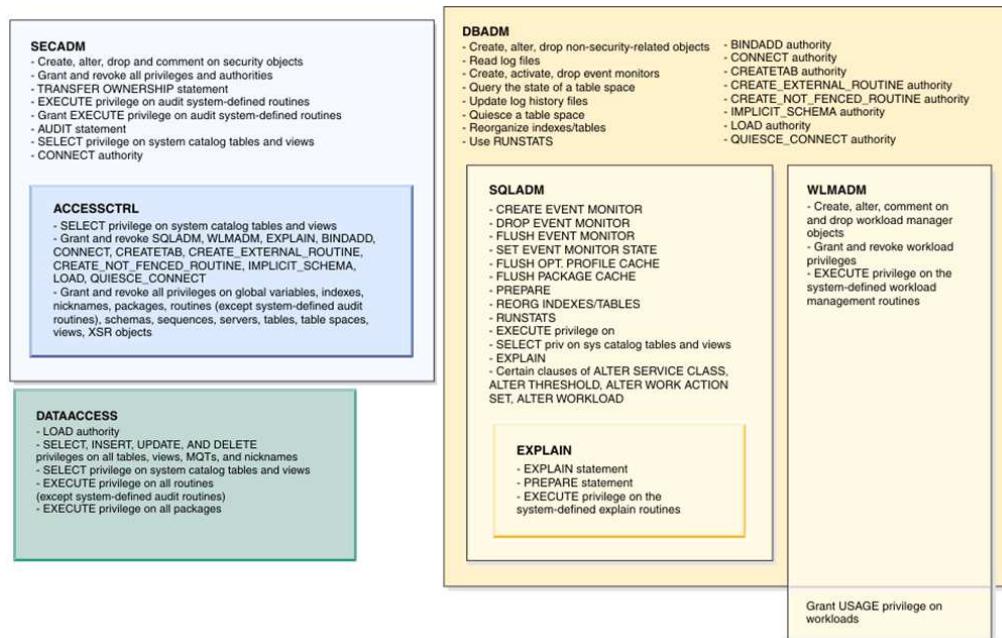


Рисунок 10.5. Полномочия уровня базы данных и их функции

new in
V9.7**Примечание.**

В DB2 9.7 для повышения конфиденциальности данных и соответствия систем управлению модель авторизации была обновлена таким образом, чтобы четко разграничивать обязанности системного администратора, администратора баз данных и администратора системы безопасности.

В общем, функциональная сфера нескольких полномочий была сокращена по сравнению с предыдущими версиями DB2. К примеру, полномочие SYSADM больше не дает права доступа к данным любой базы данных. Полномочие DBADM больше не дает права доступа к данным базы данных, которую такой пользователь администрирует. С другой стороны, пользователи с полномочием SECADM получили дополнительные функциональные возможности, например, возможность предоставлять и отменять полномочия и привилегии пользователей, ролей и групп.

Также были созданы новые полномочия, позволяющие повысить степень структурированности и уровень контроля системы безопасности. Это также уменьшает риск несанкционированного доступа к данным, поскольку пользователи получают только уровень доступа, необходимый им для выполнения должностных обязанностей.

Обновление DB2 9.7.2 также включает улучшения проверок, которые теперь дают возможность воспроизвести прошлые действия с базой данных. К примеру, если необходимо проанализировать влияние определенного запроса, оставленного несколько лет назад, на таблицы, теперь можно воспользоваться информацией проверки базы данных для получения требуемых для анализа сведений.

10.2.2.3 Включение для полномочий SYSADM и DBADM таких же функций, как в версиях DB2 до 9.7

Если необходимо, чтобы полномочие SYSADM работало так же, как в версиях DB2 до выхода DB2 9.7, следует рассмотреть два случая:

- Если пользователи с полномочием SYSADM являются создателями базы данных, они автоматически получают полномочия DATAACCESS, ACCESSCTRL, SECADM и DBADM для этой базы данных. Таким образом, пользователи с полномочием SYSADM будут иметь такие же возможности, как и в версиях DB2 до 9.7.
- Если пользователи с полномочием SYSADM не являются создателями базы данных, для получения таких же возможностей, как в предыдущих версиях DB2 (за исключением SECADM) необходимо, чтобы пользователь с полномочием SECADM выполнил операцию GRANT DBADM с полномочиями DATAACCESS и ACCESSCTRL (по умолчанию) для пользователей SYSADM соответствующей базы данных.

Рассмотрим несколько аргументов о полномочии SECADM:

- По умолчанию полномочие SECADM получает создатель базы данных.
- Если пользователь с полномочием SECADM предоставит полномочие SECADM пользователю с полномочием SYSADM, пользователь SYSADM сможет предоставлять полномочие SECADM другим пользователям.
- Если пользователь с полномочием SECADM предоставит пользователю полномочие DBADM, пользователь DBADM по умолчанию получит также полномочия DATAACCESS и ACCESSCTRL.

При выполнении миграции базы данных DB2 9.5 возможности полномочий SYSADM и DBADM не изменятся, поскольку после миграции DB2 автоматически предоставляет полномочия DBADM, DATAACCESS и ACCESSCTRL группе SYSADM. DB2 также автоматически предоставляет полномочия DATAACCESS и ACCESSCTRL всем идентификаторам авторизации, имеющим после миграции полномочие DBADM. Кроме того, DB2 автоматически предоставляет полномочие SECADM идентификатору пользователя, который выполняет миграцию, если в базе данных нет идентификатора авторизации типа USER с полномочием SECADM. Полномочие SYSADM подразумеваемую возможность предоставлять или отменять полномочия DBADM и SECADM; теперь такую возможность имеет только SECADM.

10.2.3 Роли

Роли дают администратору системы безопасости возможность предоставлять привилегии или полномочия нескольким пользователям или группам. Роли очень похожи на группы, но они определяются в DB2, а потому предоставляют некоторые преимущества. К примеру, предоставленные ролям привилегии и полномочия всегда используются при создании таких объектов, как представления или триггеры, а в случае групп это не происходит. С другой стороны, для роли нельзя назначить полномочия уровня экземпляра, например SYSADM, а только привилегии и полномочия уровня базы данных; для групп же можно назначить все привилегии и полномочия.

Для работы с ролями необходимо выполнить несколько шагов:

1. Администратор системы безопасности (SECADM) должен сначала создать роль с помощью команды, например

```
CREATE ROLE TESTER
```

2. Затем DBADM должен назначить для роли привилегии и полномочия. К примеру, чтобы предоставить привилегию SELECT для таблиц STAFF и DEPT базы данных SAMPLE роли TESTER (тестировщик), выполните следующее:

```
GRANT SELECT ON TABLE STAFF TO ROLE TESTER  
GRANT SELECT ON TABLE DEPT TO ROLE TESTER
```

3. Затем администратор системы безопасности назначает роль TESTER для пользователей RAUL и JIN:

```
GRANT ROLE TESTER TO USER RAUL, USER JIN
```

4. Затем, если пользователь JIN покидает отдел TEST, администратор системы безопасности отменяет роль TESTER для этого пользователя:

```
REVOKE ROLE TESTER FROM USER JIN
```

10.3 Особенности применения привилегий группы

Решив использовать группы вместо ролей, необходимо учитывать следующее:

- Когда группа получает привилегии, члены этой группы получают неявные привилегии, унаследованные ввиду членства в группе.
- Когда пользователи удаляются из группы, они теряют неявные привилегии группы, но сохраняют за собой явные привилегии, предоставленные им ранее. Привилегии, предоставленные пользователю явно, необходимо так же явно отзывать.

10.4 Группа PUBLIC

DB2 определяет внутреннюю группу под названием PUBLIC. Все пользователи, прошедшие идентификацию в операционной системе или сетевой службе аутентификации, по умолчанию становятся членами группы PUBLIC. При создании базы данных группа PUBLIC автоматически получает приведенные ниже привилегии:

- CONNECT
- CREATETAB
- IMPLICIT SCHEMA
- BINDADD

Для дополнительной безопасности рекомендуем отзвать у группы PUBLIC все привилегии, как показано ниже:

```
REVOKE CONNECT ON DATABASE FROM PUBLIC  
REVOKE CREATETAB ON DATABASE FROM PUBLIC  
REVOKE IMPLICIT_SCHEMA ON DATABASE FROM PUBLIC  
REVOKE BINDADD ON DATABASE FROM PUBLIC
```

10.5 Операторы GRANT и REVOKE

Операторы GRANT и REVOKE являются частью стандарта SQL и используются для предоставления привилегий пользователям, группам и ролям, а также их отзыва. Пользователь, выполняющий эти команды, должен иметь хотя бы полномочие ACCESSCTRL. Ниже показаны примеры этих операторов:

Предоставление привилегии SELECT для таблицы T1 пользователю USER1:

```
GRANT SELECT ON TABLE T1 TO USER user1
```

Предоставление всех привилегий для таблицы T1 группе GROUP1:

```
GRANT ALL ON TABLE T1 TO GROUP group1
```

Отмена всех привилегий группы GROUP1 для таблицы T1:

```
REVOKE ALL ON TABLE T1 FROM GROUP group1
```

Предоставление привилегии EXECUTE для процедуры p1 пользователю USER1:

```
GRANT EXECUTE ON PROCEDURE p1 TO USER user1
```

Отмена привилегии EXECUTE пользователя USER1 для процедуры p1:

```
REVOKE EXECUTE ON PROCEDURE p1 FROM USER user1
```

10.6 Проверка авторизации и привилегий

Проще всего проверить авторизацию и привилегии через Центр управления. На рис. 10.6 показано, как открыть диалоговое окно «Привилегии для таблицы» для таблицы *EMPLOYEE* в Центре управления.

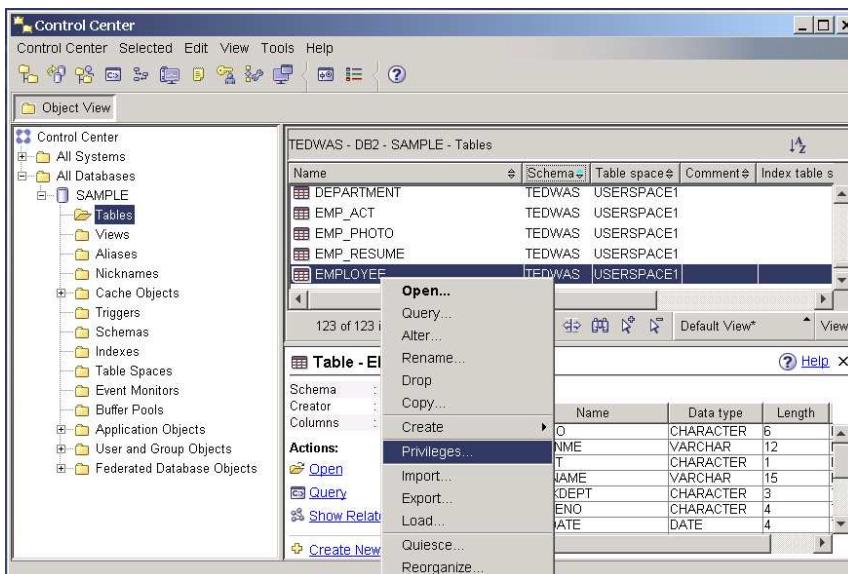


Рисунок 10.6. Запуск диалогового окна «Привилегии для таблицы»

Как показано на рис. 10.6, нужно выбрать требуемую таблицу, щелкнуть на ней правой кнопкой мыши и выбрать пункт *Привилегии*. В результате откроется диалоговое окно *Привилегии для таблицы* (см. рис. 10.7). На рисунке также объяснены различные поля и элементы этого диалогового окна.

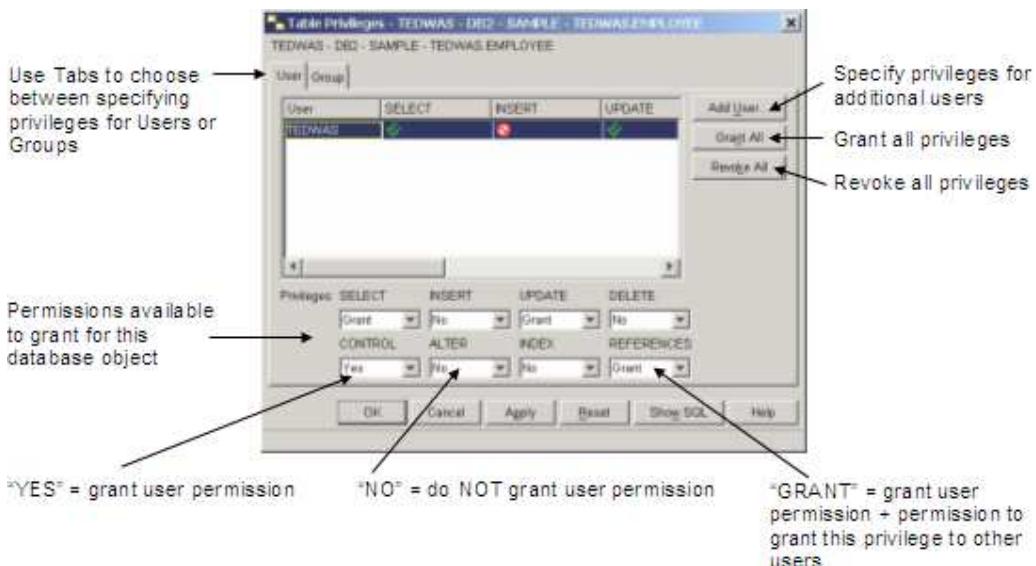


Рисунок 10.7. Диалоговое окно «Привилегии для таблицы»

Также можно сделать запрос для представлений каталога DB2 SYSCAT, в котором содержится авторизационная информация. К примеру, чтобы узнать, имеет ли пользователь **DB2ADMIN** привилегию **SELECT** в таблице **T2**, а также кто предоставил эту привилегию, можно выполнить следующий запрос:

```
SELECT grantor, grantee, selectauth
FROM syscat.tabauth
WHERE tabname = 'T2'
```

| GRANTOR | GRANTEE | SELECTAUTH |
|----------|----------|------------|
| ARFCHONG | DB2ADMIN | Y |

В примере выше пользователь **ARFCHONG** предоставил привилегию **SELECT** пользователю **DB2ADMIN**.

10.7 Расширенная безопасность в Windows

Для предотвращения доступа к файлам и каталогам DB2 (например, к тем, в которых DB2 хранит информацию об экземплярах) через операционную систему Windows,

при установке DB2 по умолчанию включает расширенную безопасность. Функция расширенной безопасности создает две группы:

- DB2ADMNS: эта группа (наравне с локальными администраторами) будет иметь полный доступ к объектам DB2 через операционную систему.
- DB2USERS: эта группа будет иметь доступ для чтения и выполнения объектов DB2 через операционную систему.

new in V9.7 В DB2 9.7 члены группы DB2ADMNS будут автоматически получать полномочие SYSADM в DB2, если включена расширенная безопасность и не задан параметр конфигурации базы данных SYSADM_GROUP.

10.8 Краткий обзор

В этой главе рассматривались аспекты безопасности DB2, начиная от всестороннего обсуждения различий между аутентификацией и авторизацией и важности этих процессов. Затем мы изучили различные уровни полномочий, обеспечивающие безопасность экземпляров и баз данных.

После этого мы рассмотрели новое понятие ролей, способы их применения для повышения безопасности, а также ограничения настроек безопасности с помощью групп. В частности, была рассмотрена группа PUBLIC и предложены способы обезопасить её так, чтобы обычные пользователи не имели доступа к серверу данных.

Кроме того, мы изучили операторы GRANT и REVOKE и, наконец, разобрались с тем, как проверить уровни авторизации и привилегий с помощью Центра управления и таблиц системного каталога.

10.9 Практические задания

До сих пор для выполнения всех команд в базе данных вы использовали учетную запись администратора экземпляра (SYSADM). Эта учетная запись имеет широкий доступ ко всем утилитам, данным и объектам базы данных. Поэтому очень важно обеспечить защиту этой учетной записи, чтобы предотвратить случайную или преднамеренную потерю данных. В большинстве случаев рекомендуется создавать другие учетные записи или группы с ограниченным набором привилегий. В этом упражнении мы создадим новую учетную запись пользователя и назначим для неё определенные привилегии.

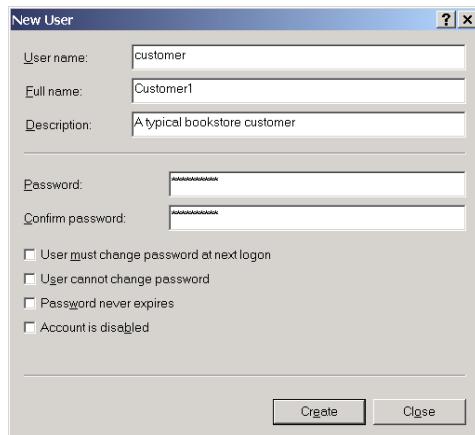
Часть 1. Работа с привилегиями

В этой части упражнения мы поупражняемся в предоставлении и отмене привилегий с помощью Центра управления.

Процедура

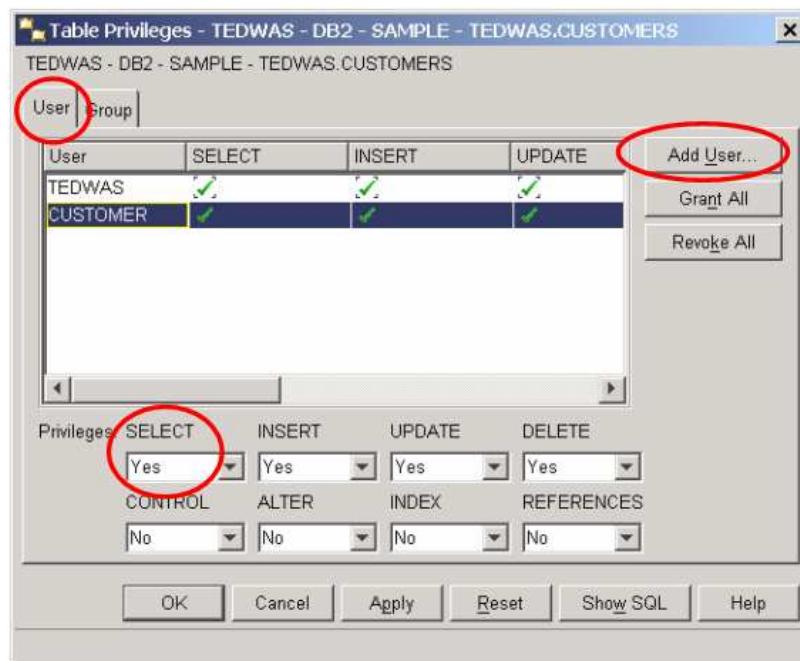
1. Запустите консоль управления компьютером Windows, щелкнув правой кнопкой мыши на пиктограмме *Мой компьютер* на рабочем столе и выбрав в меню пункт *Управление*.

2. Разверните список *Служебные программы* на левой панели дерева элементов, а затем разверните папку *Локальные пользователи и группы*. Щелкните правой кнопкой мыши папку *Пользователи* и выберите элемент *Новый пользователь*.
3. В диалоговом окне *Новый пользователь* укажите следующую информацию: в поле *Пользователь* введите *customer*, а в поле *Полное имя* — *Customer1*. В поле *Описание* введите *A typical bookstore customer*. В полях *Пароль* и *Подтверждение* введите *ibmdb2ibm*. Снимите флажок возле пункта *Потребовать смену пароля при следующем входе в систему* и нажмите кнопку *Создать* для создания нового пользователя, а затем кнопку *Закрыть*, чтобы закрыть диалоговое окно.



4. Откройте Центр управления и выберите расширенный вид. Чтобы переключиться к расширенному виду, выберите пункт меню *Инструменты ->Настроить Центр управления*. Затем выберите вариант *Расширенный* и нажмите кнопку *OK*.
5. Раскройте дерево объектов Центра управления на левой панели, чтобы отобразить *Все базы данных -> EXPRESS -> Таблицы*.
6. Предоставьте созданному пользователю необходимые привилегии. В списке таблиц базы данных *EXPRESS* щелкните правой кнопкой мыши на таблице *CUSTOMERS* и выберите пункт *Привилегии*, чтобы открыть диалоговое окно *Привилегии для таблицы*.
7. Нажмите кнопку *Добавить пользователя* и выберите только что созданного пользователя *customer*. Нажмите кнопку *OK*, чтобы закрыть диалоговое окно *Добавить пользователя*.
8. Вы увидите, что пользователь *customer* был добавлен в список пользователей, но не имеет привилегий. Чтобы предоставить пользователю привилегии *SELECT, INSERT, UPDATE* и *DELETE*, выберите в каждом из раскрывающихся списков пункт *Да*. Пользователям сети Интернет необходимо иметь возможность просматривать/добавлять/обновлять/удалять данные своих учетных записей.

Мы не предоставляем пользователям прочие разрешения, поскольку они им не требуются. Нажмите кнопку **OK**, чтобы закрыть диалоговое окно «Привилегии для таблицы» и принять внесенные изменения.



9. Повторите шаги 7—9 для таблиц *BOOKS* и *SALES*. Для таблицы *BOOKS* предоставьте только привилегию *SELECT*, поскольку клиент не должен иметь возможность менять учётные данные магазина. Для таблицы *SALES* предоставьте только привилегии *SELECT* и *INSERT*. Клиент НЕ может иметь привилегии *DELETE* или *UPDATE*, поскольку только работники магазина должны иметь возможность менять данные торговых операций.
10. Подключитесь к базе данных с помощью созданного выше идентификатора пользователя *customer*, выполнив в командном окне DB2 следующее:

```
db2 connect to express user customer using ibmdb2ibm
```

Попробуйте выбрать (*SELECT*) данные в таблице *customers*. Что происходит? Попробуйте удалить (*DELETE*) или обновить (*UPDATE*) данные таблицы *SALES*. Что происходит?

Часть 2. Работа с полномочиями SYSADM, DBADM и SECADM

В этой части упражнения мы поупражняемся в предоставлении полномочий SYSADM и DBADM и узнаем принцип их работы.

Процедура

1. Выполните шаги, описанные в первой части этого упражнения, чтобы создать нового пользователя *mysysadm*
2. Создайте группу Windows *mysysadmgrp*. Повторите шаги, использовавшиеся при создании пользователя, но щелкните правой кнопкой мыши не на папке *Пользователи*, а на папке *Группы*, и выберите пункт *Создать группу*. В поле «Имя группы» введите *mysysadmgrp*. В разделе «Члены группы» щелкните *Добавить*, чтобы добавить нового члена, и введите *mysysadm*. Нажмите кнопку *Проверить имена*, чтобы проверить правильность ввода имени. Если всё правильно, нажмите OK. Нажмите кнопку *Создать*, а затем *Закрыть*.
3. Таким образом, мы создали одного пользователя *mysysadm* и одну группу *mysysadmgrp*, к которой принадлежит пользователь *mysysadm*. Всё это сделано в операционной системе Windows. Теперь нужно сообщить DB2 о том, что группа *mysysadmgrp* должна быть группой SYSADM. Для этого выполните следующую команду в командном окне DB2:

```
db2 update dbm cfg using SYSADM_GROUP mysysadmgrp
```

Поскольку параметр SYSADM_GROUP не является динамическим, необходимо остановить и снова запустить экземпляра. Выполнение принудительной остановки db2stop будет гарантировать удаление всех подключений до db2stop.

```
db2stop force  
db2start
```

4. Подключитесь к базе данных SAMPLE как пользователь *mysysadm* из командного окна DB2 и запустите запрос SELECT * для таблицы STAFF. Обратите внимание на то, что понадобится правильная схема, которая использовалась при создании этой таблицы. В приведенном ниже примере в качестве схемы используется *arfchong*.

```
db2 connect to sample user mysysadm using ibmdb2ibm  
db2 select * from arfchong.staff
```

Должно появиться следующее сообщение об ошибке. Почему? Вы не имеете полномочия SYSADM?

SQL0551N "MYSYSADM" не имеет необходимой авторизации или привилегии для выполнения операции "SELECT" с объектом "ARFCHONG.STAFF".
SQLSTATE=42501

Начиная с DB2 9.7, SYSADM не получает полномочие DBADM по умолчанию. Именно поэтому и появилось сообщение об ошибке.

5. Подключитесь к базе данных SAMPLE как пользователь Windows, который использовался для создания этой базы данных. В нашем примере это пользователь ARFCHONG. Теперь предоставьте полномочие DBADM без

DATAACCESS пользователю *mysysadm* и попробуйте снова выполнить операцию SELECT в таблице STAFF как *mysysadm*. Операция сработала? Почему?

```
db2 connect to sample user arfchong using ibmdb2ibm
db2 grant dbadm without dataaccess on database to user mysysadm
db2 connect to sample user mysysadm using ibmdb2ibm
db2 select * from arfchong.staff
```

Как видите, появляется такое же сообщение об ошибке, даже после предоставления пользователю *mysysadm* полномочия DBADM. Так и должно быть, поскольку мы добавили выражение WITHOUT DATAACCESS, которое означает, что полномочие DATAACCESS не предоставлялось, а потому пользователь *mysysadm* всё ещё не имеет возможности получить доступ к данным. Это пример того, как можно ограничить доступ к данным для DBADM.

6. Теперь предоставим пользователю *mysysadm* полномочие DATAACCESS и снова попробуем выполнить операцию SELECT.

```
db2 connect to sample user arfchong using ibmdb2ibm
db2 grant DATAACCESS on database to user mysysadm
db2 connect to sample user mysysadm using ibmdb2ibm
db2 select * from arfchong.staff
```

Теперь операция SELECT должна работать правильно. В этом упражнении продемонстрирован новый принцип работы SYSADM и DBADM, впервые представленный в DB2 9.7. Главное, что вы должны понять из этого упражнения, — теперь доступ к данным отделен от возможностей SYSADM и DBADM.

7. Сбросьте значение SYSADM_GROUP до NULL, чтобы группа локальных администраторов и локальная системная учетная запись снова получили полномочие SYSADM:

```
db2 update dbm cfg using sysadm_group NULL
db2stop force
db2start
```


11

Глава 11. Резервное копирование и восстановление

В этой главе рассматриваются запись в журнал базы данных DB2, а также вопросы создания полной или частичной резервной копии базы данных с помощью служебной программы BACKUP и восстановления данных с помощью служебной программы RESTORE.

Примечание.

Чтобы получить более подробную информацию о записи в журнал, резервном копировании и восстановлении, просмотрите видео:

<http://www.channeldb2.com/video/video/show?id=807741:Video:4282>

11.1 Запись в журнал базы данных

Работая в текстовом редакторе, чтобы убедиться в том, что документ сохранен, необходимо нажать кнопку *Сохранить*. В случае баз данных эту функцию выполняет оператор COMMIT. Каждое выполнение оператора COMMIT гарантирует, что все изменения, внесенные в данные, будут где-либо сохранены.

Кроме того, при работе в текстовом редакторе время от времени в нижнем правом углу появляется сообщение «автосохранение». Это происходит и в случае баз данных, поскольку во время выполнения любых операций с данными, например UPDATE, INSERT или DELETE, изменения где-либо сохраняются.

«Где-либо» в предыдущих абзацах обозначает журналы базы данных. Журналы базы данных хранятся на диске и используются для записи действий и транзакций. В случае сбоя системы или базы данных журналы используются при восстановлении для воспроизведения и проведения выполненных транзакций.

На *рис. 11.1* схематически показаны процессы, связанные с записью в журнал базы данных.

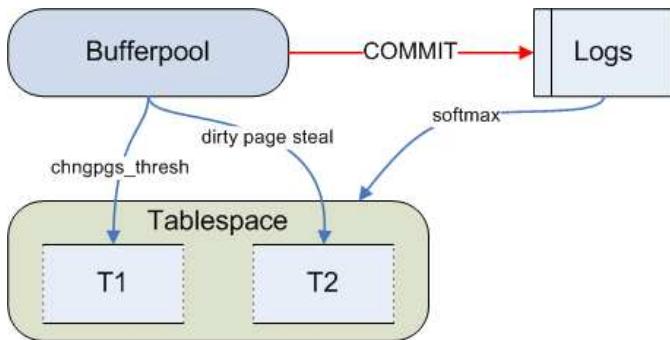


Рисунок 11.1. Запись в журнал базы данных

На рис. 11.1 показано табличное пространство и журналы. Эти элементы хранятся на дисках, хотя рекомендуется использовать для хранения разные диски. При выполнении, к примеру, операции UPDATE, страницы соответствующих строк заносятся в буферный пул (память). Обновления выполняются в буферном пуле, и как старые, так и новые значения сохраняются в файлах журнала — иногда сразу, а иногда после заполнения буфера журнала. Если после операции UPDATE выполнить команду COMMIT, старые и новые значения будут сразу сохранены в файлах журнала. Этот процесс повторяется для многих SQL-операций, выполняемых с базой данных. Страницы буферного пула «формируются» и записываются на диск табличного пространства только при выполнении определенных условий, к примеру, при достижении порога смены страницы, указанного для параметра CHNGPGS_THRES. Параметр CHNGPGS_THRES указывает процент буферного пула с заполненными страницами, т. е. страницами, содержащими изменения.

С точки зрения производительности, нет смысла дважды выполнять запись для каждой операции COMMIT: один раз для записи в журнал, а второй — на диск табличного пространства. Именно поэтому копирование данных из журнала на диск табличного пространства выполняется только при достижении порогового значения таких параметров как CHNGPGS_THRES.

11.2 Типы журналов

Существует два типа журналов:

- **Первичные журналы**

Эти журналы создаются предварительно; количество доступных первичных журналов определяется параметром конфигурирования базы данных LOGPRIMARY.

- **Вторичные журналы**

Эти журналы создаются динамически, по мере потребности DB2. Максимальное количество вторичных журналов задается параметром конфигурирования базы данных LOGSECOND. Динамическое создание журнала является дорогостоящей операцией, поэтому рекомендуется пользоваться первичным журналом. Файлы вторичного журнала удаляются при остановке экземпляра.

Можно использовать бесконечную запись в журнал, установив для параметра LOGSECOND значение -1, однако не рекомендуем этого делать, поскольку таким образом можно быстро израсходовать пространство файловой системы.

11.3 Способы записи в журнал

Существует два способа записи в журнал: циклическая запись в журнал (по умолчанию) и архивирование журналов.

11.3.1 Циклическая запись в журнал

Циклическая запись в журнал используется по умолчанию и включена, если для параметров конфигурации базы данных LOGARCHMETH1 и LOGARCHMETH2 задано значение OFF. Эти параметры обозначают способ архивирования журналов, но если их отключить, журналы архивироваться не будут — именно так работает циклическая запись в журнал. На рис. 11.2 проиллюстрирован принцип циклической записи в журнал.

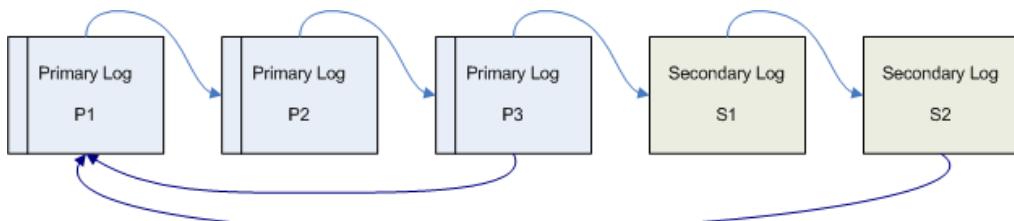


Рисунок 11.2. Работа с первичными и вспомогательными журналами

На рис. 11.2 показано 3 первичных журнала, поэтому можно предположить, что для параметра LOGPRIMARY установлено значение 3. Для простоты в данном примере выполняется только одна транзакция. По мере выполнения транзакции заполняется файл журнала P1, а затем P2. Если вносятся новые данные и позже информация выводится на диск табличного пространства, P1 и P2 можно перезаписать, поскольку содержащаяся в них информация больше не требуется для аварийного восстановления (этот вопрос рассмотрен более подробно в последующих разделах этой главы). С другой стороны, если транзакция длится так долго, что используются файлы P1, P2, P3 и требуется дополнительное пространство в журнале, поскольку транзакция не была завершена или утверждена, динамически назначается вторичный журнал (на данном рисунке — S1). Если транзакция продолжается, новые вторичные журналы назначаются до достижения максимального значения LOGSECOND. Если после этого требуются дополнительные журналы, отображается сообщение об ошибке, указывающее на заполнение журналов, и транзакция откатывается. В качестве альтернативы можно настроить DB2 с помощью параметра конфигурирования BLK_LOG_DSK_FUL так, чтобы продолжать запись журналов каждые пять минут, позволяя некоторым транзакциям останавливаться без откатывания. За это время администратор базы данных может найти новое пространство, и транзакция возобновится.

Циклическая запись в журнал позволяет выполнить аварийное восстановление, но если требуется восстановить данные состоянием на определенный момент времени, ближайшей доступной точкой восстановления будет последнее резервное копирование, созданное с помощью служебной программы BACKUP.

11.3.2 Ведение архивных журналов

При ведении архивных журналов, также известном как способ сохранения журналов, файлы журнала не перезаписываются, а сохраняются в онлайновом или автономном режиме. Онлайновые архивные журналы хранятся с активными журналами, необходимыми для аварийного восстановления. Автономные архивные журналы перемещаются на другие носители, например ленточные, с помощью рутинных операций USEREXIT, Tivoli Storage Manager или других продуктов для архивирования от сторонних разработчиков.

Чтобы включить ведение архивных журналов, установите для параметра конфигурации базы данных LOGARCHMETH1 или LOGARCHMETH2 (или обоих) значение, отличное от OFF. Также можно установить для параметра конфигурирования LOGRETAIN значение RECOVERY. В результате для LOGARCHMETH1 автоматически будет установлено значение LOGRETAIN. Однако параметр LOGRETAIN считается устаревшим и оставлен, прежде всего, для совместимости с более старыми версиями DB2.

Обычно ведение архивных журналов применяется в производственных системах; поскольку журналы сохраняются, появляется возможность восстановить базу данных состоянием на любой момент времени от создания первого файла журнала. Используя ведение архивных журналов, администратор баз данных может выполнить восстановление при ошибках, допущенных людьми. К примеру, если пользователь системы случайно запустит транзакцию, длящуюся несколько дней, при обнаружении проблемы администратор баз данных сможет восстановить систему до состояния перед запуском такой транзакции. Однако для правильного повторного выполнения транзакции могут потребоваться несколько ручных операций.

Ведение архивных журналов необходимо для поэтапного восстановления отмененных изменений и выполнения резервного копирования в оперативном режиме. На рис. 11.3 показан процесс ведения архивных журналов.

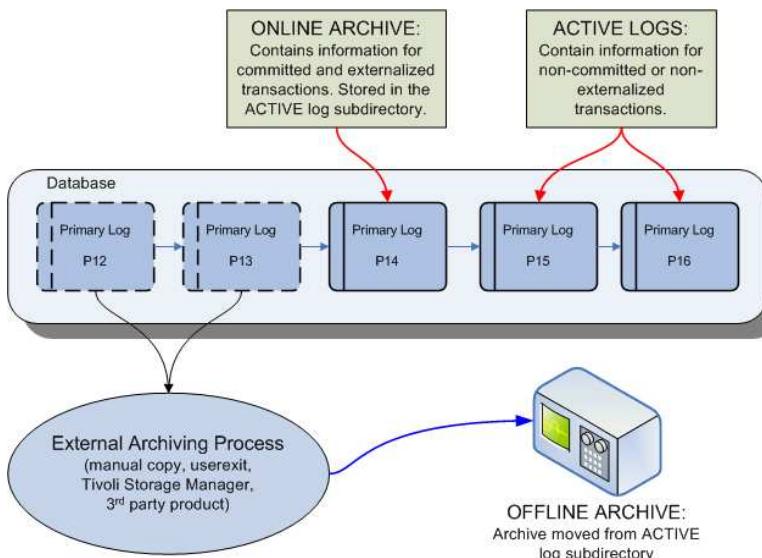


Рисунок 11.3. Ведение архивных журналов

11.4 Запись в журнал базы данных из Центра управления

Запись в журнал базы данных можно настроить в Центре управления, щелкнув правой кнопкой мыши на соответствующей базе данных и выбрав пункт **Конфигурировать запись в журнал базы данных**. Это изображено на рис. 11.4.

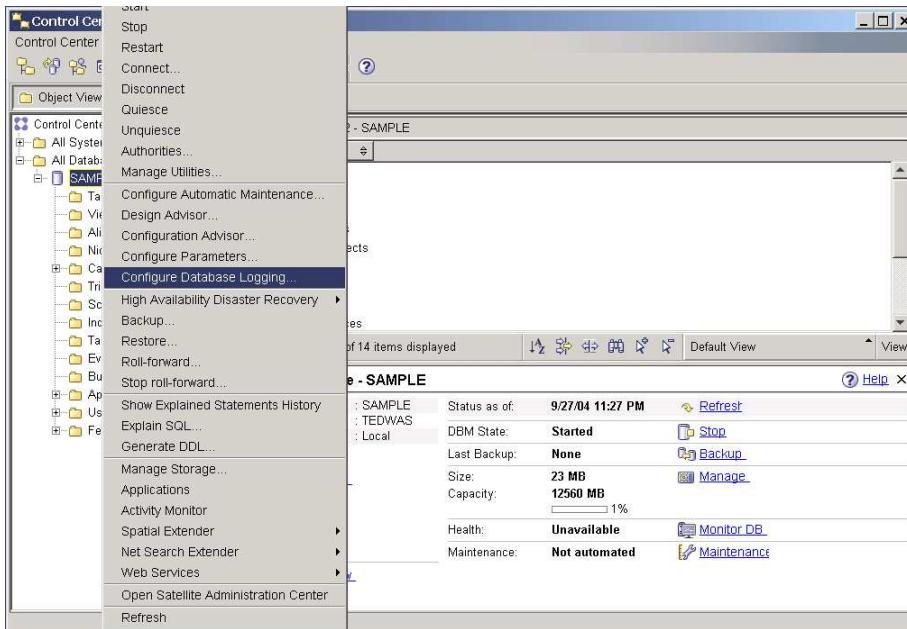


Рисунок 11.4. Конфигурирование записи в журнал базы данных из Центра управления

На рис. 11.5 показан мастер по конфигурированию записи базы данных в журнал, в котором можно выбрать циклическую запись либо архивирование журналов.

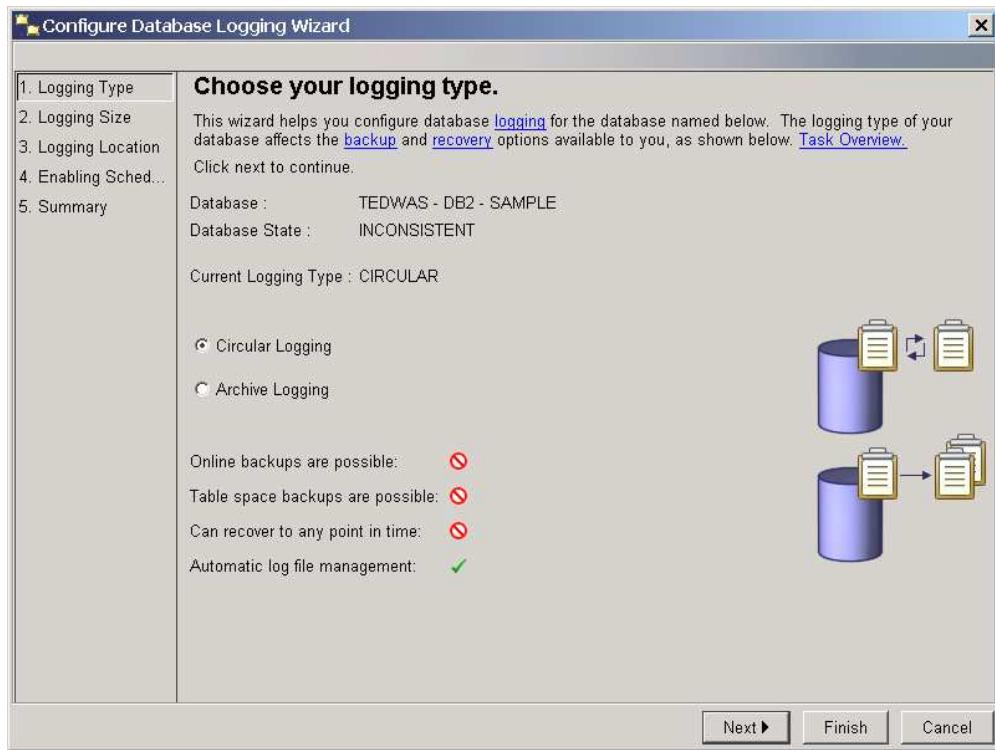


Рисунок 11.5. Мастер по конфигурированию записи базы данных в журнал

11.5 Параметры записи в журнал

С записью в журнал связан ряд параметров DB CFG. В Таблице 11.1 перечислены основные из этих параметров.

| Параметр | Описание |
|---------------|--|
| logbufsz | Объем памяти, используемый в качестве буфера для записей журнала до их сохранения на диск |
| logfilsz | Размер каждого настроенного журнала, число страниц по 4 КБ |
| logprimary | Количество первичных журналов размера logfilsz, которое будет создано |
| logsecond | Количество вспомогательных журналов, при потребности создаваемых и используемых для восстановления |
| newlogpath | Активные и онлайновые архивные журналы базы данных изначально создаются в каталоге вашей базы данных, в подкаталоге SQLOGDIR. Это размещение можно изменить, отредактировав значение этого параметра конфигурирования так, чтобы он указывал на другой каталог или другое устройство |
| mirrorlogpath | Чтобы защитить первичные журналы от сбоев или случайного удаления диска, можно задать ведение идентичного набора |

| | |
|------------------------------|---|
| | журналов на вспомогательном (зеркальном) пути к журналам. Иными словами, две копии файлов первичного журнала будут храниться в двух разных местах. |
| logarchmeth1 logarchmeth2 | Указывает размещение для хранения архивных файлов журналов, отличное от пути к активному журналу. Если задать оба параметра, каждый файл журнала будет архивироваться дважды. Иными словами, две копии архивных файлов журнала будут храниться в двух разных местах. Доступные значения: OFF (т. е. включена циклическая запись в журнал), LOGRETAIN, USEREXIT, DISK, TSM, VENDOR |
| loghead | Имя активного файла журнала |
| softmax | Ограничение стоимости аварийного восстановления |
| overflowlogpath | Указание размещения, в котором DB2 сможет найти файлы журнала, необходимые для поэтапного восстановления отмененных изменений. Действие аналогично опции OVERFLOW LOG PATH команды ROLLFORWARD |
| blk_log_dsk_ful | Устанавливается для предотвращения ошибок, связанных с заполнением диска и появляющихся, если DB2 не может создать новый файл журнала на пути к активному журналу. Вместо этого DB2 будет пытаться создать файл журнала каждые пять минут до успешного выполнения этой операции. Незаблокированный SQL, доступный только для чтения, может продолжать работу |
| max_log | Максимальный процент пространства, выделяемого под активный журнал для одной транзакции |
| num_log_span | Количество файлов активного журнала для одной активной элементарной операции |
| mincommit | Число объединений в группу до записи на диск |

Таблица 11.1. Параметры записи в журнал

11.6 Резервное копирование базы данных

Команда резервного копирования DB2 дает возможность создать мгновенную копию базы данных в момент выполнения команды. Ниже представлена самая простая синтаксическая структура для выполнения этой команды:

```
BACKUP DATABASE <имя_бд> [ TO <путь> ]
```

Большинство команд и утилит могут выполняться как в онлайновом, так и в автономном режиме. Онлайновый режим подразумевает, что во время выполнения вами команды другие пользователи могут подключаться к базе данных и выполнять свои операции. Автономный режим означает, что во время выполнения вами операции к базе данных не подключены другие пользователи. Чтобы разрешить выполнение операции в онлайновом режиме, добавьте в синтаксис команды ключевое слово ONLINE; в противном случае по умолчанию команда будет выполняться в автономном режиме.

К примеру, если необходимо создать резервную копию базы данных **SAMPLE** на пути C:\BACKUPS, можно выполнить следующую команду в командном окне DB2 или командном процессоре Linux:

```
db2 BACKUP DB sample TO C:\BACKUPS
```

Обратите внимание, что каталог C:\BACKUPS должен существовать до выполнения этой команды. Также убедитесь в отсутствии подключений к базе данных при выполнении вышеописанной команды, иначе появится сообщение об ошибке, поскольку при наличии подключений невозможно выполнить резервное копирование в автономном режиме.

Чтобы проверить наличие подключений к базам данных в экземпляре, выполните следующую команду в командном окне DB2 или командном процессоре Linux:

```
db2 list applications
```

Чтобы принудительно закрыть подключения всех баз данных в экземпляре, выполните следующую команду в командном окне DB2 или командном процессоре Linux:

```
db2 force applications all
```

Не советуем использовать последнюю команду в производственной среде с большим числом пользователей — ваши коллеги могут остаться недовольными! Также обратите внимание, что последняя команда работает асинхронно. Это значит, что при попытке сразу же запустить команду резервного копирования она может всё ещё не работать. Подождите несколько секунд и повторите команду резервного копирования, если при первой попытке появилось сообщение об ошибке.

После успешного выполнения команды резервного копирования создается новый файл, содержащий образ резервной копии базы данных. Имя этого файла задается в соответствии с условными обозначениями, показанными на *рис. 11.6*.

Linux/UNIX/Windows

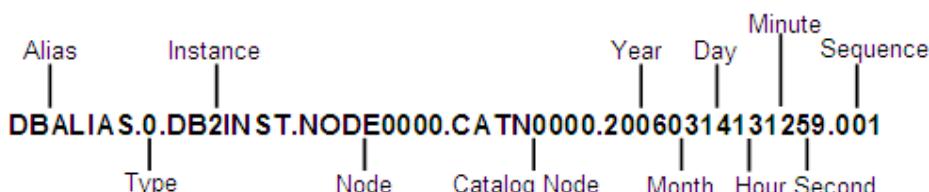


Рисунок 11.6. Условные обозначения в имени образа резервной копии

Тип «0» означает, что резервная копия является полной. Тип «3» означает, что это резервная копия табличного пространства. В неразделенных базах данных для узла установлено значение NODE0000; это относится ко всем редакциям DB2, за исключением редакции DB2 Enterprise с функцией DPF. Для узла каталога также устанавливается значение CATN0000. С более подробной информацией можно ознакомиться в руководствах по DB2.

Если по одному пути создано и сохранено несколько резервных копий, для различия образов резервных копий в конце имени файла проставляется метка времени. Команда RESTORE может использовать такие метки времени для восстановления из определенной резервной копии. Мы рассмотрим это подробнее в следующем разделе.

11.7 Восстановление базы данных

Восстановление базы данных подразумевает возврат исходной базы данных из резервной копии и/или журналов. При восстановлении из резервной копии база данных воссоздается в том виде, какой она имела на момент резервного копирования.

Если до создания резервной копии было включено архивирование журналов, можно выполнить восстановление не только из образа резервной копии, но и из журналов. Как описано в следующем разделе, поэтапное восстановление отмененных изменений дает возможность выполнить восстановление из резервной копии, а затем (поэтапно) применить журналы до самого последнего из них или же до определенного момента времени.

Обратите внимание, что хотя для обозначения восстановления часто употребляется английский термин «recovery», для восстановления используется команда RESTORE.

11.7.1 Типы восстановления

Существует три типа восстановления:

- **Восстановление после сбоя или перезагрузки**

Предположим, вы работаете на настольном ПК и выполняете важные транзакции в базе данных DB2. Внезапно пропало электричество, либо кто-нибудь случайно выдернул шнур питания компьютера из розетки — что случится с базой данных?

При следующем запуске компьютера и программы DB2 будет автоматически выполнено восстановление после сбоя. Для восстановления после сбоя DB2 автоматически выполнит команду RESTART DATABASE, считает и повторит/отменит транзакции на основании записей в активном журнале. После завершения команды база данных гарантированно будет согласованной, т. е. зафиксированные результаты операций будут сохранены, а для незафиксированных операций будет выполнен откат.

- **Восстановление версии или образа**

Этот тип восстановления подразумевает выполнение восстановления только из резервной копии; соответственно, база данных будет восстановлена до состояния на момент резервного копирования. Все транзакции, выполнявшиеся в базе данных после резервного копирования, будут утрачены.

- **Поэтапное восстановление отмененных изменений**

Используя этот тип восстановления, вы не просто выполняете восстановление (RESTORE) из образа резервной копии, а и запускаете команду ROLLFORWARD, чтобы применить журналы поверх резервной копии и восстановить состояние на

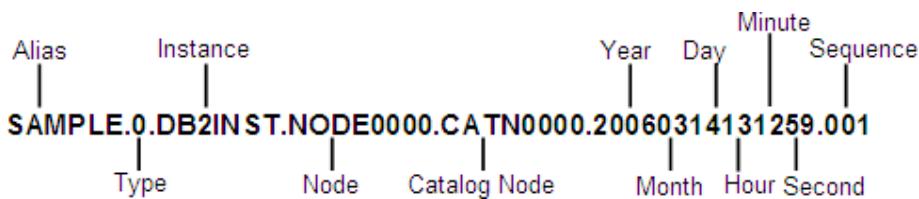
определенный момент времени. При использовании этого типа восстановления потери данных будут минимальными.

11.7.2 Восстановление базы данных

Выполните команду RESTORE, чтобы восстановить базу данных из образа резервной копии. Ниже представлена наиболее простая синтаксическая структура этой команды:

```
RESTORE DATABASE <имя_бд> [from <путь>] [taken at <метка_времени>]
```

К примеру, имеется файл образа резервной копии базы данных *sample* с таким именем:



Можно выполнить следующее:

```
RESTORE DB sample FROM <путь> TAKEN AT 20060314131259
```

11.8 Прочие операции с BACKUP и RESTORE

Ниже перечислены еще несколько возможностей команд BACKUP и RESTORE. Рекомендуем ознакомиться с руководствами по DB2, чтобы узнать об этих возможностях более подробно.

- Резервное копирование базы данных 32-разрядного экземпляра и её восстановление в 64-разрядном экземпляре.
- Восстановление поверх существующей базы данных.
- Перенаправленное восстановление при восстановлении в системе с количеством дисков, отличным от указанного в образе резервной копии.
- Резервное копирование или восстановление только табличного пространства, а не целой базы данных.
- Разностное и пошаговое резервное копирование; при разностном резервном копировании записываются только изменения, внесенные с момента создания последней резервной копии, а при пошаговом резервном копировании все изменения записываются и аккумулируются на каждом образе резервной копии.
- Резервное копирование моментальной копии (flash copy) — требуется соответствующее аппаратное обеспечение.
- Восстановление удаленных таблиц (если эта опция была включена для соответствующей таблицы).

-
- Невозможно выполнить резервное копирование на одной платформе (например, Windows) и восстановление на другой (например, Linux). Для этого сценария воспользуйтесь командами db2look и db2move. При использовании редакций DB2 с поддержкой ОС UNIX следует учитывать, что в ОС UNIX некоторые платформы поддерживают резервное копирование и восстановление между разными платформами UNIX.

11.9 Краткий обзор

В этой главе мы рассмотрели функцию записи в журнал в DB2, в том числе два типа журналов (первичные и вспомогательные) и два типа записи в журнал (циклическую запись и архивирование журналов), а также разнообразные параметры базы данных, относящиеся к записи в журнал. Мы обсудили, когда и зачем применяется каждый из типов записи в журнал, а также научились настраивать типы из Центра управления.

Мы также узнали, как с помощью командной строки DB2 выполнить резервное копирование и восстановление, и подробно ознакомились с тремя типами восстановления баз данных: аварийным восстановлением, восстановлением версии и поэтапным восстановлением отмененных изменений.

11.10 Упражнения

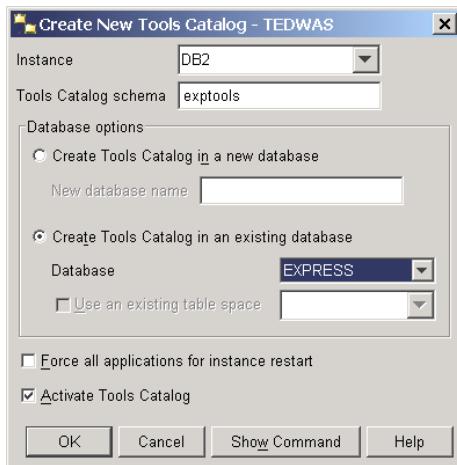
Хотя в DB2 можно автоматизировать некоторые операции обслуживания баз данных, иногда необходимо настроить время выполнения определенных операций. В этом упражнении мы создадим пользовательское расписание резервного копирования в ночное время для базы данных **EXPRESS**.

Процедура

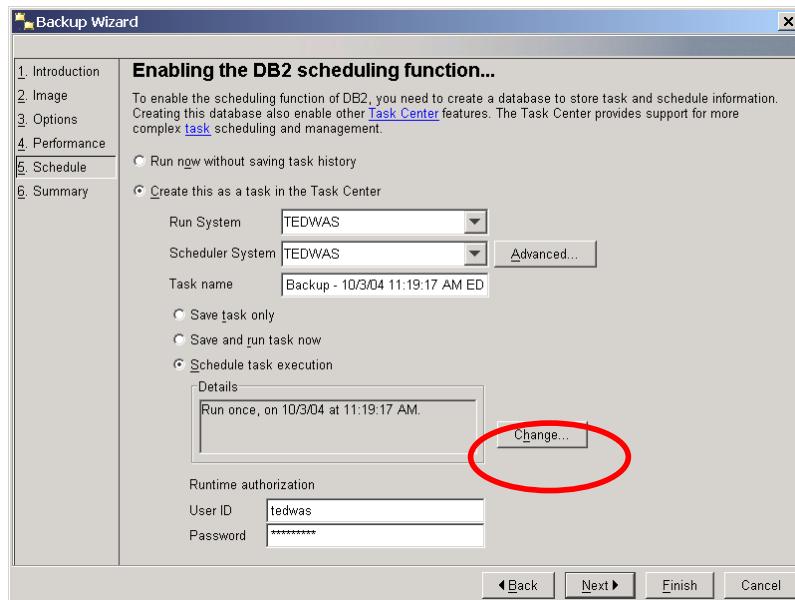
1. На панели дерева объектов Центра управления откройте *Центр управления* -> *Все базы данных*. Щелкните правой кнопкой мыши базу данных **EXPRESS** и выберите пункт *Резервное копирование*. Откроется *Мастер по резервному копированию*.
2. На странице *Введение* подытожено текущее состояние базы данных, а также указано время последнего резервного копирования и метод записи в журнал. Нажмите кнопку *Далее*, чтобы перейти к следующей странице мастера.
3. На странице *Образ* выберите место хранения образа резервной копии. Рекомендуется выбирать не тот физический диск, на котором хранится существующая база данных. На данном этапе создайте в файловой системе новую папку *C:\db2backup* и укажите её как место хранения резервной копии. В окне мастера выберите элемент *Файловая система* из раскрывающегося списка *Тип носителя*. Нажмите кнопку *Добавить*, выберите папку, которую только что создали, и нажмите *OK*. Нажмите кнопку *Далее*, чтобы перейти к следующей странице мастера.
4. Можете ознакомиться со страницами *Опции* и *Производительность*, однако обычно установленных по умолчанию параметров вполне достаточно, поскольку DB2 автоматически выполняет резервное копирование базы

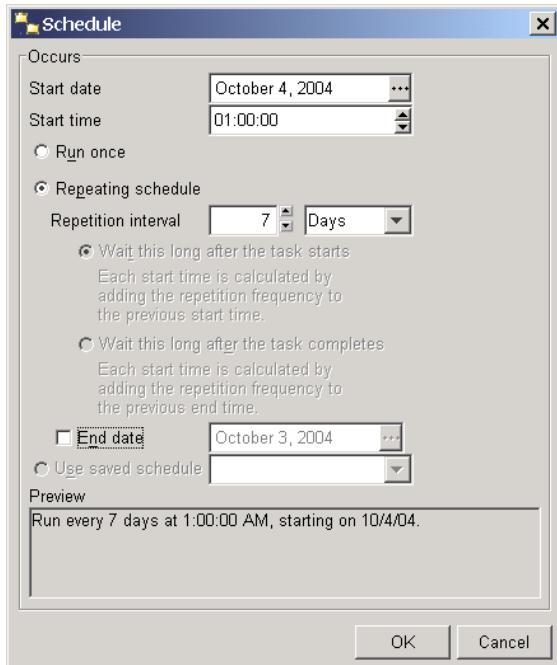
данных наиболее оптимальным способом. После изучения перейдите на страницу *Расписание*.

- На странице *Расписание* включите планировщик, если он еще не включен. Выберите систему, в которой будет создан каталог инструментов, и создайте новый каталог инструментов. Укажите схему каталога инструментов и выберите её создание в существующей базе данных **EXPRESS**. В каталоге инструментов хранятся метаданные обо всех запланированных задачах. Нажмите кнопку **OK**, чтобы продолжить. После создания каталога инструментов нажмите кнопку **Далее**, чтобы перейти к следующей странице мастера.



- На странице *Расписание* выберите создание расписания для выполнения задачи. Назначьте резервное копирование на каждый день, начиная в час ночи. Нажмите кнопку **Далее**, чтобы перейти к следующей странице.





7. На странице *Сводка* можно посмотреть, какие назначенные задачи будут созданы. Просмотрев информацию о предстоящих изменениях, нажмите кнопку *Готово*, чтобы создать задачу.
8. Запустите Центр задач, чтобы просмотреть или изменить созданную только что задачу резервного копирования.

12

Глава 12. Задачи обслуживания

В этой главе рассматриваются некоторые задачи, необходимые для обслуживания базы данных. Общим направлением в DB2 является автоматизация большинства таких задач. Редакция DB2 Express-C, как и прочие текущие редакции DB2, включает такие автоматизированные возможности. Возможность автоматизированного обслуживания является значительным преимуществом для малых и средних компаний, которые не могут нанять администратора баз данных на полную ставку для обслуживания сервера данных. С другой стороны, если в штате работает администратор базы данных, у него (неё) будет больше времени на выполнение сложных операций, что повысит итоговые показатели компании.

Примечание.

Чтобы получить более подробную информацию о задачах обслуживания, просмотрите видео: <http://www.channeldb2.com/video/video/show?id=807741:Video:4302>

12.1 REORG, RUNSTATS, REBIND

В DB2 есть три основные задачи обслуживания (см. *рис. 12.1*): REORG, RUNSTATS и REBIND.

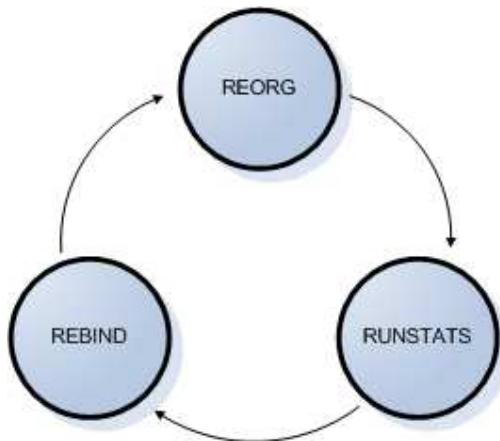


Рисунок 12.1. Задачи обслуживания: REORG, RUNSTATS, REBIND

На *рис. 12.1* показано, что задачи обслуживания выполняются циклическим образом. При выполнении задачи REORG рекомендуется также выполнить задачу RUNSTATS,

а затем REBIND. Через некоторое время таблицы базы данных будут изменены в связи с выполнением операций UPDATE, DELETE и INSERT. Тогда цикл снова начнется с задачи REORG.

12.1.1 Команда REORG

Со временем в результате выполнения в базе данных операций INSERT, UPDATE и DELETE данные становятся все более фрагментированными на страницах базы данных. Команда REORG восстанавливает нерационально использованное пространство и выполняет реорганизацию данных для их более эффективного поиска. Польза от выполнения команды REORG будет наиболее ощутима в часто изменяемых таблицах. Команду REORG можно выполнить не только для таблиц, но и для индексов, как в онлайновом, так и в автономном режиме.

В автономном режиме команда REORG выполняется быстрее и более эффективно, но при этом нет доступа к таблице. С другой стороны, в онлайновом режиме команда REORG предоставляет доступ к таблице, но может потреблять большее количество системных ресурсов — эта опция лучше всего подходит для небольших таблиц.

Синтаксис:

```
REORG TABLE <имя_таблицы>
```

Пример:

```
REORG TABLE employee
```

Перед командой REORG можно выполнить команду REORGCHK, чтобы определить, нужно ли исправлять таблицу или индекс.

12.1.2 Команда RUNSTATS

Оптимизатор является «мозгом» DB2. Он находит наиболее эффективные пути доступа для поиска и извлечения данных. Оптимизатор заботится о ресурсах системы и использует статистические сведения об объектах базы данных, которые хранятся в таблицах каталога, чтобы максимально повысить производительность базы данных. К примеру, таблицы каталога содержат статистические сведения о количестве столбцов и строк в таблице, количестве и типах доступных для таблицы индексов и пр.

Статистическая информация не обновляется в динамическом режиме. Это сделано преднамеренно, чтобы DB2 не приходилось постоянно обновлять статистику для каждой операции, выполненной с базой данных, что негативно отразилось бы на производительности всей базы данных. Вместо этого DB2 предоставляет команду RUNSTATS для обновления такой статистики. Важно поддерживать статистику базы данных в актуальном состоянии. Оптимизатор DB2 может кардинально изменить пути доступа, если одной строке таблицы противопоставлен миллион строк. Если статистика базы данных актуальна, DB2 может выбрать более эффективный план

доступа. Частота сбора статистических данных должна определяться частотой изменения данных в таблице.

Синтаксис:

```
RUNSTATS ON TABLE <схема.имя_таблицы>
```

Пример:

```
RUNSTATS ON TABLE myschema.employee
```

12.1.3 BIND/REBIND

После успешного выполнения команды RUNSTATS последние статистические данные будут использоваться не во всех запросах. Планы доступа статического SQL определяются при первом выполнении команды BIND, и использованная на тот момент статистика может отличаться от текущей. Рис. 12.2 иллюстрирует этот принцип.

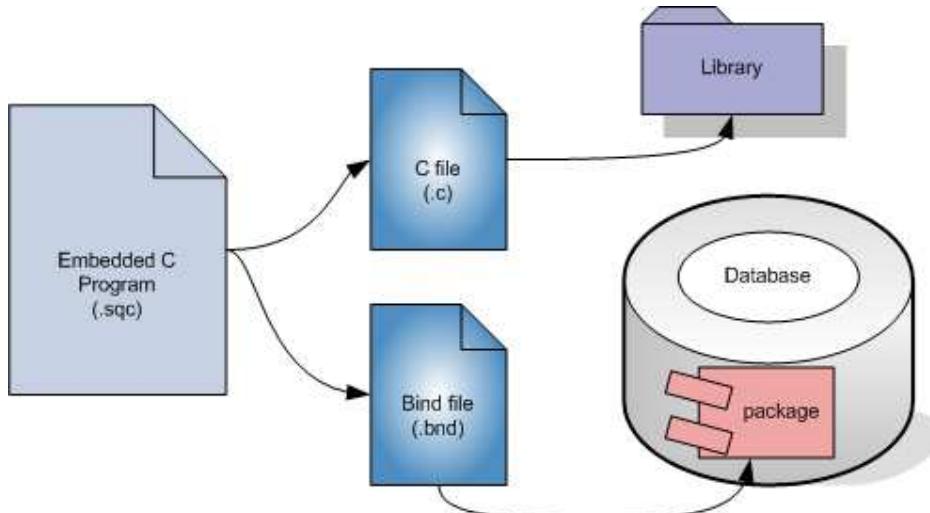


Рисунок 12.2. Процесс связывания статического SQL

На рис. 12.2 предварительно компилируется встроенная программа на С (хранившаяся как файл с расширением «.sqc»). После предварительной компиляции генерируются два файла — файл «.c», содержащий программный код на С, в котором все SQL-элементы выведены в комментарии, и файл «.bnd», содержащий все SQL-операторы. Файл С с расширением «.c» компилируется как обычно с помощью компилятора С, создавая «библиотеку», как показано в верхней правой части рисунка. Файл «.bnd» связан таким же образом, генерируя пакет, хранящийся в базе данных. Связывание эквивалентно компиляции операторов SQL, при которой наилучший план доступа определяется на основе доступной в определенный момент статистики, а затем операторы сохраняются в пакете.

Теперь рассмотрим ситуацию, когда миллион строк вставляют в таблицу, используемую в SQL для такой встроенной программы на С. Если выполнить

команду RUNSTATS после такой вставки, статистика будет обновлена; однако пакет не будет автоматически обновляться для переопределения пути доступа на основе последних статистических данных. Чтобы повторно связать все существующие пакеты с учетом последних статистических данных, можно выполнить команду db2rbind.

Синтаксис:

```
db2rbind database_alias -l <файл_для сообщений>
```

Пример:

Чтобы повторно связать все пакеты базы данных SAMPLE и сохранить сообщения результатов в файле mylog.txt, выполните следующую команду:

```
db2rbind sample -l mylog.txt
```

12.1.4 Задачи обслуживания в Центре управления

В Центре управления можно выполнить команды REORG и RUNSTATS. На рис. 12.3 показано, как это сделать.

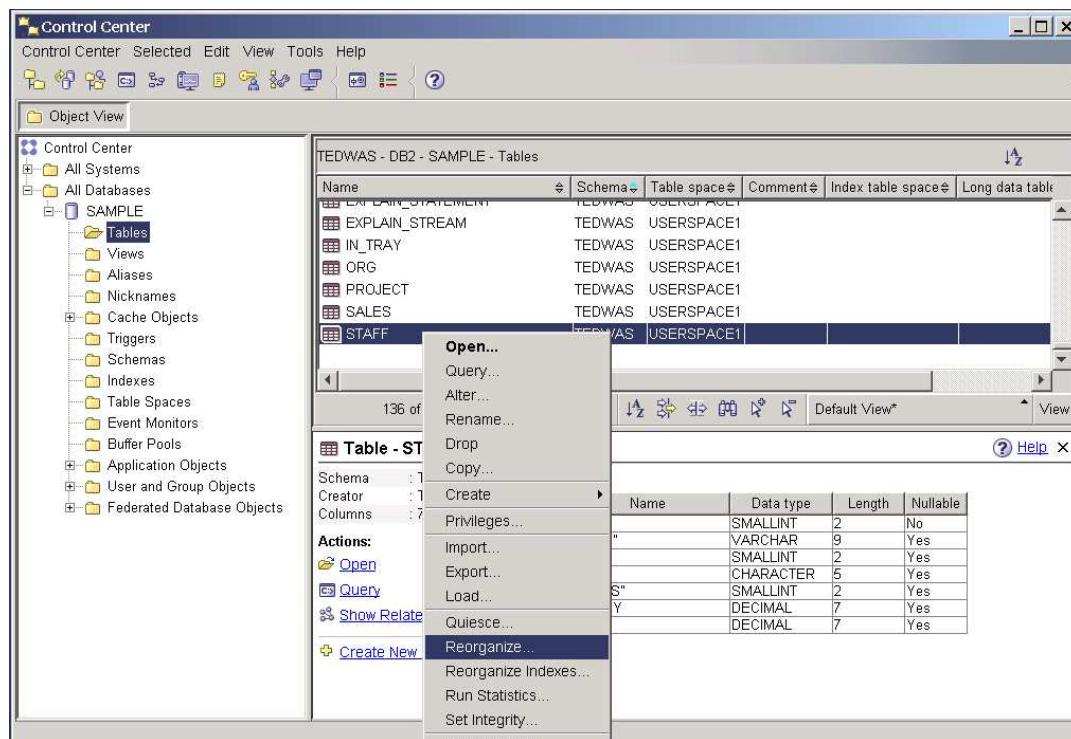


Рисунок 12.3. Выполнение команд REORG и RUNSTATS в Центре управления

Необходимо выбрать нужную таблицу, щелкнуть на нее правой кнопкой мыши и выбрать пункт «Реорганизовать» (для команды REORG) или «Запустить статистику» (для команды RUNSTATS).

12.1.4.1 Операционный вид базы данных

При выборе базы данных в нижней правой части Центра управления в области операционного вида отображается такая информация об этой базе данных, как её размер, дата последнего резервного копирования, состояние автоматического обслуживания и пр. Этот вид дает возможность оперативно определить, в каком обслуживании нуждается база данных. Эта информация показана на *рис. 12.4*.

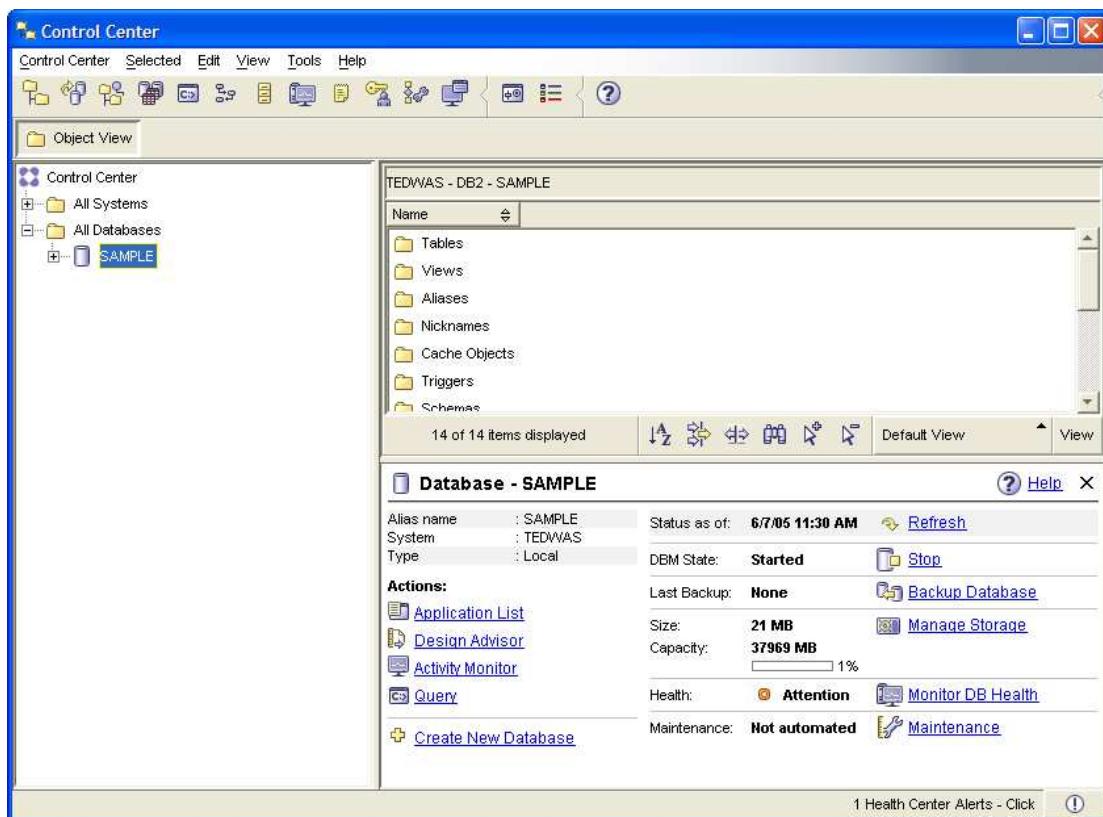


Рисунок 12.4. Операционный вид базы данных в Центре управления

12.2 Варианты обслуживания

Есть три способа выполнения задач обслуживания:

- Обслуживание вручную.

Техническое обслуживание выполняется вручную по мере необходимости.

- Создание сценариев обслуживания.

Можно создать сценарии, содержащие команды обслуживания, и назначить их регулярное выполнение.

- Автоматическое обслуживание

DB2 автоматически выполняет задачи обслуживания (REORG, RUNSTATS, BACKUP).

Основное внимание в этом разделе удалено автоматическому обслуживанию.

Автоматическое обслуживание состоит из таких этапов:

- Пользователь определяет окно обслуживания, во время которого задачи могут выполняться с наименьшими перебоями. К примеру, если система менее всего активна по воскресеньям с 2:00 до 4:00 утра, этот период времени можно использовать как окно обслуживания.
- Определяется два окна обслуживания: один для оперативного обслуживания в оперативном режиме, второй — в автономном режиме.
- При необходимости DB2 будет автоматически выполнять задачи обслуживания во время окон обслуживания.

В Центре управления можно запустить *Мастер конфигурирования автоматического обслуживания*, как показано на рис. 12.5.

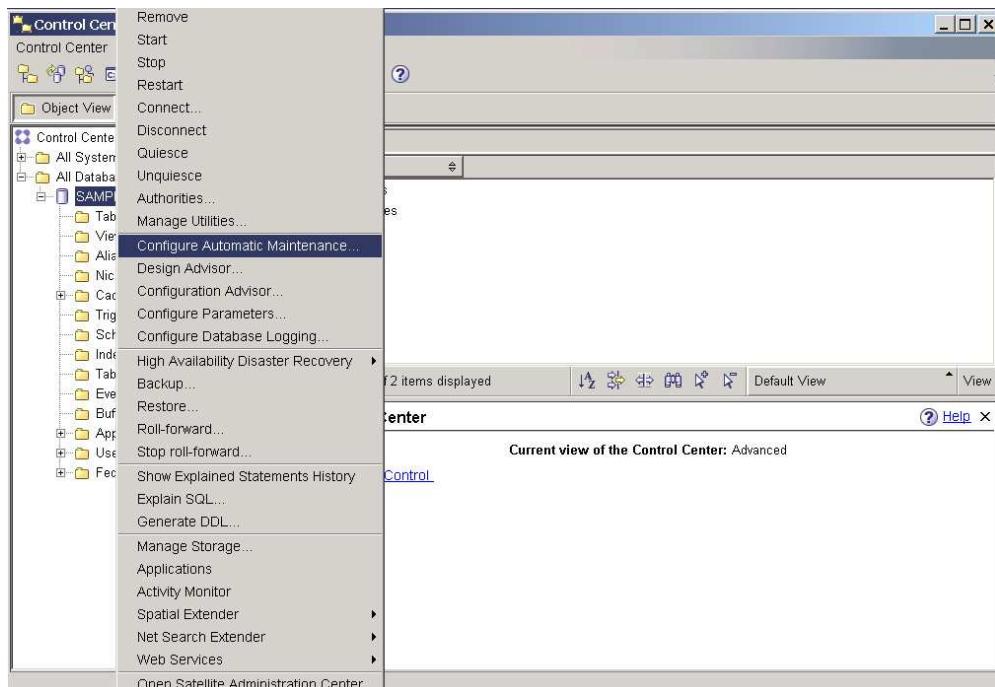


Рисунок 12.5. Запуск мастера конфигурирования автоматического обслуживания

На рис. 12.6 показан Мастер конфигурирования автоматического обслуживания.

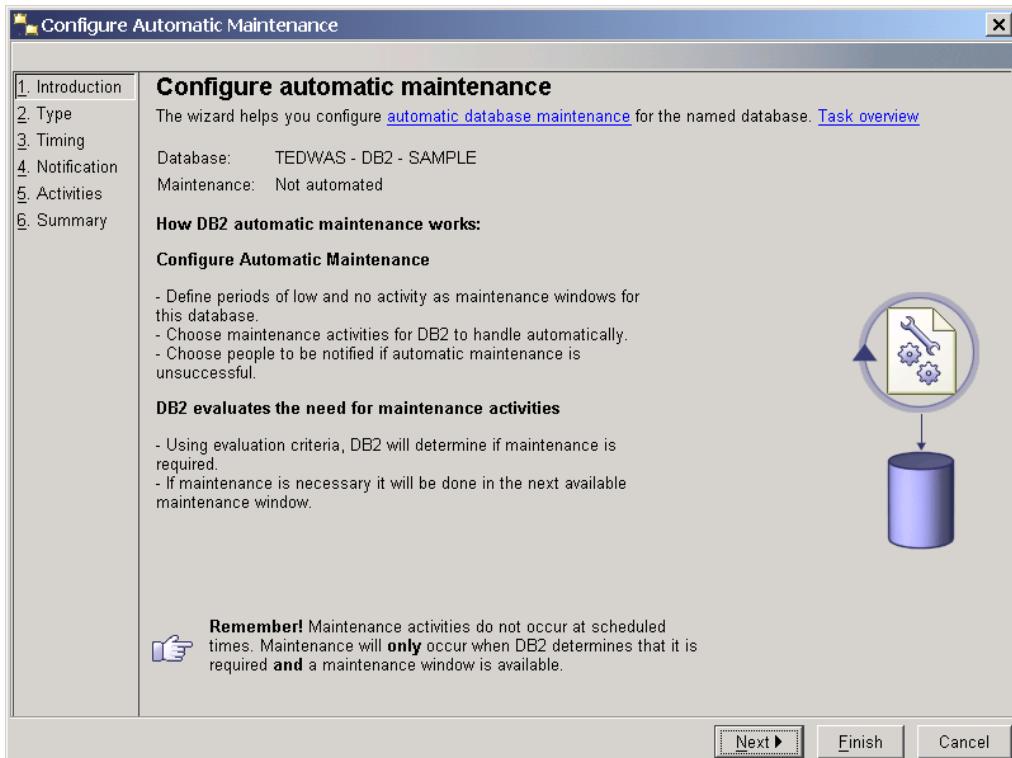


Рисунок 12.6. Мастер конфигурирования автоматического обслуживания

12.3 Краткий обзор

В этой главе рассмотрена важность технического обслуживания базы данных, в частности роль циклов REORG, RUNSTATS и REBIND. Команда REORG, как отображено в её названии, проводит реорганизацию данных для устранения фрагментации и выполнения быстрого вывода данных. RUNSTATS обновляет статистическую информацию, используемую инструментом оптимизации DB2 для повышения производительности данных. Процесс BIND или REBIND обновляет пакеты базы данных с последними путями доступа.

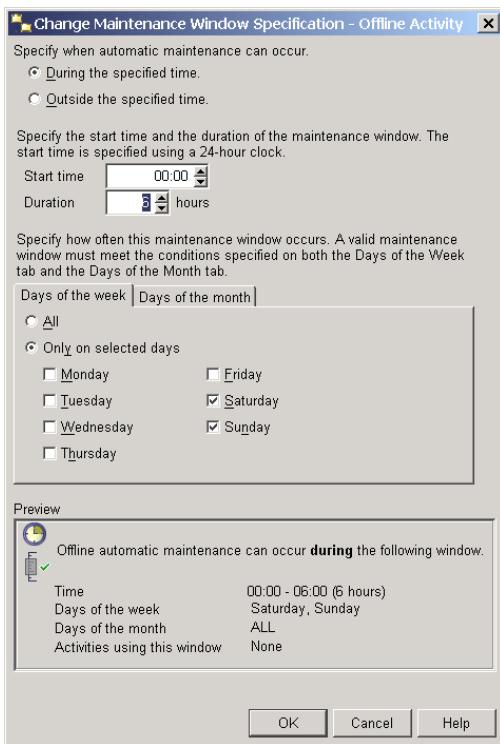
Мы также рассмотрели графические инструменты, представленные в Центре управления DB2 для выполнения обслуживания вручную, по сценарию и автоматически.

12.4 Упражнения

В этом упражнении мы настроим автоматическое обслуживание базы данных SAMPLE в DB2.

Процедура

- На панели дерева объектов в Центре управления щелкните правой кнопкой мыши на базе данных **SAMPLE** и выберите пункт **Конфигурировать автоматическое обслуживание**. Откроется *Мастер конфигурирования автоматического обслуживания*.
- На странице мастера *Введение* отображены текущие настройки автоматического обслуживания. Если база данных создавалась с опцией автоматического обслуживания, автоматическое обслуживание уже сконфигурировано. С помощью этого мастера можно изменить параметры конфигурации автоматического обслуживания. Нажмите кнопку *Далее*, чтобы перейти к следующей странице мастера.
- На странице мастера *Тип* необходимо выбрать либо отключение автоматизации, либо изменение параметров автоматизации. Выберите вариант «Изменить параметры автоматизации». Нажмите *Далее*.
- На странице мастера *Временные характеристики* необходимо указать окна обслуживания. Настройте базу данных так, чтобы она работала в автономном режиме по субботам и воскресеньям с полуночи до 6 часов утра, как показано ниже. Нажмите кнопку *Изменить* возле области предварительного просмотра информации об окне обслуживания в автономном режиме и выберите требуемое время. Указав необходимую информацию, нажмите кнопку *OK*, чтобы вернуться в окно мастера. Оставьте окно обслуживания в оперативном режиме без изменений (обслуживание в оперативном режиме может выполняться в любое время). Нажмите кнопку *Далее*.



5. На странице мастера *Уведомление* можно указать контактную информацию на случай сбоя автоматического обслуживания. Пока пропустим этот шаг. Нажмите кнопку *Далее*.
6. На странице *Операции* мастера можно по отдельности определить, будут ли автоматизированы определенные операции, а также настроить уведомления об определенных операциях. Для этого упражнения убедитесь, что установлены все флагки блока *Автоматизировать* и сняты все флагки блока *Уведомлять*. Нажмите кнопку *Далее*.
7. Прежде чем переходить к следующей странице мастера, настройте расположение резервной копии базы данных. Рекомендуется хранить резервные копии на другом физическом диске, на случай выхода основного диска из строя. На странице *Операции* выберите пункт *Резервное копирование базы данных* и нажмите кнопку *Конфигурировать параметры*.
8. На вкладке «Критерий резервного копирования» диалогового окна «Конфигурировать параметры» выберите вариант *Сбалансировать восстановимость базы данных с производительностью*. На вкладке *Положение резервной копии* выберите существующее расположение резервной копии и нажмите кнопку *Изменить*. Укажите другое расположение для выполнения резервного копирования (убедитесь в том, что на выбранном накопителе достаточно свободного пространства). На вкладке *Режим резервного копирования* убедитесь, что выбран пункт *Резервное копирование в автономном режиме*. Нажмите кнопку *OK*, чтобы закрыть вкладку *Критерий резервного копирования*. Нажмите кнопку *Далее*.
9. На странице *Сводка* мастера *Конфигурирования автоматического обслуживания* содержится итоговая информация о выбранных параметрах. Нажмите кнопку *Готово*, чтобы принять и применить внесенные изменения.

13

Глава 13. Параллельное использование и блокировка

Эта глава рассматривает вопросы о том, как открыть доступ к одной базе данных одновременно нескольким пользователям, избегая взаимного влияния и поддерживая согласованность их операций. Мы рассмотрим принципы транзакций, параллельного использования и блокировки.

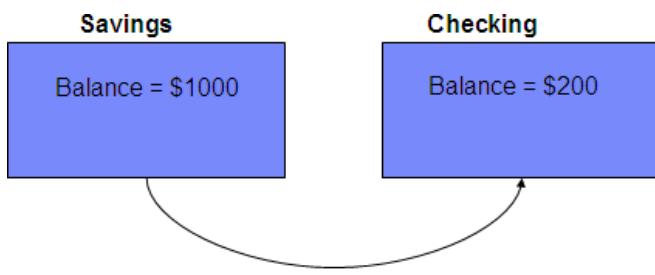
Примечание.

Чтобы получить более подробную информацию о параллельном использовании и блокировке, просмотрите видео:

<http://www.channeldb2.com/video/video/show?id=807741:Video:4322>

13.1 Транзакции

Транзакция (или единица работы) состоит из одного или нескольких операторов SQL, которые при выполнении рассматриваются как отдельная единица; иными словами, сбой одного оператора транзакции приводит к сбою целой транзакции, при этом все операторы, выполненные до момента сбоя, откатываются. Транзакция заканчивается оператором COMMIT, который также обозначает начало новой транзакции. На рис. 13.1 приведен пример транзакции.



Transfer \$100 from Savings to Checking:

- Debit \$100 from Savings account
- Credit \$100 to Checking account

Рисунок 13.1. Пример транзакции

В случае рис. 13.1, например, вы хотите перевести 100 долларов со сберегательного на текущий счёт. Как показано на рисунке, для выполнения такой задачи может потребоваться следующая последовательность действий:

- Списать 100 долларов со сберегательного счёта
- Зачислить 100 долларов на текущий счёт

Представьте, что бы произошло при отключении электропитания после списания средств со сберегательного счёта, но до их зачисления на текущий счёт, если бы вышеописанная последовательность событий не рассматривалась как одна единица работы, т. е. транзакция. Вы потеряли бы 100 долларов!

13.2 Параллельное использование

Параллельное использование подразумевает, что несколько пользователей могут одновременно работать над одними и теми же объектами базы данных. Сервер данных DB2 разработан как многопользовательская база данных. Доступ к данным необходимо должным образом и, в то же время, прозрачно координировать с помощью механизма обеспечения целостности и согласованности данных. Рассмотрим рис. 13.2 в качестве примера.

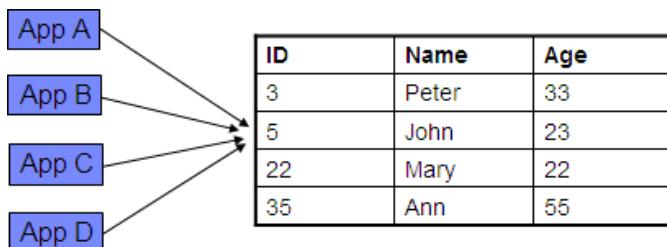


Рисунок 13.2. Пример параллельного использования и демонстрация необходимости его контроля

На рис. 13.2 показано четыре приложения, App A, App B, App C и App D, которые пытаются получить доступ к одной и той же строке (строке 2) таблицы. Если не контролировать параллельное использование, все приложения смогут выполнять операции в одной строке. Допустим, все приложения обновляют столбец Age (Возраст) для второй строки, задавая разные значения. В таком случае окончательное значение установит то приложение, которое выполнит обновление последним. Из этого примера видно, что для получения согласованных результатов требуется контроль параллельного использования. Такой контроль основан на применении блокировки.

Концепции блокировки и параллельного использования тесно взаимосвязаны. Блокировка временно запрещает приложениям выполнять другие операции до завершения текущей операции. Чем активнее в системе применяется блокировка, тем меньше остается возможностей параллельного использования. С другой стороны, чем реже блокировка применяется в системе, тем больше появляется возможностей параллельного использования.

Блокировка срабатывает автоматически по мере необходимости для поддержки транзакции и отключается после прерывания такой транзакции (с помощью команды COMMIT или ROLLBACK). Блокировка может устанавливаться на таблицы или строки. Существует два основных типа блокировки:

- Общая блокировка (S) — устанавливается, когда приложение считывает данные и не допускает внесения изменений в ту же строку другими приложениями.
- Эксклюзивная блокировка (X) — устанавливается, когда приложение обновляет, вставляет или удаляет строку.

Теперь рассмотрим рис. 13.3, который отличается от рис. 13.2 наличием блокировки.

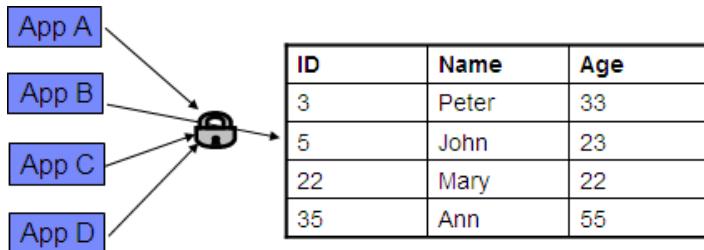


Рисунок 13.3. Пример параллельного использования и демонстрация необходимости блокировки

Например, на рис. 13.2, если первым доступ ко второй строке получает приложение App B, которое выполняет операцию UPDATE, именно App B имеет X-блокировку для этой строки. Если App A, App C и App D попытаются получить доступ к той же строке, они не смогут выполнить операцию UPDATE из-за наличия X-блокировки. Такой контроль позволяет поддерживать согласованность и целостность данных.

13.3 Проблемы, связанные с отсутствием контроля параллельного использования

Без какой-либо формы контроля параллельного использования могут возникнуть следующие проблемы:

- Потерянное обновление.
- Недостоверное чтение.
- Неповторяющееся чтение.
- Фантомное чтение.

13.3.1 Потерянное обновление

Проблема потерянного обновления похожа на проблему, описанную в этом разделе выше, когда окончательное значение устанавливает то приложение, которое выполняет обновление последним.

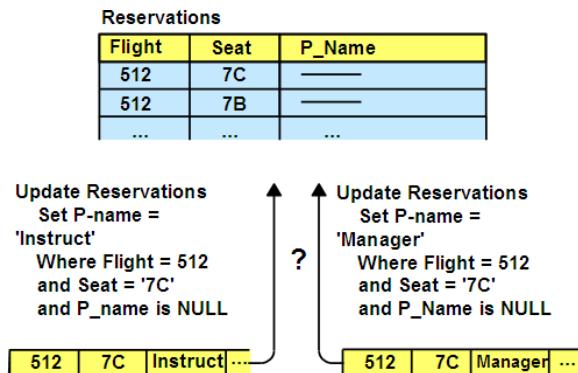


Рисунок 13.4. Потерянное обновление

На рис. 13.4 показаны два приложения, пытающихся обновить одну и ту же строку. Слева расположено приложение App1, а справа — App2. Происходит такая последовательность событий:

1. App1 обновляет строку.
2. App2 обновляет ту же строку.
3. App1 фиксирует данные.
4. App2 фиксирует данные.

Обновления, внесенные приложением App1, теряются, когда приложение App2 вносит собственные обновления. Этим объясняется термин «Потерянное обновление».

13.3.2 Недостоверное чтение

Недостоверное (или «грязное») чтение позволяет приложению считывать информацию, которая еще не была зафиксирована, а потому может быть неточной.

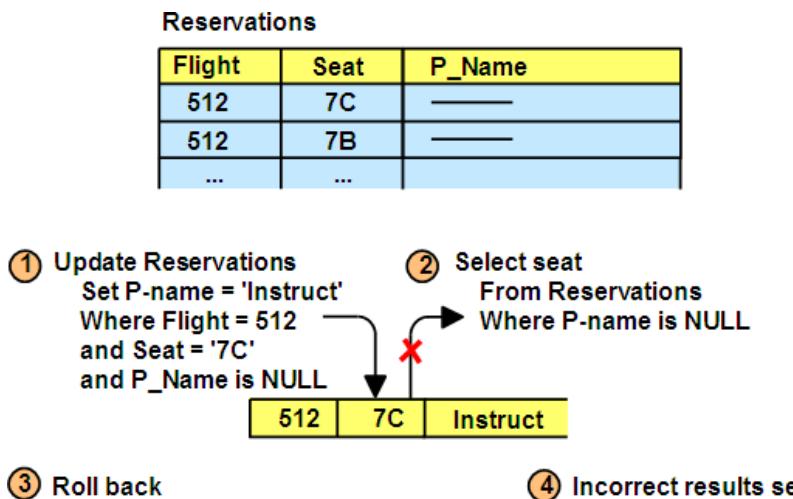


Рисунок 13.5. Недостоверное чтение

На рис. 13.5 показана такая последовательность событий:

1. App1 обновляет строку.
2. App2 считывает новое значение из этой строки.
3. App1 откатывает свои изменения, внесенные в эту строку.

App2 считывает незафиксированные (т. е. недостоверные) данные, поэтому эту проблему называют «недостоверным чтением».

13.3.3 Неповторяющееся чтение

Неповторяющееся чтение означает, что невозможно получить такой же результат, выполнив то же чтение в такой же операции.

| FLIGHT | SEAT | NAME | DESTINATION | ORIGIN |
|--------|------|------|-------------|----------|
| 512 | 7B | — | DENVER | DALLAS |
| | | | | |
| | | | | |
| 814 | 8A | — | SAN JOSE | DENVER |
| | | | | |
| 134 | 1C | — | HONOLULU | SAN JOSE |
| | | | | |

Рисунок 13.6. Неповторяющееся чтение

Рассматривая *рис. 13.6*, предположим, что вы пытаетесь забронировать билет на авиарейс из Далласа в Гонолулу. Происходит такая последовательность событий:

1. App1 открывает курсор (также известный как набор результатов), получая данные, показанные на *рис. 13.6*.
2. App2 удаляет одну из строк курсора (например, строку с пунктом назначения «Сан-Хосе»).
3. App2 фиксирует изменения.
4. App1 закрывает и снова открывает курсор.

В данном случае, поскольку приложение App1 не сможет получить те же данные при повторном считывании, оно не может сократить набор данных; поэтому проблему называют «неповторяющимся чтением».

13.3.4 Фантомное чтение

Проблема фантомного чтения похожа на проблему неповторяющегося чтения с той разницей, что при дальнейших обращениях можно получить дополнительные строки, а не меньшее их количество. На *рис. 13.7* приведен пример этой проблемы.

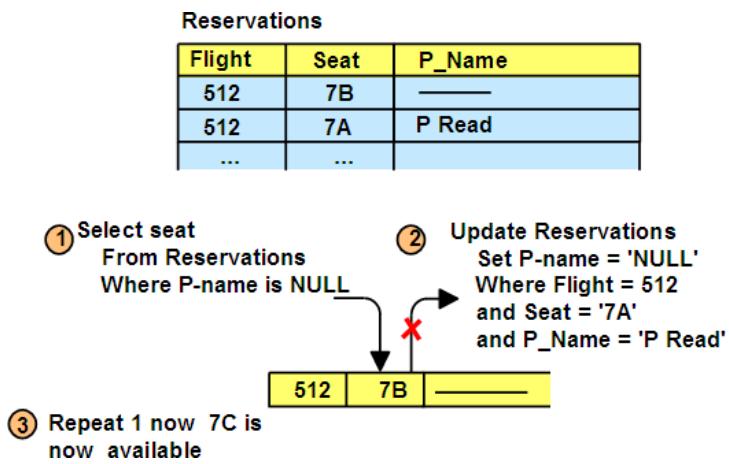


Рисунок 13.7. Фантомное чтение

На рис. 13.7 показана такая последовательность событий:

1. App1 открывает курсор.
2. App2 добавляет в базу данных строки, удовлетворяющие условию их присутствия в курсоре.
3. App2 фиксирует изменения.
4. App1 закрывает и снова открывает курсор.

В данном случае приложение App1 при повторном считывании получит не те же данные, а большее количество строк; поэтому проблему называют «фантомным чтением».

13.4 Уровни изоляции

Уровни изоляции можно рассматривать как политики блокировки, в которых, в зависимости от выбранного уровня изоляции, можно достичь различных вариантов блокировки базы данных приложением.

DB2 предоставляет разные уровни защиты для изоляции данных:

- Недостоверное чтение (Uncommitted Read, UR)
- Стабильность курсора (Cursor Stability, CS)
- Стабильность чтения (Read Stability, RS)
- Многократное чтение (Repeatable Read, RR)

13.4.1 Недостоверное чтение

Недостоверное чтение также называют «грязным». Это самый низкий уровень изоляции, который допускает наивысшую степень параллельного использования. При операциях чтения строки не блокируются, за исключением случаев, когда другое приложение пытается удалить или изменить таблицу; операции обновления выполняются так же, как при использовании следующего уровня изоляции — уровня «Стабильность курсора».

При использовании этого уровня изоляции остаются следующие проблемы:

- Недостоверное чтение.
- Неповторяющееся чтение.
- Фантомное чтение.

При использовании этого уровня изоляции предотвращаются следующие проблемы:

- Потеря обновления

13.4.2 Стабильность курсора

Стабильность курсора — это уровень изоляции по умолчанию. Он обеспечивает минимальную степень блокировки. В сущности, при этом уровне изоляции блокируется «текущая» строка курсора. Если строка только считывается, блокировка сохраняется до перехода на новую строку или завершения операции. Если строка обновляется, блокировка сохраняется до завершения операции.

При использовании этого уровня изоляции остаются следующие проблемы:

- Неповторяющееся чтение.
- Фантомное чтение.

При использовании этого уровня изоляции предотвращаются следующие проблемы:

- Потеря обновления.
- Недостоверное чтение.

13.4.2.1 Значения, принятые на текущий момент (*Currently committed*)

До выхода DB2 9.7, при использовании уровня изоляции «Стабильность курсора» выполнение записи (операция UPDATE) закрывало для чтения (операция SELECT) доступ к той же строке. Логической основой служило то, что поскольку операция записи вносит в строку изменения, средству чтения следует дождаться завершения обновлений, чтобы получить окончательное зафиксированное значение. В DB2 9.7, по умолчанию, используется другой подход к уровню изоляции «Стабильность курсора» для новых баз данных. Этот новый подход реализован с помощью принятой на текущий момент (*currently committed, CC*) семантики. При использовании CC-семантики, операция записи не закрывает доступ к той же строке для операции чтения. Ранее, такой подход был возможен при использовании уровня изоляции UR; разница с текущим подходом заключается в том, что при UR операция чтения получает **недостоверные** значения, а при CC-семантике — значения, **принятые на текущий момент**. Принятые на текущий момент значения — это значения, зафиксированные до начала операции записи.

К примеру, имеется таблица T1 со следующим содержимым:

| FIRSTNAME | LASTNAME |
|-----------|----------|
| Raul | Chong |
| Jin | Xie |

Приложение App A вызывает оператор, но не фиксирует значения:

```
update T1 set lastname = 'Smith' where firstname = 'Raul'
```

new in
V9.7

Затем приложение App B вызывает следующий оператор:

```
select lastname from T1 where firstname = 'Raul' with CS
```

До выхода DB2 9.7 мы наблюдали бы зависание такого оператора в связи с ожиданием снятия эксклюзивной блокировки, установленной оператором обновления приложения App A (операцией записи).

При использовании DB2 9.7 с включенным вариантом принятых на текущий момент значений (устанавливается по умолчанию для новых баз данных) наш оператор вернет зафиксированное на текущий момент значение, т. е. *Chong*.

Обратите внимание, что, хотя стабильность курсора используется по умолчанию, мы, для большей четкости, включаем в оператор выражение «with CS». В одной из следующих частей этой главы мы рассмотрим данное выражение более подробно.

Если App B запустит следующий оператор:

```
select lastname from T1 where firstname = 'Raul' with UR
```

Поскольку используется изоляция UR, в результате получим значение *Smith*, т. е. недостоверное значение.

Из этого примера видно, что СС-семантика предоставляет приложениям больше возможностей параллельного использования, давая операции чтения доступ к строке, которую обновляет операция записи.

Еще один сценарий, который до выхода DB2 9.7 привел бы к возникновению конфликта, — ситуация, когда операция чтения закрывает доступ к строке для операции записи. Этот сценарий был одной из причин того, почему команда COMMIT рекомендовалась даже для операций чтения, поскольку таким образом можно было гарантировать снятие общей блокировки (S). Благодаря СС-семантике, это больше не является проблемой, поскольку операция чтения не блокирует операцию записи.

По умолчанию, операция чтения будет пропускать операции INSERT с незафиксированными значениями; иными словами, в наборе результатов соответствующие строки отображаться не будут. В случае команды DELETE, операция чтения также будет пропускать (игнорировать) соответствующие строки, но это зависит от значения переменной реестра DB2_SKIPDELETED. Другие переменные и свойства реестра в командах BIND и PREPARE могут менять действие СС-семантики, определённое по умолчанию.

Не забывайте: принятые на текущий момент значения — это текущая зафиксированная информация, поэтому незафиксированные операции INSERT или DELETE будут игнорироваться.

Как уже упоминалось, вариант СС-семантики включен по умолчанию для новых баз данных. Чтобы выключить его или же включить для базы данных, созданной до выхода DB2 9.7 и обновленной до этой версии продукта, можно задать для конфигурации базы данных значение CUR_COMMIT. К примеру, для отключения СС-семантики в базе данных SAMPLE выполните следующее:

```
db2 update db cfg for sample using CUR_COMMIT off  
db2stop  
db2start
```

13.4.3 Стабильность чтения

Благодаря стабильности чтения, все строки, получаемые приложением в пределах одной единицы работы, блокируются. Для заданного курсора блокируются все строки, соответствующие набору результатов. К примеру, если из таблицы на 10 000 строк запрос возвращает 10 строк, блокироваться будут только эти 10 строк. Стабильность чтения использует среднюю степень блокировки.

При использовании этого уровня изоляции остаются следующие проблемы:

- Фантомное чтение.

При использовании этого уровня изоляции предотвращаются следующие проблемы:

- Потеря обновления.
- Недостоверное чтение.
- Неповторяющееся чтение.

13.4.4 Повторяющееся чтение

Повторяющееся чтение — это наивысший уровень изоляции. Он предоставляет наивысшую степень блокировки и меньше всего возможностей параллельного использования. Блокировка устанавливается на строки, которые обрабатываются для построения набора результатов; иными словами, могут блокироваться даже те строки, которые не попадут в конечный пакет результатов. Другие приложения не могут обновлять, удалять или вставлять строки, которые повлияют на набор результатов, пока выполняемая операция не будет завершена. Повторяющееся чтение гарантирует, что один и тот же запрос, созданный приложением несколько раз за одну операцию, каждый раз будет выводить одинаковые результаты.

При использовании этого уровня изоляции остаются следующие проблемы:

- Нет

При использовании этого уровня изоляции предотвращаются следующие проблемы:

- Потеря обновления.
- Недостоверное чтение.
- Неповторяющееся чтение.
- Фантомное чтение.

13.4.5 Сравнение уровней изоляции

На *рис. 13.8* показано сравнение различных уровней изоляции при получении данных. На рисунке видно, что при уровне изоляции UR блокировка не устанавливается. Уровень изоляции CS блокирует строку 1 при получении из неё данных, но снимает блокировку, как только переходит к строке 2 и т. д. При использовании уровня изоляции RS или RR блокируются все строки, из которых получаются данные, и блокировка не пропадает до завершения транзакции (в точке фиксации данных).

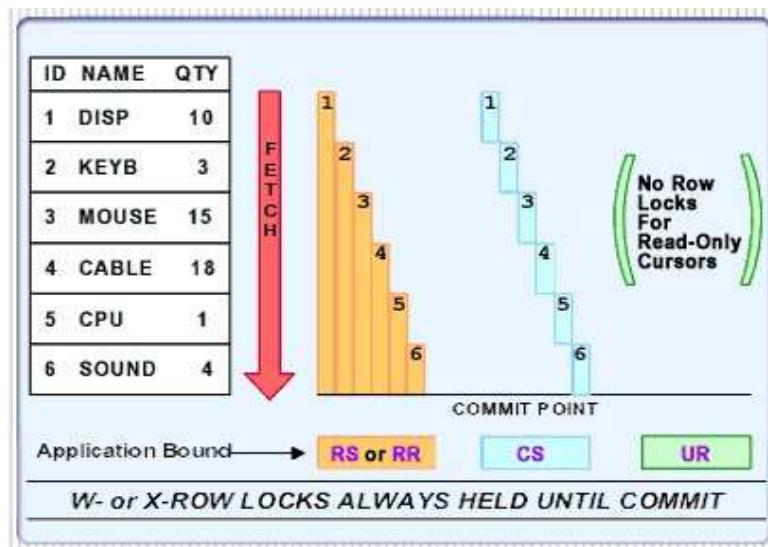


Рисунок 13.8. Сравнение уровней изоляции при получении данных

13.4.6 Установка уровня изоляции

Уровни изоляции можно задать на многих уровнях:

- уровень сеанса (приложения);
- уровень соединения;
- уровень оператора.

Обычно уровень изоляции определяется на уровне сеанса или *приложения*. Если для приложения не задан уровень изоляции, то, по умолчанию, используется уровень изоляции CS - «Стабильность курсора». Например, в таблице 13.1 показаны возможные уровни изоляции, принятые в .NET или JDBC, а также то, как они соответствуют уровням изоляции DB2.

| DB2 | .NET | JDBC |
|---|-----------------|------------------------------|
| Недостоверное чтение (Uncommitted Read, UR) | ReadUncommitted | TRANSACTION_READ_UNCOMMITTED |
| Стабильность курсора (Cursor Stability, CS) | ReadCommitted | TRANSACTION_READ_COMMITTED |
| Стабильность чтения (Read Stability, RS) | RepeatableRead | TRANSACTION_REPEATABLE_READ |
| Многократное чтение (Repeatable Read, RR) | Serializable | TRANSACTION_SERIALIZABLE |

Таблица 13.1. Сравнение терминологии, относящейся к уровням изоляции

Уровень изоляции оператора можно задать с помощью выражения WITH {уровень изоляции}. Например:

```
SELECT ... WITH {UR | CS | RS | RR}
```

Пример сценария:

Приложению необходимо знать примерное количество строк в таблице. Наиболее важным показателем является производительность. Требуется уровень изоляции стабильности курсора, за исключением одного SQL-оператора:

```
SELECT COUNT(*) FROM tab1 WITH UR
```

Для встроенного SQL-запроса уровень устанавливается при связывании, а для динамического SQL-запроса — при его выполнении.

Выбор уровня изоляции зависит от приложения. Если приложение не нуждается в точных количествах, как в примере выше, выберите изоляцию UR. Если приложению необходимо очень тщательно контролировать данные, с которыми ведется работа, выберите изоляцию RR.

Чтобы зафиксированную семантику на момент связывания или подготовки, воспользуйтесь следующей синтаксической структурой:

BIND:

```
>--+-----+-->
'--CONCURRENTACCESSRESOLUTION---+--USE CURRENTLY COMMITTED---+--'
      '--WAIT FOR OUTCOME-----'
```

PREPARE:

```
concurrent-access-resolution:
|---+--USE CURRENTLY COMMITTED---+-----|
  '---WAIT FOR OUTCOME-----'
```

В JDBC-приложениях, использующих драйвер «IBM Data Server для JDBC и SQLJ» можно воспользоваться свойством concurrentAccessResolution для включения принятых на текущий момент значений.

13.5 Эскалация блокировок

Каждая установленная блокировка потребляет определенный объем памяти. Если оптимизатор считает, что лучше установить одну блокировку для целой таблицы, чем множество блокировок на уровне строк, происходит эскалация блокировок. Этот процесс проиллюстрирован на рис. 13.9.



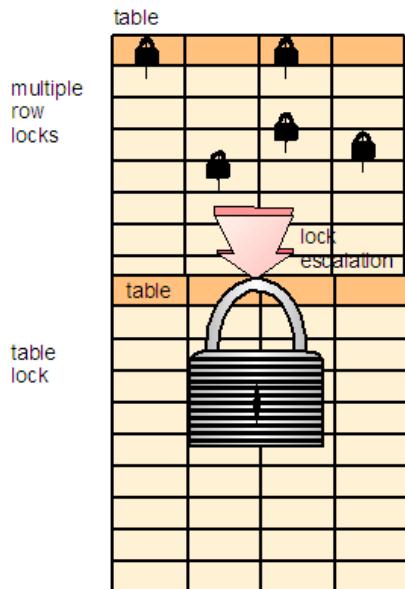


Рисунок 13.9. Эскалация блокировок

С эскалацией блокировок связаны два основных параметра конфигурации базы данных:

- **LOCKLIST** — объем памяти (в страницах по 4 КБ), отложенный для управления блокировками всех подключенных приложений.
- **MAXLOCKS** — максимальный процент общего количества блокировок, который может использовать одно приложение.

По умолчанию для обоих параметров задано значение AUTOMATIC, т. е. размер будет модифицироваться менеджером по автонастройке памяти (self-tuning memory manager – STMM). Если не включить STMM-менеджер и задать значения самостоятельно, эти значения будут влиять на время эскалации блокировок. Например, если задать для параметра LOCKLIST значение 200 КБ, а для MAXLOCKS — 22%, эскалация блокировок будет происходить, когда одно приложение потребует больше 44 КБ памяти ($200 \text{ КБ} * 22\% = 44 \text{ КБ}$). Если при таких настройках эскалация блокировок происходит слишком часто, повысьте значения параметров LOCKLIST и MAXLOCKS. Эскалация блокировок положительно сказывается на производительности, поскольку уменьшается возможность параллельного использования. Чтобы определить, происходит ли эскалация блокировок, можно воспользоваться файлом диагностического журнала DB2 (db2diag.log). См. Приложение А, чтобы узнать об этом файле более подробно.

13.6 Мониторинг блокировок

Отслеживать использование блокировок можно с помощью срезов блокировки приложений DB2. Чтобы включить срезы для блокировок, выполните следующую команду:

```
UPDATE MONITOR SWITCHES USING LOCK ON
```

После включения коммутатора будет собираться контрольная информация. Чтобы получить отчет о блокировках в определенный момент, выполните следующую команду:

```
GET SNAPSHOT FOR LOCKS FOR APPLICATION AGENTID <handle>
```

На рис. 13.9 показаны результаты среза образца блокировки приложений.

| Application Lock Snapshot | |
|---------------------------|--------------------------------|
| Snapshot timestamp | = 11-05-2002 00:09:08.672586 |
| Application handle | = 9 |
| Application ID | = *LOCAL.DB2.00B9C5050843 |
| Sequence number | = 0001 |
| Application name | = db2bp.exe |
| Authorization ID | = ADMINISTRATOR |
| Application status | = UOW Waiting |
| Status change time | = Not Collected |
| Application code page | = 1252 |
| Locks held | = 4 |
| Total wait time (ms) | = 0 |
| List Of Locks | |
| Lock Name | = 0x05000700048001000000000052 |
| Lock Attributes | = 0x00000000 |
| Release Flags | = 0x40000000 |
| Lock Count | = 255 |
| Hold Count | = 0 |
| Lock Object Name | = 98308 |
| Object Type | = Row |
| Tablespace Name | = TEST4K |
| Table Schema | = ADMINISTRATOR |
| Table Name | = T2 |
| Mode | = X |

Рисунок 13.9. Срез блокировки приложений

new in
V9.7

Примечание.

В DB2 9.7 осуществляются попытки отделить мониторинг базы данных от системного монитора и технологии срезов и приблизить его к SQL-доступу к внутренней памяти,

в частности, к использованию функций таблицы управления операциями и инструментов IBM Data Studio. С более подробной информацией можно ознакомиться в официальной документации по DB2.

13.7 Ожидание блокировки

Если двум и больше приложениям необходимо выполнить операцию с одним и тем же объектом, одному из них придется подождать, чтобы установить требуемую блокировку. По умолчанию, приложение будет ждать бесконечно долго. Временем ожидания блокировки приложением управляет параметр конфигурации базы данных LOCKTIMEOUT. По умолчанию этот параметр имеет значение -1 (бесконечное ожидание).

Для установки времени ожидания блокировки в определенном подключении можно использовать реестр CURRENT LOCK TIMEOUT. По умолчанию для этого реестра задано значение LOCKTIMEOUT. Чтобы изменить это значение, можно воспользоваться оператором SET LOCK TIMEOUT. Если задать значение этого реестра для определенного подключения, оно будет использоваться во всех транзакциях.

Пример:

```
SET LOCK TIMEOUT=WAIT n
```

13.8 Причины и обнаружение взаимоблокировки

Взаимоблокировка возникает, когда два или больше приложений, подключенных к одной базе данных, бесконечно долго ожидают ресурса. Период ожидания не может закончиться, поскольку каждое из приложений удерживает ресурс, необходимый другому приложению. В большинстве случаев проблема взаимоблокировки связана со структурой приложений. Взаимоблокировка проиллюстрирована на рис. 13.10.

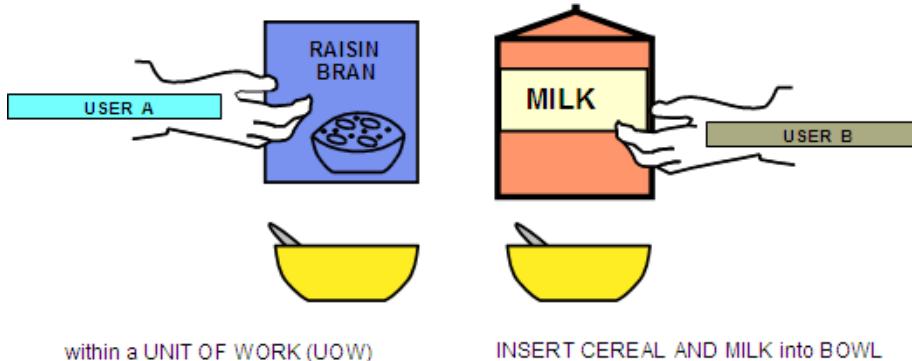


Рисунок 13.10. Сценарий взаимоблокировки

На рис. 13.10 пользователь А держит пачку хлопьев и не отпустит её, пока не получит молоко. С другой стороны, пользователь В держит молоко и не отпустит его, пока не получит хлопья. Соответственно, возникла ситуация взаимоблокировки.

new in V9.7

В DB2 9.7 использование СС-семантики существенно сократило частоту взаимоблокировки, поскольку одно приложение не будет ждать, пока другое снимет блокировку, а просто получит доступ к значению, принятому на текущий момент.

Чтобы смоделировать ситуацию взаимоблокировки в DB2, выполните следующие шаги:

1. Отключите использование значений, принятых на текущий момент:

```
db2 update db cfg for sample using cur_commit off
db2stop force
db2start
```

2. Откройте командные окна DB2 (далее — «CLP1» и «CLP2», соответственно), представляющие два разных приложения, подключающихся к базе данных.
3. В CLP1 выполните команды:

```
db2 connect to sample
db2 +c update employee set firstnme = 'Mary' where empno = '000050'
```

Сначала мы подключаемся к базе данных **SAMPLE**, а затем запускаем оператор обновления для строки со значением «empno = 000050» в таблице «employee». Опция «+с» в операторе означает, что мы не хотим, чтобы командное окно DB2 автоматически фиксировало этот оператор. Мы делаем это преднамеренно, чтобы сохранить блокировку.

4. В CLP2 выполните команды:

```
db2 connect to sample
db2 +c update employee set firstnme = 'Tom' where empno = '000030'
```

В окне CLP2, которое соответствует второму приложению, мы также подключаемся к базе данных SAMPLE, но обновляем другую строку таблицы «employee».

5. В CLP1 выполните:

```
db2 +c select firstnme from employee where empno = '000030'
```

После нажатия клавиши Enter для выполнения показанного выше оператора SELECT может сложиться впечатление, что оператор SELECT завис. На самом деле он не завис, а просто ожидает снятия эксклюзивной блокировки, установленной CLP2 для этой строки во время выполнения шага 4. Если на этом этапе для параметра LOCKTIMEOUT оставлено значение по умолчанию -1, приложение CLP1 будет ждать до бесконечности.

6. В CLP2 выполните:

```
db2 +c select firstnme from employee where empno = '000050'
```

Запуская показанный выше оператор SELECT, мы создаем взаимоблокировку. Будет складываться впечатление, что этот оператор SELECT также завис,

поскольку он ожидает снятия эксклюзивной блокировки, установленной CLP1 для этой строки во время выполнения шага 3.

В представленном выше сценарии взаимоблокировки DB2 проверит параметр конфигурации базы данных DLCHKTIME. Этот параметр задает интервал, с которым проверяется наличие взаимоблокировок. К примеру, если для этого параметра указано значение 10 секунд, DB2 каждые 10 секунд будет проверять, не возникла ли взаимоблокировка. Обнаружив действительно возникшую взаимоблокировку, DB2 воспользуется внутренним алгоритмом, чтобы определить, какую из двух транзакций необходимо откатить, а какую продолжить.

Если возникло несколько взаимоблокировок, повторно проверьте существующие транзакции и по возможности выполните реструктуризацию.

13.9 Оптимальные подходы к параллельному использованию и блокировке

Ниже представлено несколько подсказок для обеспечения наивысшего уровня параллельного использования:

1. Убедитесь в том, что СС-семантика включена, если позволяет логика приложения.
2. Используйте как можно более короткие транзакции. Для этого часто запускайте операторы COMMIT (даже в транзакциях только для чтения), если позволяет логика приложения.
3. Записывайте информацию о транзакции в журнал только в случае необходимости.
4. Быстро очищайте данные. Можно выполнить команду

```
ALTER TABLE ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE
```

**new in
V9.7**

а теперь в DB2 9.7 — команду TRUNCATE:

```
TRUNCATE <имя_таблицы>
```

5. Выполняйте модификацию данных для пакетов/групп. Например:

```
DELETE FROM (
    SELECT * FROM tedwas.t1 WHERE c1 = ... FETCH FIRST 3000 ROWS ONLY)
```

6. Используйте функции параллельного использования в инструментах перемещения данных DB2.
7. Установите значение параметра уровня базы данных LOCKTIMEOUT (рекомендуемое время — 30—120 секунд). Не оставляйте используемое по умолчанию значение -1. Также можно воспользоваться задержкой блокировки на основе сеанса.

-
8. Не извлекайте больше данных, чем требуется. Используйте, например, выражение FETCH FIRST n ROWS ONLY операторов SELECT.

Примечание.

Более подробную информацию об оптимальных подходах к параллельному использованию и блокировке см. в документах «Best Practices» (оптимальные подходы) по адресу <http://www.ibm.com/developerworks/data/bestpractices/>

13.10 Краткий обзор

В этой главе мы рассмотрели вопрос поддержания целостности данных посредством контроля транзакций, параллельного доступа пользователей и уровней блокировки. Все уровни параллельного использования имеют определенные недостатки, негативно влияющие на доступ к данным и управление ими.

Мы также подробно изучили роль установки уровней изоляции для устранения таких недостатков, а также научились управлять уровнями изоляции для обеспечения максимальной гибкости, необходимой для приложений и данных.

Мы рассмотрели эскалацию блокировок, ожидание и мониторинг блокировок, а также причины, обнаружение и устранение проблемы взаимоблокировок.

Наконец, мы ознакомились с оптимальными решениями для получения наилучших возможных результатов параллельного использования.

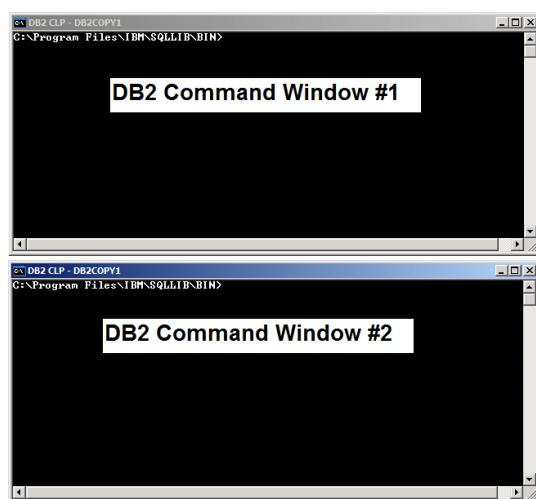
13.11 Упражнения

В этом упражнении мы подробнее изучим рассмотренные в этой главе понятия параллельного использования и блокировок, используя командное окно DB2. Этот инструмент по умолчанию использует блокировку уровня изоляции CS. Выполнив оператор SQL, командное окно автоматически зафиксирует значения (это действие также известно как автофиксация). Для иллюстрации этого упражнения воспользуемся меткой `+c`, чтобы отключить автофиксацию, а также опцией `WITH <уровень изоляции>` после выполнения нескольких SQL-операторов, чтобы заменить используемую по умолчанию изоляцию CS.

Часть 1. Тестирование проблемы фантомного чтения и уровня изоляции RR

Процедура:

1. Откройте командные окна DB2, как показано на рисунке ниже. Окно, расположенное вверху, будем называть «Командное окно DB2 #1», а внизу — «Командное окно DB2 #2».



2. В командном окне DB2 #1 выполните:

```
db2 connect to sample  
db2 +c select * from staff
```

На выходе будет 35 записей.

3. В командном окне DB2 #2 выполните:

```
db2 connect to sample  
db2 +c insert into staff (id,name) values (400, 'test')
```

4. В командном окне DB2 #1 выполните:

```
db2 +c select * from staff
```

На выходе все ещё будет 35 записей

5. В командном окне DB2 #2 выполните:

```
db2 commit
```

6. В командном окне DB2 #1 выполните:

```
db2 +c select * from staff
```

Теперь на выходе 36 записей.

Командное окно DB2 #1 обозначает одно приложение, открывающее курсор или набор результатов (`select * from staff`) и получая 35 записей. В рамках той же транзакции (поскольку в этом окне мы не выполняем операторы `commit`) приложение открывает тот же курсор и всё ещё видит 35 записей, даже после того, как приложение в командной строке DB2 #2 вставляет (но не фиксирует) новую запись.

Затем приложение командного окна DB2 #2 фиксирует вставку, после чего приложение командного окна DB2 #1 в третий раз открывает курсор. В наборе результатов появляется ещё одна строка (phantomное чтение), и на выходе получается 36 записей. Этот пример иллюстрирует проблему фантомного чтения: открытие одинаковых курсоров в рамках одной транзакции приводит к появлению дополнительных строк. Мы используем уровень изоляции CS, который, как уже упоминалось в этой главе, не предотвращает фантомное чтение.

7. Прежде чем перейти к следующим шагам, очистите вставленную запись:

В командном окне DB2 #1:

```
db2 rollback
```

В командном окне DB2 #2:

```
db2 delete from staff where id = 400
```

```
db2 select * from staff
```

В результате снова будет 35 записей.

8. Теперь проверим, сможет ли уровень изоляции RR предотвратить проблему фантомного чтения.

В командном окне DB2 #1 выполните:

```
db2 connect to sample
```

```
db2 +c select * from staff with RR
```

В результате будет 35 записей.

В командном окне DB2 #2:

```
db2 connect to sample
```

```
db2 +c insert into staff (id,name) values (400, 'test')
```

Как и ожидалось, этот оператор зависнет.

Поскольку опция WITH RR добавляется в операторе SELECT в командном окне DB2 #1, этот уровень изоляции предотвращает выполнение операции INSERT в строке, которая может повлиять на набор результатов. Этот пример показал, что уровень изоляции RR действительно предотвращает проблему фантомного чтения.

9. Выполните очистку, прежде чем перейти ко второй части упражнения:

В командном окне DB2 #2:

Ctrl-C (чтобы прервать)

Закройте окно

В командном окне DB2 #1:

```
db2 rollback
```

Закройте окно

Часть 2. Тестирование уровней CC и UR

Процедура 1. Анализ действия уровня изоляции CS без принятых на текущий момент значений

1. Откройте командное окно DB2 и выполните следующие операторы:

```
db2 connect to sample
```

```
db2 select * from staff
```

Проверьте содержимое таблицы *STAFF*, обращая особое внимание на *ID* со значением «10». Соответствующий столбец *NAME* имеет значение *Sanders*. Закройте окно.

2. Для новых баз данных принятые на текущий момент значения используются по умолчанию. Убедитесь в том, что принятые на текущий момент значения действительно включены для образца базы данных:

```
db2 get db cfg for sample
```

Ближе к концу вывода результатов найдите строку, в которой сказано:

```
Currently Committed (CUR_COMMIT) = ON
```

Если значение ON, сначала измените его на OFF, чтобы мы смогли проанализировать действие уровня изоляции CS так, как до выхода версии DB2 9.7:

```
db2 update db cfg for sample using CUR_COMMIT off
db2 force applications all
```

Добавление опции *force* гарантирует отсутствие подключений, поэтому обновление CUR_COMMIT вступит в силу при следующем подключении)

Убедитесь, что CUR_COMMIT отключен. В строке должно быть написано:

```
Currently Committed (CUR_COMMIT) = DISABLED
```

3. Откройте два командных окна DB2, как в первой части — одно над другим. Проверим, как уровень изоляции CS работает без принятых на данный момент значений, если операции update (средство записи) и select (средство чтения) направлены на одну и ту же строку. Обратите внимание на то, что нет необходимости указывать «WITH CS» после операторов (поскольку такое значение используется по умолчанию).

В командном окне DB2 #1 (средство записи):

```
db2 connect to sample
db2 +c update staff set name = 'Chong' where id = 10
```

В командном окне DB2 #2 (средство чтения):

```
db2 connect to sample
db2 +c select * from staff
```

Оператор SELECT ожидает снятия эксклюзивной (X) блокировки приложением командного окна #1 DB2. Как видим, действие CS по умолчанию до выхода DB2 9.7 дает меньше возможностей параллельного использования

В командном окне DB2 #2:

CTRL-C (нажмите эти клавиши для прерывания)

Закройте окно

В командном окне DB2 #1:

db2 rollback

Закройте окно

Процедура 2. Анализ действия уровня изоляции CS с принятыми на текущий момент значениями

1. Установите для принятых на текущий момент значений значение ON:

Откройте командное окно DB2 и выполните команду:

```
db2 update db cfg for sample using CUR_COMMIT on  
db2 force applications all
```

Закройте окно.

2. Откройте два командных окна DB2, как в первой части — одно над другим. Затем выполните следующее:

В командном окне DB2 #1:

```
db2 connect to sample  
db2 +c update staff set name = 'Chong' where id = 10
```

В командном окне DB2 #2:

```
db2 connect to sample  
db2 +c select * from staff
```

Теперь оператор SELECT работает! Он не зависает, а отображает значение *Sanders*, т. е. принятое на текущий момент значение.

В командном окне DB2 #1:

db2 rollback

Закройте окно.

В командном окне DB2 #2:

db2 rollback

Закройте окно.

Процедура 3. Анализ действия уровня изоляции UR

1. Откройте два командных окна DB2, как в первой части — одно над другим.
Затем выполните следующее:

В командном окне DB2 #1:

```
db2 connect to sample  
db2 +c update staff set name = 'Chong' where id = 10
```

В командном окне DB2 #2:

```
db2 connect to sample  
db2 +c select * from staff with UR
```

Оператор SELECT работает, но обратите внимание на то, что отображается значение *Chong*, т. е. незафиксированное значение.

В командном окне DB2 #1:

```
db2 rollback  
Закройте окно.
```

В командном окне DB2 #2:

```
db2 rollback  
Закройте окно.
```

ЧАСТЬ III — ИЗУЧЕНИЕ DB2: РАЗРАБОТКА ПРИЛОЖЕНИЙ

14

Глава 14. Введение в разработку приложений DB2

IBM DB2 — это мощное программное обеспечение сервера данных для управления как реляционными, так и XML-данными. IBM DB2 обеспечивает гибкость не только администраторам, но и разработчикам баз данных. Независимо от того, какой язык используется для разработки программ, DB2 предоставляет все драйверы, адаптеры и расширения, необходимые для работы с базами данных как частью разрабатываемого приложения. Более того, с DB2 Express-C приложения можно разрабатывать бесплатно, без ограничений на размер базы данных и с таким же уровнем поддержки языков программирования, как в других версиях DB2. Приложения, разработанные с помощью DB2 Express-C, могут без каких-либо модификаций работать в любых редакциях DB2.

Примечание.

Этот раздел является всего лишь введением в разработку приложений в DB2. В рамках проекта *DB2 on Campus Book Series* разрабатывается серия из более чем 25 бесплатных электронных книг, среди которых — книга, посвященная исключительно разработке приложений в DB2. В серию также входят книги на темы, не относящиеся к DB2, в частности о разработке на Java, PHP, Ruby on Rails, Python, Perl, Web 2.0, SOA, Eclipse, разработке с открытым исходным кодом, облачных вычислениях и пр. Часть книг более подробно рассматривает такие технологии IBM, как pureQuery, Data Studio, InfoSphere Data Architect. Эти книги будут доступны, начиная с октября 2009 года.

14.1 Разработка приложений DB2: общий обзор

DB2 предлагает разработчикам баз данных гибкость, необходимую для использования преимуществ функций разработки на стороне сервера, таких как хранимые процедуры и пользовательские функции, в то же время, не ограничивая разработчиков в выборе языка программирования. Гибкость продукта проиллюстрирована на Рисунке 14.1.

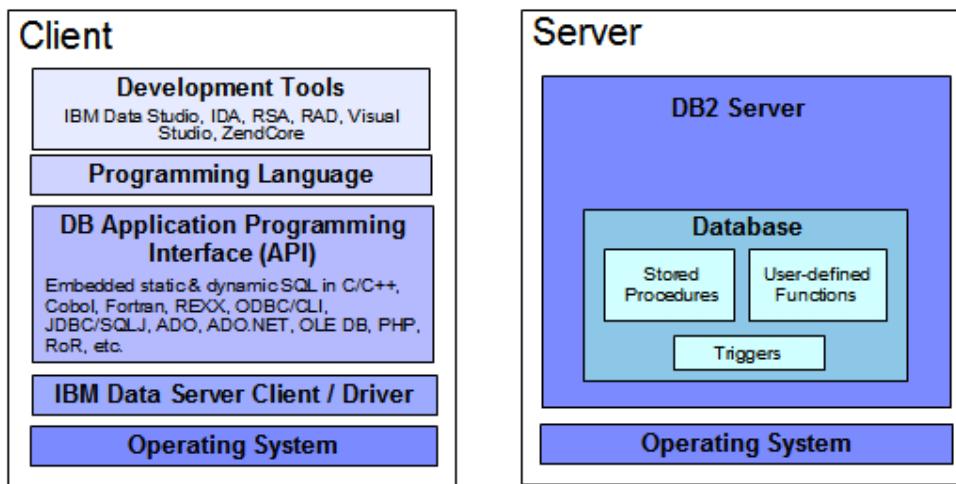


Рисунок 14.1. DB2 для всех: для разработчиков приложений и баз данных

На рис. 14.1 слева показан компьютер-клиент, на котором программист разрабатывает и выполняет свою программу. На этом компьютере-клиенте кроме операционной системы может быть установлен еще и клиент IBM Data Server, в зависимости от типа разрабатываемого приложения. Клиент IBM Data Server включает необходимые драйверы соединения, в частности драйверы JDBC и ODBC/CLI. Эти драйверы также можно загрузить отдельно с веб-сайта IBM DB2 Express-C по адресу ibm.com/db2/express.

С помощью таких программных средств, как IBM Data Studio, InfoSphere Data Architect (IDA), Rational Software Architect (RSA), Rational Application Developer (RAD) и пр., можно разрабатывать приложения на любом языке программирования. Библиотеки API с поддержкой таких языков также содержатся в клиенте IBM Data Server, и при подключении к серверу DB2 все программные инструкции с помощью этих API должным образом конвертируются в операторы SQL или XQuery, поддерживаемые DB2. В Таблице 1.1 представлено краткое описание упомянутых выше инструментов.

| Название инструмента | Описание |
|----------------------|---|
| IBM Data Studio | IBM Data Studio — это инструмент на базе Eclipse, позволяющий пользователям управлять серверами данных и разрабатывать хранимые процедуры, функции и веб-службы данных. Инструмент IBM Data Studio рассматривался в предыдущих разделах этой книги. |
| IDA | IDA — это инструмент моделирования данных. Он помогает спроектировать логические и физические схемы базы данных. |
| RSA | RSA — это программное средство на базе Eclipse для построения диаграмм UML. |
| RAD | RAD — это инструмент быстрой разработки приложений на базе Eclipse для разработчиков программного обеспечения. |

| | |
|---------------|--|
| Visual Studio | Microsoft Visual Studio — это среда IDE, позволяющая разрабатывать приложения на платформе Windows с применением технологий Microsoft. |
| ZendCore | Ранее известная как ZendCore для IBM, это бесплатная среда IDE для разработки приложений PHP. |

Таблица 14.1. Инструменты, помогающие разрабатывать приложения в DB2

С правой стороны *рис. 14.1* изображен сервер DB2 с одной базой данных. Эта база данных содержит хранимые процедуры, пользовательские функции и триггеры. Все эти объекты рассмотрены более подробно в последующих разделах.

14.2 Разработка на стороне сервера

Разработка на стороне сервера в DB2 подразумевает разработку и хранение объектов приложений в базе данных DB2. В этом разделе кратко рассмотрены следующие объекты приложений:

- хранимые процедуры;
- пользовательские функции (UDF);
- триггеры.

14.2.1 Хранимые процедуры

Хранимая процедура — это объект приложения базы данных, содержащий операторы SQL и бизнес-логику. Хранение части логики приложения в базе данных повышает производительность, поскольку сокращается объем трафика между приложением и базой данных. Кроме того, хранимые процедуры предоставляют централизованное местоположение для хранения программного кода, и соответственно, другие приложения могут воспользоваться теми же хранимыми процедурами. Для вызова хранимой процедуры используется оператор `CALL`. В DB2 хранимые процедуры можно разрабатывать на нескольких языках, среди которых SQL PL, Java, C/C++, CLR, OLE и COBOL. Ниже показан простой пример создания и вызова хранимой процедуры на SQL PL из командного окна DB2 (для Windows) или терминала (для Linux):

```
db2 create procedure P1 begin end
db2 call P1
```

В этом примере процедура P1 — это пустая хранимая процедура, не выполняющая никаких действий. Из примера видно, насколько легко создать хранимую процедуру. Для разработки хранимых процедур с более сложной логикой рекомендуем использовать IBM Data Studio со встроенным отладчиком.

14.2.2 Пользовательские функции (UDF)

UDF — это объект приложения базы данных, позволяющий пользователям расширить язык SQL собственной логикой. Функция всегда возвращает значение или значения, обычно как результат включенной в функцию бизнес-логики. Чтобы вызвать функцию, используйте её в составе оператора SQL или с функцией **values**. В DB2 пользовательские функции можно разрабатывать на нескольких языках, среди которых SQL PL, Java, C/C++, OLE и CLR.

Ниже показан простой пример создания и вызова пользовательской функции на SQL PL из командного окна DB2 (для Windows) или командного процессора (для Linux):

```
db2 create function F1() returns integer begin return 1000; end  
db2 values F1
```

В этом примере функция F1 — это функция, возвращающая целое значение 1000. Для вызова этой функции можно воспользоваться оператором **values**. Как и в случае хранимых процедур, рекомендуем создавать функции с помощью IBM Data Studio.

14.2.3 Триггеры

Триггер — это объект, который автоматически выполняет операцию с таблицей или представлением. Определенное действие с объектом, для которого определен триггер, вызывает запуск триггера. Обычно триггер не считается объектом приложения; соответственно, обычно триггеры создаются не разработчиками, а администраторами баз данных. Мы включили триггеры в этот раздел, поскольку для их создания требуется определенный уровень программирования. Ниже представлен пример триггера:

```
create trigger myvalidate no cascade before insert on T1  
referencing NEW as N  
for each row  
begin atomic  
    set (N.myxmlcol) = XMLVALIDATE(N.myxmlcol  
        according to xmlschema id myxmlschema);  
end
```

В этом примере триггер срабатывает перед выполнением операции INSERT в таблице T1. Триггер вставит значение (являющееся XML-документом), но при этом запустит функцию XMLVALIDATE для проверки этого XML-документа по заданной схеме. Документы XML и схемы XML рассмотрены более подробно в Главе 15 «Технология pureXML в DB2».

14.3 Разработка на стороне клиента

Сам термин указывает на то, что при разработке на стороне клиента разработчики программируют приложения на компьютере-клиенте, а затем устанавливают связь и получают доступ к базе данных DB2 с помощью API, предоставленных в DB2. В этом разделе рассмотрены следующие вопросы:

- Встроенный SQL.
- Статический и динамический SQL.
- CLI и ODBC.
- JDBC, SQLJ и pureQuery.
- OLE DB.
- ADO.NET.
- PHP.
- Ruby on Rails.
- Perl.
- Python.

14.3.1 Встроенный SQL

Приложения на встроенном SQL — это приложения, в которых язык SQL встроен в язык хоста, например, C, C++ или COBOL. Приложение на встроенном SQL может содержать как статический, так и динамический SQL (см. следующий раздел). На рис. 14.2 показан принцип построения приложения на встроенном SQL.

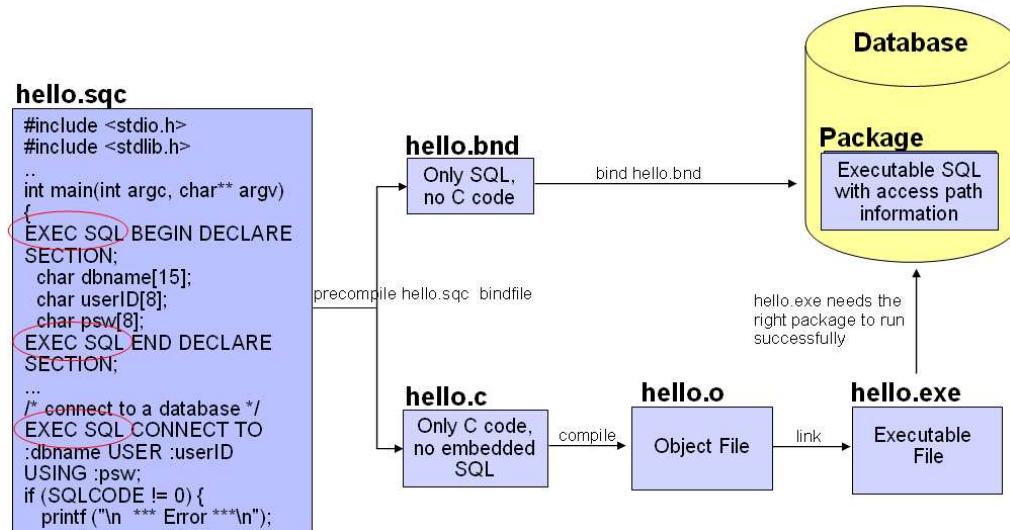


Рисунок 14.2. Построение приложений на встроенном SQL

На рисунке программа на C `hello.sqc` содержит встроенный SQL. API встроенного SQL для языка С использует EXEC SQL (выделено на Рисунке 14.2), чтобы разрешить процессу предварительной компиляции различать операторы встроенного SQL и фактический код С. В листинге `hello.sqc` также можно заметить, что некоторые переменные имеют двоеточие в качестве префикса, например `:dbname`, `:userID` и `:psw`. Такие переменные называются переменными хоста. Переменные хоста — это переменные языка хоста, ссылки на которые содержатся в операторах встроенного SQL.

Выполнение команды `precompile` (также известной как команда `prep`) с опцией `bindfile` генерирует два файла: файл редактора связей `hello.bnd`, содержащий только операторы SQL, и файл `hello.c`, содержащий только программный код С. Файл оператора связей компилируется с помощью команды `bind` для получения **пакета**, который хранится в базе данных. Пакет включает скомпилированный/выполнимый SQL и план доступа, по которому DB2 будет извлекать данные. Для выполнения команды `bind` должно существовать соединение с базой данных. В нижней части рисунка показано, что файл `hello.c` компилируется и подключается, как любая стандартная программа на С. Получаемый в результате выполнимый файл `hello.exe` для успешного выполнения должен соответствовать пакету, хранимому в базе данных.

14.3.2 Статический и динамический SQL

Статические SQL-операторы — это операторы, структура SQL которых полностью известна во время прекомпиляции. Например:

```
SELECT lastname, salary FROM employee
```

В этом примере имена столбцов (`lastname`, `salary`) и таблицы (`employee`), на которые ссылается оператор, полностью известны во время прекомпиляции. Ниже приведен еще один пример статического SQL-оператора:

```
SELECT lastname, salary FROM employee WHERE firstnme = :fname
```

Во втором примере переменная хоста `:fname` используется как часть встроенного SQL-оператора. Хотя значение переменной хоста до запуска неизвестно, тип данных этой переменной известен из программы, а все прочие объекты (имена столбцов, имена таблиц) заранее известны полностью. DB2 использует оценки этих переменных хоста для расчета плана доступа наперёд; соответственно, этот случай также рассматривается как статический SQL.

Перед запуском приложения необходимо выполнить предварительную компиляцию, связать и скомпилировать статически выполненные SQL-операторы. Статический SQL лучше всего использовать в базах данных, статистика которых не сильно изменяется. Теперь рассмотрим еще один пример:

```
SELECT ?, ? FROM ?
```

В этом примере имена столбцов и таблицы, на которые ссылается оператор, неизвестны до момента выполнения. Соответственно, план доступа рассчитывается только при выполнении с использованием доступных статистических данных. Операторы такого типа считаются **динамическими SQL-операторами**.

Некоторые программные API, такие как JDBC и ODBC, всегда используют динамический SQL, независимо от того, включает SQL-оператор известные объекты или нет. К примеру, в операторе `SELECT lastname, salary FROM employee` все имена столбцов и таблиц известны заранее, но при использовании JDBC или ODBC прекомпиляция операторов не выполняется. Все планы доступа для операторов рассчитываются при их выполнении.

В целом, для рассмотрения SQL-оператора как динамического используется два оператора:

- **PREPARE**: этот оператор готовит или компилирует SQL-оператор, рассчитывая план доступа, который будет применяться для извлечения данных.
- **EXECUTE**: этот оператор выполняет SQL.

PREPARE и **EXECUTE** также можно выполнить с помощью одного оператора: **EXECUTE IMMEDIATE**

В листинге 14.1 показан пример подготавливаемого и выполняемого динамического оператора SQL на встроенным С.

```
strcpy(hVStmtDyn, "SELECT name FROM emp WHERE dept = ?");  
PREPARE StmtDyn FROM :hVStmtDyn;  
EXECUTE StmtDyn USING 1;  
EXECUTE StmtDyn USING 2;
```

Листинг 14.1. Динамический SQL-оператор на встроенном С, использующий PREPARE и EXECUTE

В листинге 14.2 показан тот же пример, что и в листинге 14.1, но с использованием оператора **EXECUTE IMMEDIATE**.

```
EXECUTE IMMEDIATE SELECT name from EMP where dept = 1  
EXECUTE IMMEDIATE SELECT name from EMP where dept = 2
```

Листинг 14.2. Динамический SQL-оператор на встроенном С, использующий EXECUTE IMMEDIATE

На многих динамических языках программирования, таких как PHP или Ruby on Rails, в которых SQL выполняется динамически, программисты записывают одинаковые SQL-операторы с разными значениями полей следующим образом:

```
SELECT lastname, salary FROM employee where firstname = 'Raul'  
SELECT lastname, salary FROM employee where firstname = 'Jin'  
...
```

В этом примере операторы идентичны, за исключением значения столбца `firstname`. DB2 воспринимает эти два динамических SQL-оператора как разные, а

потому при запуске готовит и затем выполняет их отдельно. Лишняя подготовка одинаковых операторов несколько раз может сократить производительность, поэтому до выхода DB2 9.7 рекомендовалось записывать операторы следующим образом:

```
SELECT lastname, salary FROM employee where firstname = ?
```

Вопросительный знак (?) в операторе называют **маркером параметров**. Используя маркеры параметров, программисты могут подготовить оператор один раз, а затем задать операторы EXECUTE с разными значениями маркера параметров.

new in V9.7

В DB2 9.7 была представлена технология под названием **концентратор операторов**, в которой все операторы, отличающиеся только значениями полей, автоматически сливаются в единый оператор с маркерами параметров, а затем выполняются операторы EXECUTE с разными значениями. Концентратор операторов не может определить, когда не следует сливать некоторые операторы, например, если некоторые разделы добавлены специально для воздействия на оптимизатор DB2.

С точки зрения производительности, статический SQL обычно работает лучше, чем динамический, поскольку план доступа в статическом SQL формируется при преподавании, а не при выполнении. Однако в средах с высоким уровнем активности, в частности операций INSERT и DELETE, определенная при преподавании статистика может терять актуальность, а потому план доступа статического SQL может не быть оптимальным. В таком случае лучше выбрать динамический SQL при условии, что для сбора текущей статистики часто выполняется команда RUNSTATS.

Примечание.

Многие считают, что встроенный SQL может быть только статическим. На самом деле он может быть как статическим, так и динамическим.

14.3.3 CLI и ODBC

Интерфейс уровня вызовов (Call Level Interface, CLI) был разработан компанией компаниями X/Open и SQL Access Group. Этот интерфейс был спецификацией вызываемого интерфейса SQL для разработки портативных приложений на C/C++ независимо от поставщика СУРБД. На основе предварительного проекта интерфейса уровня вызовов X/Open корпорация Microsoft разработала **открытые средства связи с базами данных (Open Database Connectivity, ODBC)**, а позже международный стандарт ISO CLI принял большинство спецификаций интерфейса уровня вызовов X/Open. CLI DB2 основан на ODBC и Международном стандарте для SQL/CLI (см. рис. 14.3).

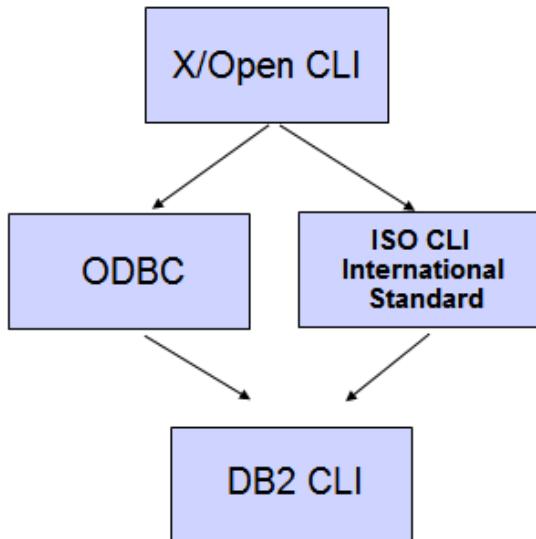


Рисунок 14.3. CLI DB2 основан на ODBC и Международном стандарте CLI ISO

CLI DB2 соответствует ODBC 3.51 и может использоваться в качестве драйвера ODBC при загрузке диспетчером драйверов ODBC. Рис. 14.4 поможет наглядно представить поддержку ODBC, предоставляемую CLI DB2.

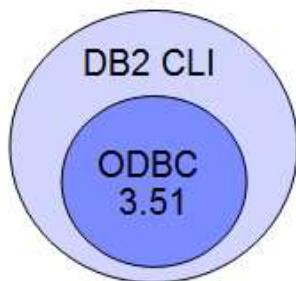


Рисунок 14.4. CLI DB2 соответствует ODBC 3.51

CLI/ODBC имеет следующие характеристики:

- Программный код легко портируется между продуктами нескольких поставщиков СУРБД.
- В отличие от встроенного SQL, нет необходимости использовать компилятор предварительного прохода или переменные хоста.
- Выполнение динамического SQL.
- Высокая распространенность.

Для выполнения приложения CLI/ODBC достаточно иметь драйвер CLI DB2. Этот драйвер устанавливается с любым из перечисленных клиентов и драйверов, доступных для бесплатной загрузки и использования на сайте www.ibm.com/db2/express:

- Клиент IBM Data Server
- Клиент IBM Data Server Runtime
- Драйвер IBM Data Server для ODBC и CLI.

Для **разработки** приложения CLI/ODBC необходимы драйвер CLI DB2 и соответствующие библиотеки. Они содержатся только в клиенте IBM Data Server.

Рассмотрим следующий пример, чтобы лучше разобраться с процессом установки драйвера CLI DB2 для приложений. На рис. 14.5 показано три разных компьютера: один в Индонезии, второй в Бразилии, а третий — в Канаде.

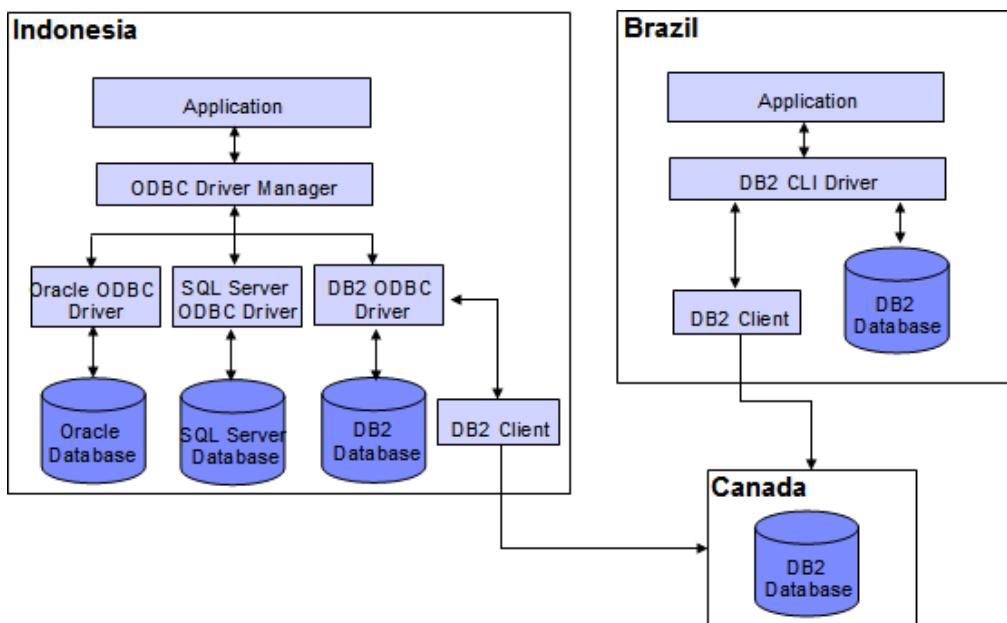


Рисунок 14.5. Пример сценария CLI/ODBC DB2

На рисунке проиллюстрировано два случая:

Для левой части рисунка предположим, что на компьютере в Индонезии выполняется *приложение ODBC*, которое может работать с любой СУРБД, например, Oracle, Microsoft SQL Server или DB2. Диспетчер драйверов ODBC загрузит соответствующий драйвер ODBC в зависимости от базы данных, к которой получают доступ. Если приложение получает доступ к DB2 в Канаде, соединение должно проходить через клиент DB2 с компонентами для удаленного подключения.

Для правой стороны рисунка предположим, что *приложение CLI* выполняется на компьютере в Бразилии. Это CLI-приложение, поскольку оно может использовать определенные функции, недоступные в ODBC, а также приложение будет работать только с базой данных DB2. Приложение CLI пройдет через драйвер CLI DB2. Приложение может подключиться к локальной базе данных DB2 в Бразилии. Если понадобится соединение с удаленной базой данных в Канаде, приложение пройдет через клиент DB2.

Последнее, на что необходимо обратить внимание в этом разделе, — сравнение приложения CLI/ODBC и динамического приложения на встроенным SQL C. Сравнение проиллюстрировано на рис. 14.6.

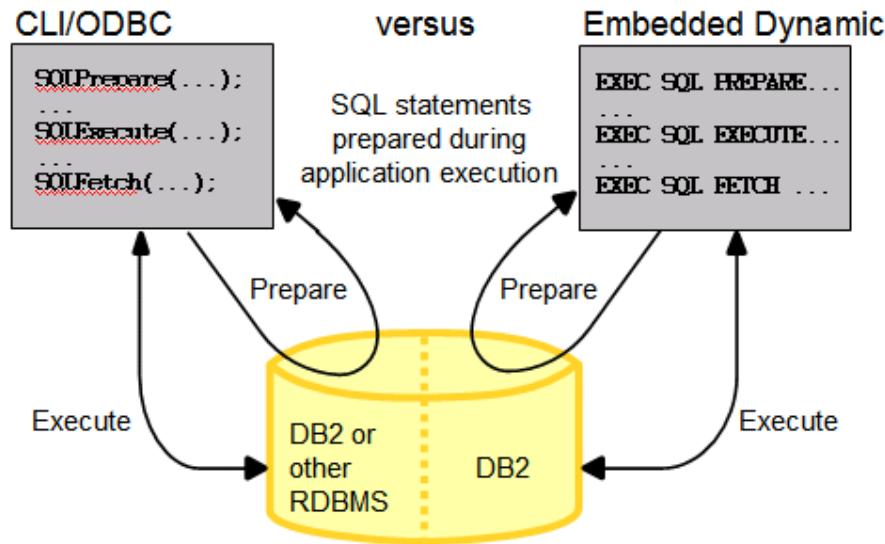


Рисунок 14.6. Сравнение приложения CLI/ODBC и динамического приложения на встроенным SQL C

Как показано на рис. 14.6, единственным отличием между приложением CLI/ODBC и динамическим приложением на встроенным SQL C является то, что для CLI/ODBC используется портативный программный код с возможностью доступа к другим СУРБД с помощью простой замены строки соединения, а в динамической версии встроенного SQL C можно программировать конкретные элементы под DB2. Безусловно, еще одним отличием является способ вызова различных функций для **PREPARE** и **EXECUTE**.

14.3.4 JDBC, SQLJ и pureQuery

Соединение с базами данных на Java (Java Database Connectivity, JDBC) — это программный API Java, определяющий стандарты средств для работы с базами данных и доступа к ним. Программный код JDBC легко портируется между продуктами нескольких поставщиков СУРБД. Обычно в код необходимо вносить только изменения, касающиеся выбора драйвера JDBC для загрузки и строки соединения. JDBC использует только динамический SQL и пользуется большой популярностью.

SQLJ — это стандарт встраивания SQL в Java-программы. Он используется преимущественно со статическим SQL, хотя совместим с JDBC (см. рис. 14.7). Хотя обычно он является более компактным, чем программы JDBC, и обеспечивает более высокую производительность, этот стандарт не получил широкого применения. Программы SQLJ необходимо выполнять через препроцессор (транслятор SQLJ), прежде чем их можно будет скомпилировать.

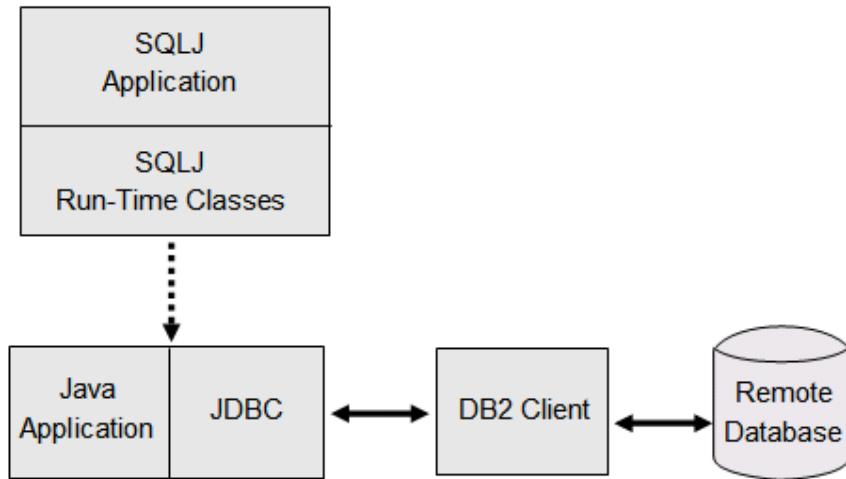


Рисунок 14.7. Отношения между приложениями SQLJ и JDBC

На рис. 14.7 клиент DB2 может требоваться или нет, в зависимости от используемого типа драйвера JDBC, как описано далее в этом разделе.

pureQuery — это разработанный компанией IBM подключаемый модуль на базе Eclipse для управления реляционными данными как объектами. Доступный с 2007 года, модуль pureQuery может автоматически генерировать код для установки объектно-реляционного отображения (object-relational mapping, ORM) между объектно-ориентированным программным кодом и объектами реляционной базы данных. Для начала вы создаете проект Java в Optim Development Studio (ODS) и подключаетесь к базе данных DB2, а затем ODS обнаруживает все объекты базы данных. С помощью графического интерфейса пользователя ODS можно выбрать таблицу, а затем генерировать код pureQuery, который трансформирует все имеющиеся в таблице реляционные объекты в объекты Java. Генерируется код для создания соответствующих SQL-операторов и родительских объектов Java, формирующих эти операторы. Сгенерированные объекты Java и содержащиеся в них SQL-операторы поддаются дальнейшей настройке. С помощью pureQuery во время выполнения можно выбрать режим статического или динамического SQL. pureQuery поддерживает Java, и .NET.

14.3.4.1 Драйверы JDBC и SQLJ

Хотя существует несколько типов драйверов JDBC, в частности типы 1, 2, 3 и 4, типы 1 и 3 используются нечасто, и их поддержка в DB2 считается устаревшей. Как описано в одной из последующих частей этой главы, существует два драйвера типа 2, но один из них также считается устаревшим.

Типы 2 и 4 поддерживаются в DB2 (см. Таблицу 14.2). Для драйверов типа 2 требуется наличие установленного клиента DB2, поскольку они используют клиент для установки связи с базой данных. Драйвер типа 4 — это «чистый» клиент Java, поэтому клиент DB2 не требуется, но этот драйвер необходимо установить на том компьютере, на котором выполняется приложение JDBC.

| Тип драйвера | Имя драйвера | Пакет | Поддержка | Минимальный требуемый уровень SDK для Java |
|---------------|---|-------------------------|--------------------------------|--|
| Тип 2 | Драйвер DB2 JDBC типа 2 для Linux, UNIX и Windows (устарело*) | db2java.zip | JDBC 1.2 и JDBC 2.0 | 1.4.2 |
| Тип 2 и тип 4 | Драйвер IBM Data Server для JDBC и SQLJ | db2jcc.jar и sqlj.zip | соответствующий JDBC 3.0 | 1.4.2 |
| | | db2jcc4.jar и sqlj4.zip | JDBC 4.0 и более старые версии | 6 |

Таблица 14.2. Драйверы DB2 JDBC и SQLJ

* Термин «устарело» означает, что решение всё ещё поддерживается, но уже не улучшается.

Как уже упоминалось, а также демонстрировалось в *Таблице 14.2*, существует два разных драйвера типа 2; однако драйвер DB2 JDBC типа 2 для Linux, UNIX и Windows с именем файла db2java.zip является устаревшим.

При установке сервера DB2, клиента DB2 или драйвера IBM Data Server для JDBC и SQLJ файлы db2jcc.jar и sqlj.zip, соответствующие JDBC 3.0, автоматически добавляются к пути к классам.

14.3.5 Связывание и внедрение объектов, базы данных (Object Linking and Embedding, Database, OLE DB)

OLE DB — это набор интерфейсов, предоставляющий доступ к данным из различных источников. Он разрабатывался как замена ODBC, но был расширен для поддержки более широкого набора источников, в том числе нереляционных баз данных, например объектно-ориентированных баз данных и электронных таблиц. Реализация OLE DB осуществляется с применением технологии объектной модели программных компонентов (Component Object Model, COM).

Пользователи OLE DB могут получить доступ к базам данных DB2 с помощью решения OLE DB Provider для DB2 от компании IBM. Этот драйвер доступа имеет следующие характеристики:

- Имя драйвера доступа: IBMDADB2
- Поддерживает уровень 0 спецификации драйвера доступа OLE DB, включая некоторые дополнительные интерфейсы уровня 1.
- Соответствует версии 2.7 и более новым версиям спецификации OLE DB от Microsoft.
- Необходимо установить клиент IBM Data Server Client с компонентами доступа к данным от Microsoft (Microsoft Data Access Components, MDAC).

- Если IBMDADB2 не задан явно, по умолчанию будет использоваться драйвер OLE DB от Microsoft (MSDASQL). MSDASQL позволяет клиентам, использующим OLE DB, получать доступ к источникам данных на серверах SQL разработки не компании Microsoft с драйвером ODBC, но при этом не гарантируется полная функциональность драйвера OLE DB.

14.3.6 ADO.NET

.NET Framework — это представленная корпорацией Microsoft замена технологии COM. С помощью .NET Framework можно программировать приложения .NET более чем на сорока языках программирования, самые популярные из которых — C# и Visual Basic .NET.

Библиотека классов .NET Framework предоставляет стандартные блоки, из которых строятся приложения .NET. Эта библиотека классов не зависит от языка программирования и предоставляет средства связи со службами операционной системы и сервера приложений. Приложение .NET (независимо от языка программирования) компилируется на промежуточный язык (Intermediate Language, IL), являющийся типом байт-кода.

Общеязыковая среда исполнения (Common Language Runtime, CLR) — это «сердце» .NET Framework, которое на ходу компилирует программный код на IL, а затем выполняет его. Выполняя скомпилированный код на IL, CLR активирует объекты, проверяет их уровень защиты, распределяет память, выполняет их и проводит очистку их памяти после выполнения.

Проведем аналогию с принципами работы Java. Программа на Java может работать на разных платформах с минимальными изменениями или вовсе без них: один язык, но множество платформ. В случае .NET программа, написанная на одном из сорока поддерживаемых языков программирования, может работать на одной платформе (Windows) с минимальными изменениями или вовсе без них: множество языков, но одна платформа.

ADO.NET — это средство предоставления поддержки доступа к данным в .NET Framework. ADO.NET поддерживает доступ с подключением и без. Ключевым компонентом доступа к данным без подключения в ADO.NET является класс **DataSet**, экземпляры которого функционируют как кэш базы данных, хранящийся в памяти приложения.

Для доступа с подключением и без, приложения используют базы данных посредством так называемого **провайдера данных**. Различные продукты для работы с базами данных включают собственные провайдеры данных .NET, в том числе и DB2 для Windows.

Провайдер данных .NET характеризуется внедрением следующих базовых классов:

- Connection (соединение): устанавливает соединение с базой данных и управляет им.
- Command (команда): выполняет SQL-оператор в базе данных.
- DataReader (блок считывания данных): выполняет считывание и возвращает набор результатов из базы данных.

- DataAdapter (адаптер данных): привязывает экземпляр DataSet к базе данных. Через экземпляр DataAdapter DataSet может считывать и записывать данные таблиц базы данных.

Три провайдера данных, которые могут работать с DB2, показаны в *Таблице 14.3*.

| Провайдер данных | Характеристики |
|--|--|
| Провайдер данных .NET ODBC (не рекомендуется) | <ul style="list-style-type: none"> Выполняет вызовы ODBC к источнику данных DB2 с помощью драйвера CLI DB2. Имеет определенную поддержку ключевых слов и ограничения, как у драйвера CLI DB2. Может использоваться с .NET Framework версии 1.1, 2.0 или 3.0. |
| Провайдер данных .NET OLE DB (не рекомендуется) | <ul style="list-style-type: none"> Использует драйвер IBM DB2 OLE DB (IBMDADB2). Имеет определенную поддержку ключевых слов и ограничения, как у драйвера OLE DB DB2. Может использоваться только с .NET Framework версии 1.1, 2.0 или 3.0. |
| Провайдер данных .NET DB2 (рекомендуется) | <ul style="list-style-type: none"> Расширяет поддержку интерфейса ADO.NET в DB2. Провайдер под управлением DB2 внедряет такой же набор стандартных классов и методов ADO.NET. Определяется в области имен IBM.DATA.DB2. Устанавливается с любым из перечисленных продуктов: <ul style="list-style-type: none"> - Драйвер Data Server для ODBC, CLI и .NET - Клиент IBM Data Server Runtime - DB2 Data Server |

Таблица 14.3. Провайдеры данных ADO.NET

14.3.7 PHP

Препроцессор гипертекста PHP (PHP) — это независимый от платформы язык сценариев с открытым исходным кодом, созданный для разработки веб-приложений. PHP-код встраивается в HTML и обычно выполняется на веб-сервере, который на основе такого кода создает выводимые веб-страницы.

PHP — это язык модульного типа. Доступную функциональность можно настроить с помощью расширений. Одними из наиболее популярных являются расширения PHP для доступа к базам данных. Компания IBM поддерживает доступ к базам данных DB2 через два расширения:

- **ibm_db2**: расширение ibm_db2 предоставляет процедурный интерфейс программирования приложений для создания, чтения, обновления и записи операций с базой данных, в дополнении к широкому доступу к метаданным базы данных. Его можно скомпилировать с помощью PHP 4 или PHP 5.
- **pdo_ibm**: pdo_ibm — это драйвер для расширения PHP Data Objects (PDO, объекты данных PHP), предоставляющий доступ к базе данных DB2 через стандартный объектно-ориентированный интерфейс баз данных, представленный в PHP 5.1. Его можно скомпилировать непосредственно в библиотеках DB2.

Расширения и драйверы PHP бесплатно доступны в репозитории PECL по адресу <http://pecl.php.net/>, а также в пакете клиента IBM Data Server. ibm_db2 и pdo_ibm основаны на уровне CLI DB2 от IBM.

Компания IBM также сотрудничает с Zend Technologies Inc. для поддержки ZendCore — бесплатного готового к работе инструментария для разработки на PHP и в DB2 Express-C. Пакет ZendCore включает библиотеки PHP, веб-сервер Apache и DB2 Express-C. Чтобы скачать ZendCore, посетите страницу <http://www.ibm.com/software/data/info/zendcore>

14.3.8 Ruby on Rails

Ruby — это объектно-ориентированный язык программирования с открытым программным кодом. Rails — это веб-оболочка, созданная с помощью Ruby. Ruby on Rails — это идеальное средство для разработки веб-приложений с поддержкой баз данных. Эта новейшая технология основана на архитектуре «модель, представление, контроллер» (Model, View, Controller — MVC) и следует принципам гибкой разработки программного обеспечения.

Rails не требует особых форматов файлов или интегрированных сред разработки (integrated development environment, IDE); работу можно начать в текстовом редакторе. Однако доступны разнообразные IDE с поддержкой Rails, такие как RadRails — среда Rails для Eclipse. С более подробной информацией о RadRails можно ознакомиться по адресу <http://www.radrails.org/>.

DB2 поддерживает Ruby 1.8.5 и более новые версии, а также Ruby on Rails 1.2.1 и более новые версии. Пакет IBM_DB gem включает драйвер IBM_DB Ruby и адаптер Rails, позволяющий работать с DB2 и основанный на уровне CLI. Этот пакет необходимо установить вместе с клиентом IBM Data Server. Для установки драйвера и адаптера IBM_DB можно воспользоваться пакетом Ruby или подключаемым модулем Rails.

14.3.9 Perl

Perl — это популярный интерпретируемый язык программирования, доступный бесплатно для многих операционных систем. Он использует динамический SQL и идеально подходит для разработки прототипов приложений.

Perl предоставляет стандартный модуль под названием «интерфейс базы данных» (Database Interface, DBI) для доступа к различным базам данных. Его можно загрузить по адресу <http://www.perl.com>. Этот модуль «общается» с драйверами баз данных от различных поставщиков. В случае DB2 — это драйвер DBD::DB2, который можно загрузить по адресу <http://www.ibm.com/software/data/db2/perl>.

14.3.10 Python

Python — это динамический язык программирования, часто используемый для создания сценариев. Python ставит основное ударение на чёткость программного кода и поддерживает разнообразные парадигмы программирования, в том числе процедурное, объектно-ориентированное, аспектно-ориентированное, функциональное и метапрограммирование. Python идеально подходит для быстрой разработки приложений.

В табл. 14.4 показаны имеющиеся расширения для доступа к базам данных DB2 из приложений на Python.

| Расширение | Описание |
|-------------------|---|
| ibm_db | Определяется компанией IBM Предоставляет наивысший уровень поддержки расширенных функций. Дает возможность создавать SQL-запросы, вызывать хранимые процедуры, использовать pureXML и получать доступ к информации метаданных. |
| ibm_db_dbi | Выполняет Python Database API Specification v2.0 (API-спецификация баз данных языка Python, версия 2.0). Не имеет некоторых расширенных функций, поддерживаемых в API ibm_db. Имея приложение с драйвером, поддерживающим Python Database API Specification v2.0, можно легко перейти на ibm_db. API ibm_db и ibm_db_dbi содержатся в одном пакете. |
| ibm_db_sa | Поддерживает SQLAlchemy — популярный Python-инструментарий SQL и ORM с открытым исходным кодом. |

Таблица 14.4. IBM Data Server — расширения Python

14.4 XML и технология pureXML в DB2

XML — это технология, лежащая в основе инструментов и технологий Web 2.0, а также **SOA**. Компания IBM быстро осознала важность XML и инвестировала значительные средства в разработку технологии pureXML, предоставляющей расширенные возможности хранения XML-документов в DB2.

Представленный в 2006 году сервер данных DB2 9 является гибридным: он дает возможность сохранять в исходном формате как реляционные, так и иерархические данные. Предыдущие версии DB2 и прочие представленные на рынке серверы данных могли хранить XML-документы, однако метод хранения, используемый в DB2 9, повысил производительность и гибкость. При использовании представленной в DB2 9 технологии pureXML документы XML сохраняются внутри, в проанализированном иерархическом виде, с древообразной структурой; благодаря этому значительно расширились возможности работы с XML-документами. В новых выпусках DB2, например DB2 9.5 и DB2 9.7, поддержка pureXML была еще больше улучшена. В главе 15 «Технология pureXML в DB2» этот вопрос рассматривается более подробно.

14.5 Веб-службы

В качестве простого определения веб-службы можно рассматривать как функции, запускаемые в сети, при использовании которых не нужно знать, на каком языке программирования они написаны, в какой операционной системе и где именно они будут работать. Веб-службы дают одному приложению возможность обмениваться данными с другим приложением, применяя открытые стандартизированные протоколы вычислительной сети на базе XML. Это показано на рис. 14.8.

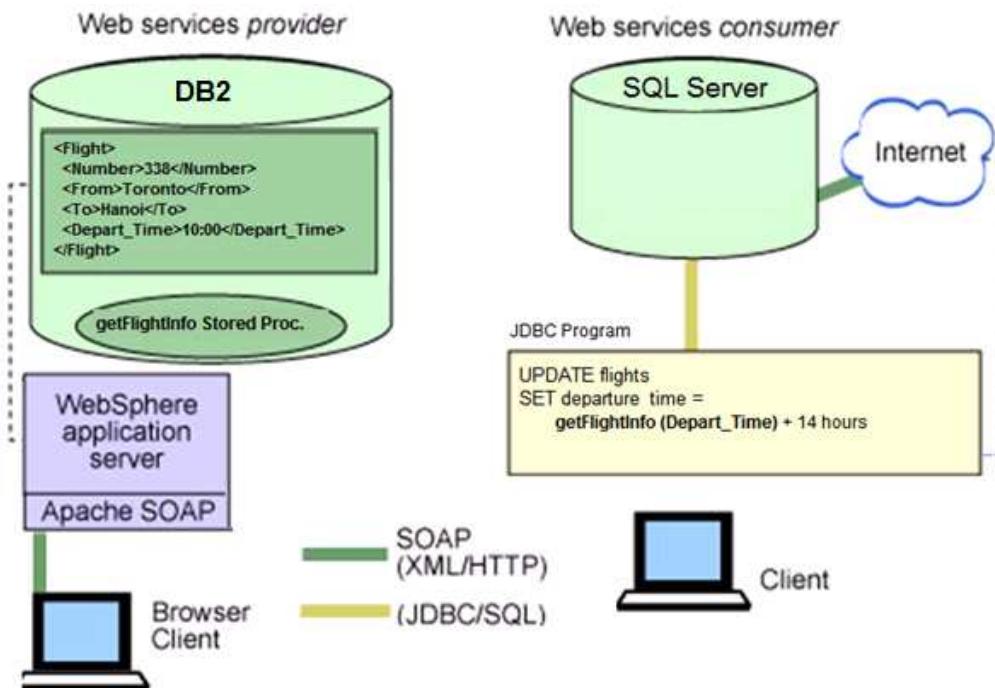


Рисунок 14.8. Пример работы веб-службы

Предположим, что в левой части рисунка показана система вымышленной авиакомпании (Air Atlantis), которая использует DB2 на Linux и сохраняет информацию об авиарейсах в формате XML в базе данных DB2. В правой части

рисунка показана система другой вымышленной авиакомпании (Air Discovery), которая использует SQL Server в ОС Windows. Теперь предположим, что Air Atlantis и Air Discovery подписали договор о сотрудничестве, согласно которому компании хотят совместить информацию о расписаниях полетов и ценах, чтобы скоординировать авиарейсы. Выполнить такой обмен информацией будет довольно сложно, поскольку компании используют разные операционные системы (Linux, Windows) и серверы данных (DB2, SQL Server). Если Air Atlantis изменит расписание авиарейса Торонто — Пекин, как Air Discovery сможет автоматически откорректировать расписание своих авиарейсов для пересадки из Пекина в Шанхай? Ответ прост — с помощью веб-служб. Air Atlantis может разгласить часть информации об авиарейсах, создав **веб-службу данных**, которая возвращает результаты хранимой процедуры (хранимой процедуры `getFlightInfo`) с информацией об авиарейсах из базы данных DB2. Веб-служба **данных** — это веб-служба, основанная на информации базы данных. Когда эта веб-служба данных будет развернута на таком сервере приложений, как WebSphere Application Server, клиенты или партнеры, например Air Discovery, смогут с помощью браузера легко получить доступ к информации об авиарейсах компании Air Atlantis. В этом примере Air Atlantis действует как провайдер веб-службы, поскольку разрабатывает веб-службу и предоставляет к ней доступ; с другой стороны, компания Air Discovery действует как клиент веб-службы, поскольку пользуется этой веб-службой.

Air Discovery может также запустить эту веб-службу из собственного приложения JDBC, чтобы выполнять расчеты, использующие данные своей базы данных SQL Server. К примеру, если авиарейс Торонто — Пекин длится в среднем 12 часов, Air Discovery может вычислить время для пересадочного рейса Пекин — Шанхай, добавив ко времени вылета авиарейса Air Atlantis из Торонто длительность полета и несколько резервных (буферных) часов. Количество буферных часов можно сохранить в базе данных SQL Server в системе компании Air Discovery, и простое уравнение для приложения JDBC будет выглядеть так:

| | | | | | | |
|---|---|---|---|---|---|---|
| Air Discovery Flight Departure time (Beijing to Shanghai) | = | Air Atlantis Flight Departure time (Toronto to Beijing) | + | Air Atlantis Flight Duration (Toronto to Beijing) 12 hours average | + | Air Atlantis Flight Buffer time (Toronto to Beijing) 2 hours |
|---|---|---|---|---|---|---|

Если компания Air Atlantis изменит время отправки рейса, эта информация автоматически попадет в систему Air Discovery после запуска веб-службы.

14.6 Административные API

DB2 предоставляет множество административных API, которыми разработчики могут воспользоваться для построения собственных утилит или инструментов. К примеру, чтобы создать базу данных, можно запустить API `sqlcrea`; чтобы запустить экземпляр, можно воспользоваться API `db2InstanceStart`; чтобы импортировать данные в таблицу, можно воспользоваться API `db2Import`. С полным списком можно ознакомиться в информационном центре DB2. URL-адрес информационного центра DB2 см. в разделе *Ресурсы*.

14.7 Прочие разработки

Некоторые пользователи DB2 Express-C также применяют продукты сторонних разработчиков, например MS Excel и MS Access, для создания простых подключаемых к DB2 форм. В этом разделе мы опишем использование таких продуктов с DB2 Express-C.

Продукт DB2 Express-C также доступен для Mac OS X, и вы можете использовать DB2 для разработки приложений баз данных непосредственно на платформе Mac. Особенно это может заинтересовать сообщество Ruby on Rails, которое предпочитает платформу Mac.

14.7.1 Работа с Microsoft Access и Microsoft Excel

Microsoft Excel и Microsoft Access — это популярные инструменты генерирования отчетов, создания форм и разработки простых приложений, благодаря которым данные приобретают элементы бизнес-аналитики. DB2 может легко взаимодействовать с этими инструментами. Администратор баз данных может хранить данные компании на безопасном сервере DB2, а обычные пользователи будут получать доступ к таким данным и генерировать отчеты с помощью Access или Excel. Это показано на Рисунке 14.9.

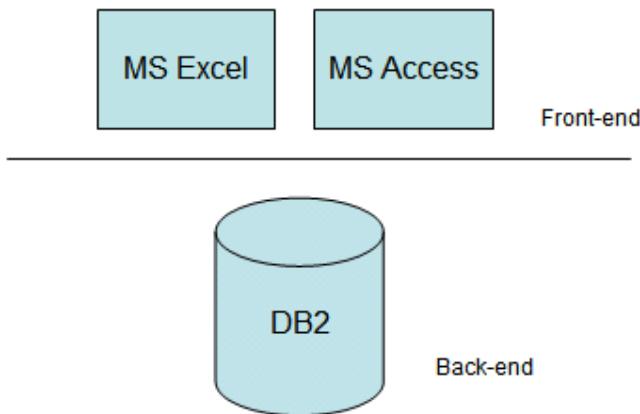


Рисунок 14.9. Работа с Excel, Access и DB2

Как показано на рисунке, Excel и Access используются для разработки интерфейской прикладной программы, а DB2 обеспечивает безопасность данных, надежность и высокую производительность в качестве серверной части приложения. Благодаря централизации всех данных в DB2 создается упрощенная модель хранения данных.

В случае Excel проще всего получить доступ к данным DB2, воспользовавшись драйвером OLE DB, например OLE DB Provider для DB2 от IBM. Эти драйверы включены в пакет установки бесплатного клиента IBM Data Server, который можно загрузить на веб-сайте DB2 Express-C по адресу www.ibm.com/db2/express. После установки необходимо выбрать источник данных с соответствующим провайдером OLE DB в меню MS Excel. Выберите **Данные → Импорт внешних данных → Импортировать данные**. Дальнейшие шаги описаны в статье *IBM DB2 Universal Database and the Application Developer... for Beginners* (Универсальная база данных

DB2 от IBM и Разработчик приложений Microsoft Excel... для начинающих) [1]. Подробную информацию см. в разделе *Справочные материалы*.

При использовании Microsoft Access, также необходимо установить что-либо из перечисленного ниже:

- Клиент IBM Data Server.
- Драйвер IBM Data Server для ODBC, CLI и .NET.
- Драйвер IBM Data Server для ODBC и CLI.

Драйвер IBM Data Server для ODBC, CLI и .Net и драйвер IBM Data Server для ODBC и CLI также известны как драйвер IBM DB2 ODBC, который соответствует драйверу CLI DB2. Этот драйвер используется для подключения Access к DB2. Установив драйвер, создайте проект Access 2007 и выберите вариант *База данных ODBC* на вкладке *Внешние данные* на ленте *Работа с таблицами*. Дальнейшие шаги описаны в статье *DB2 9 and Microsoft Access 2007 Part 1: Getting the Data...* (DB2 9 и Microsoft Access 2007. Часть 1: Получение данных...) [2]. При использовании в Microsoft Access **связанных таблиц** данные доступны пользователям Access 2007, но хранятся на сервере данных DB2.

Для версий Access до 2007 процедура установки слегка отличается. С ней можно ознакомиться в статье *Use Microsoft Access to interact with your DB2 data* (Использование Microsoft Access для взаимодействия с данными DB2) [3]. Подробную информацию см. в разделе *Справочные материалы*.

14.8 Инструменты разработки

Microsoft Visual Studio и Eclipse — наиболее популярные из используемых сегодня интегрированных сред разработки (Integrated Development Environments, IDE). С DB2 отлично работают обе среды IDE.

Для Microsoft Visual Studio DB2 предоставляет надстройку Visual Studio, чтобы после установки инструменты IBM добавлялись в меню Visual Studio. Таким образом, разработчикам не нужно переключаться на другие инструменты, чтобы работать с базами данных DB2. Надстройку Visual Studio можно загрузить на веб-сайте DB2 Express-C по адресу www.ibm.com/db2/express.

В отношении Eclipse компания IBM выпустила IBM Data Studio — бесплатный инструмент на базе Eclipse, позволяющий разрабатывать сценарии SQL и Query, хранимые процедуры, UDF и веб-службы. Поскольку этот инструмент основан на платформе Eclipse, многие разработчики могут расширить свои знания для работы с ним.

14.9 Образцы программ

Чтобы помочь вам разобраться с программированием на разных языках с DB2 в качестве сервера данных, мы предоставили образцы приложений, доступные в установочном пакете сервера DB2 в каталоге SQLLIB\samples. На Рисунке 14.10 ниже показано несколько образцов программ, предоставленных в DB2 на платформе Windows.

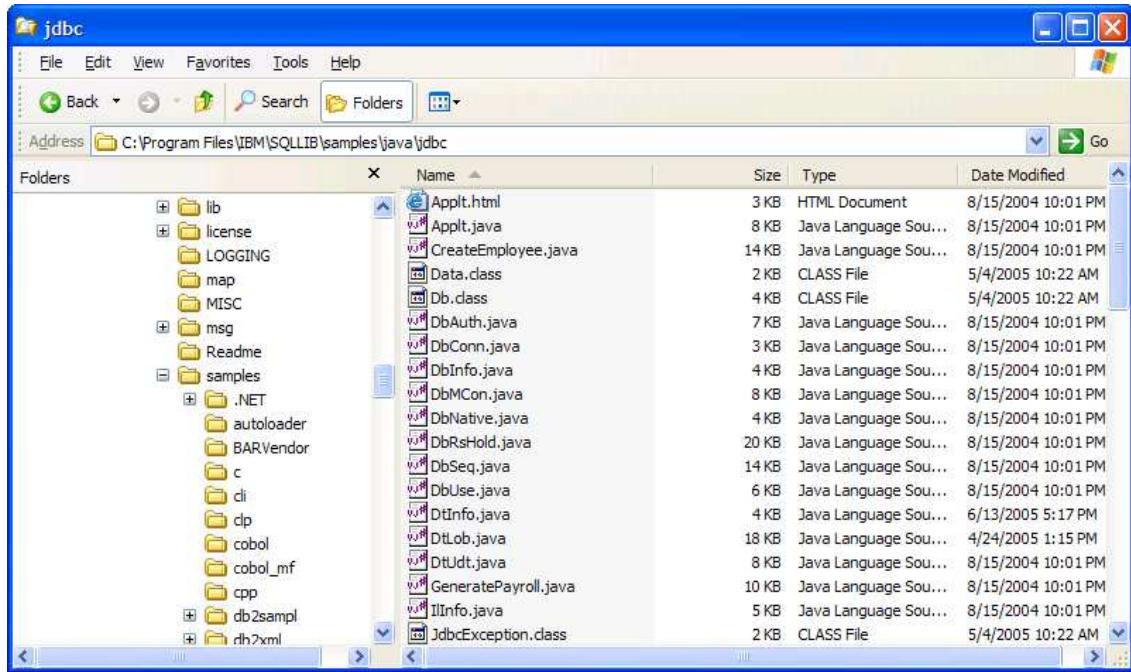


Рисунок 14.10. Образцы программ, поставляемых с DB2

14.10 Краткий обзор

В этой главе мы рассмотрели, как DB2 предоставляет гибкость программирования приложений баз данных в базе данных на сервере или через приложения на стороне клиента с подключением к серверу данных DB2.

Среди рассмотренных вопросов о приложениях на стороне сервера — хранимые процедуры, пользовательские функции и триггеры.

Для стороны клиента мы рассмотрели множество интерфейсов и методов программирования, разрешенных для разработки приложений DB2, в очередной раз продемонстрировав исключительную гибкость и возможности DB2 в качестве сервера данных.

15

Глава 15. Технология pureXML в DB2

В этой главе рассматривается pureXML — новая технология, представленная в DB2 9 для поддержки хранения данных XML в исходном формате. Многие рассмотренные в этой главе примеры и понятия взяты из Красной книги IBM: *DB2 9: pureXML. Обзор и начало работы*. Более подробную информацию об этой книге см. в разделе «Ресурсы». На Рисунке 15.1 обозначено, какая именно часть «общей картины» DB2 рассматривается в этой главе.

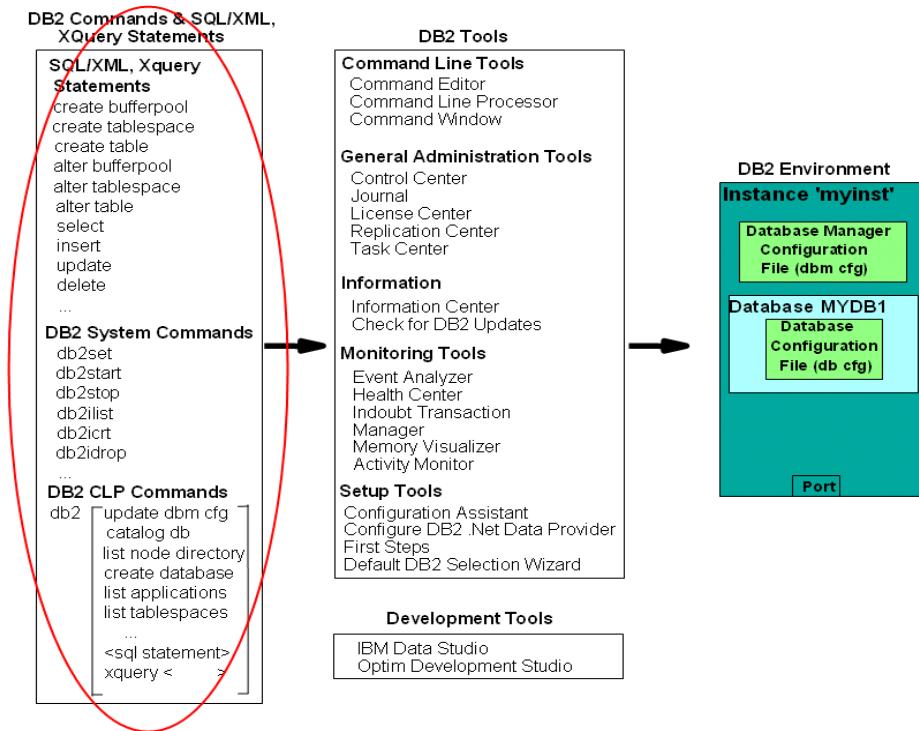


Рисунок 15.1. Общий обзор DB2: команды DB2, SQL/XML и XQuery

Примечание.

Чтобы получить более подробную информацию о технологии pureXML, просмотрите видео: <http://www.channeldb2.com/video/video/show?id=807741:Video:4382>

15.1 Использование XML в базах данных

Документы XML могут храниться в текстовых файлах, XML-репозиториях или базах данных. Есть две главные причины того, почему многие компании предпочитают хранить их в базах данных:

- Управление большими объемами XML-данных — проблемы уровня баз данных. XML — это те же данные, но в ином общем формате. Причины, по которым реляционные данные следует хранить в базах данных, относятся и к данным XML: базы данных предоставляют эффективные возможности поиска и вывода данных, мощную поддержку для сохранности данных, резервное копирование и восстановление, поддержку транзакций, производительность и масштабируемость.
- Интеграция: храня реляционные документы вместе с документами XML, можно интегрировать новые XML-данные с существующими реляционными данными и объединить SQL с XPath или XQuery в одном запросе. Более того, реляционные данные можно публиковать как данные XML, или наоборот. Благодаря интеграции базы данных лучше поддерживают веб-приложения, SOA и веб-службы.

15.2 Базы данных XML

Существует два типа баз данных для хранения данных XML:

- Базы данных с поддержкой XML.
- Истинные базы данных XML.

15.2.1 Базы данных с поддержкой XML

База данных с поддержкой XML использует реляционную модель в качестве основной модели для хранения данных XML. Следовательно, требуется либо преобразование модели данных XML (иерархической) в реляционную модель данных, либо хранение XML-данных как большого символьного объекта. Хотя эта технология считается устаревшей, её всё ещё применяют многие поставщики баз данных. На Рисунке 15.2 более подробно объяснены два варианта баз данных с поддержкой XML.

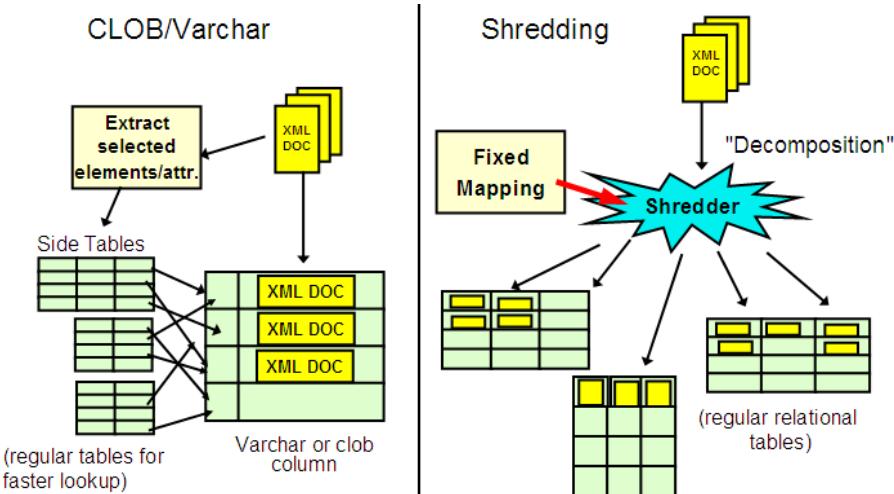


Рисунок 15.2. Два варианта хранения XML-данных в базах данных с поддержкой XML

В левой части *рис. 15.2* проиллюстрировано хранение XML-документов в базе данных по методу **CLOB и varchar**. При использовании этого метода XML-документ сохраняется как непроанализированная строка в столбце CLOB или varchar базы данных. Если XML-документ хранится в виде строки, для извлечения части документа XML программе придется извлекать целую строку и проводить её синтаксический анализ, чтобы обнаружить нужную часть. Синтаксический анализ можно рассматривать как построение дерева XML-документа в памяти для навигации по этому дереву. Этот метод потребляет много памяти и имеет невысокую гибкость.

Другой вариант для баз данных с поддержкой XML называется **нарезкой или декомпозицией** и показан в правой части *Рисунка 15.2*. При использовании этого метода весь XML-документ разделяется на меньшие части, которые хранятся в таблицах. Этот метод практически принудительно подгоняет XML-документ, основанный на иерархической модели, под реляционную модель. Этот метод имеет низкую гибкость, поскольку если документ XML изменяется, такие изменения сложно внести в соответствующие таблицы, и понадобится создание большого числа дополнительных таблиц. Этот метод также негативно сказывается на производительности: если нужно восстановить исходный XML-документ, необходимо выполнить ресурсоемкую операцию объединения SQL, которая потребляет тем больше ресурсов, чем больше используется таблиц.

15.2.2 Истинные базы данных XML

Истинные базы данных XML используют иерархическую модель XML-данных для внутреннего хранения и обработки XML. Данные сохраняются в том же формате, в котором обрабатываются: не происходит преобразование в реляционную модель, и документы XML не сохраняются в виде непроанализированных строк (CLOB или varchar). При использовании операторов XPath или XQuery они обрабатываются ядром в исходном формате, а не конвертируются в SQL. Именно поэтому такие базы данных называют «истинными» базами данных XML. На данный момент DB2 является единственным коммерческим сервером данных, предоставляющим такую поддержку.

15.3 XML в DB2

На Рисунке 15.3 ниже показано, как реляционные и иерархические данные (XML-документы) совместно хранятся в гибридной базе данных DB2. На рисунке также проиллюстрирован оператор `CREATE TABLE`, с помощью которого была создана таблица `dept`.

```
CREATE TABLE dept (deptID CHAR(8), ..., deptdoc XML);
```

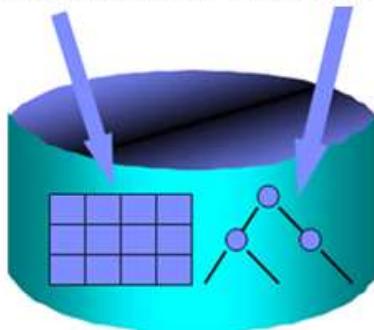


Рисунок 15.3. XML в DB2

Обратите внимание, что определение таблицы использует новый тип данных, XML, в столбце `deptdoc`. Левая стрелка на рисунке указывает на столбец реляционных данных `deptID`, хранящихся в реляционном формате (в таблицах), в то время как столбец XML `deptdoc` хранится в анализируемом иерархическом формате.

На Рисунке 15.4 показано, что теперь в DB2 9 есть четыре пути доступа к данным:

- использование SQL для доступа к реляционным данным;
- использование SQL с расширениями XML (SQL/XML) для доступа к данным XML;
- использование XQuery для доступа к данным XML;
- использование XQuery для доступа к реляционным данным.

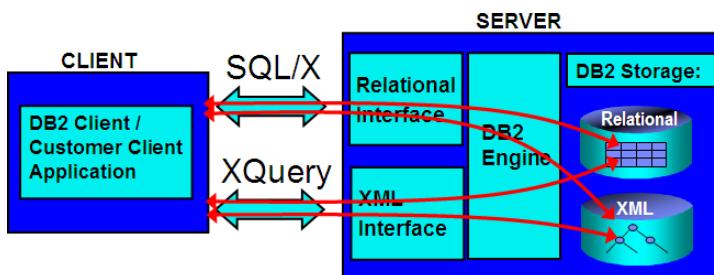


Рисунок 15.4. Четыре способа получения доступа к данным в DB2

Таким образом, в зависимости от предыдущего опыта работы, пользователи, привыкшие к SQL, могут рассматривать DB2 как реляционную СУБД мирового класса с поддержкой XML. С другой стороны, для пользователей, предпочитающих XML, DB2 станет хранилищем XML мирового класса с поддержкой SQL.

Обратите внимание на то, что для описания этой технологии IBM использует термин *pureXML*, а не *истинный XML*. Другие поставщики для хранения XML-документов до сих пор используют устаревшие технологии CLOB/varchar и называют их «истинный XML». Во избежание путаницы компания IBM решила ввести новый термин *pureXML* и защитить его товарным знаком, чтобы никто из поставщиков баз данных или XML не мог использовать этот термин для обозначения каких-либо других технологий. Поддержка технологии *pureXML* предоставляется для баз данных, созданных в формате Unicode или других форматах.

15.3.1 Преимущества технологии pureXML

Технология *pureXML* предоставляет множество преимуществ.

1. Можно беспрепятственно и эффективно использовать инвестиции в реляционные системы с учетом того, что XML-документы хранятся в столбцах таблиц в формате нового типа данных XML.
2. Можно упростить программный код. К примеру, на *рис. 15.5* показан сценарий PHP, написанный с использованием технологии *pureXML* и без. При использовании *pureXML* (меньший блок слева) строки кода сокращаются. Это означает не только то, что программный код менее сложный, но и то, что повышается общая производительность, поскольку нужно обрабатывать и обслуживать меньше строк кода.

```

<?php
$conn = db2_connect($dbname, $dbuser, $dbpass);

/* Insert Customer Documents */

$stmt = db2_prepare($conn, "VALUES (NEXT VALUE FOR
Cid)");
db2_execute($stmt);
list($Cid) = db2_fetch_array($stmt);

$fileContents = file_get_contents
("customers/c1.xml");

$stmt = db2_prepare($conn, "INSERT INTO xmlecustomer
(Cid, Info) VALUES (?, ?)");
if(!db2_execute($stmt, array($Cid, $fileContents)))
echo db2_stmt_errormsg($stmt);

/* Insert Product Documents */

$fileContents = file_get_contents
("products/p1.xml");
$dom = simplexml_load_string($fileContents);

$proid = (string) $dom["pid"];

$stmt = db2_prepare($conn, "INSERT INTO xmlproduct
(Pid, Description) VALUES (?, ?)");
if(!db2_execute($stmt, array($proid,

```

```

db2_execute($stmt);
list($Cid) = db2_fetch_array($stmt);

$fileContents = file_get_contents
("customers/c1.xml");
$dom = simplexml_load_string($fileContents);

$custName = (string) $dom->name;
$custCountry = (string) $dom->addr->country;
$custStreet = (string) $dom->addr->street;
$custCity = (string) $dom->addr->city;
$custProvince = (string) $dom->addr->("prov-state");
$custZip = (string) $dom->addr->pcode-zip";
$custPhone = (string) $dom->phone;

$stmt = db2_prepare($conn, "INSERT INTO sqlicustomer
(Cid, Name, Country, Street, City, Province, Zip,
Phone, Info) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)");
if(!db2_execute($stmt, array($Cid, $custName,
$custCountry, $custStreet, $custCity, $custProvince,
$custZip, $custPhone, $fileContents )));
echo db2_stmt_errormsg($stmt);

/* Insert Product Documents */

$fileContents = file_get_contents
("products/p1.xml");
$dom = simplexml_load_string($fileContents);

$proid = (string) $dom["pid"];

```

Рисунок 15.5. Сложность кода с *pureXML* и без

3. Вносить изменения в схемы проще при использовании XML и технологии *pureXML*. На *Рисунке 15.6* такая повышенная гибкость проиллюстрирована на примере. Рассматривая рисунок, допустим, что использовалась база данных, состоящая из таблиц *Employee* (Сотрудник) и *Department* (Отдел). Обычно при использовании баз данных без поддержки XML, если ваш начальник требует сохранить для каждого сотрудника не один номер телефона (домашний), а два (мобильный), можно добавить дополнительный столбец в таблицу *Employee* и хранить номера мобильных телефонов там. Однако такой способ противоречит правилам нормализации реляционных баз данных. Чтобы не нарушать такие правила, следует создать новую дополнительную таблицу *Phone* (Телефон) и

переместить всю информацию о номерах телефонов туда. Затем туда же можно внести номера мобильных телефонов. Создание новой таблицы *Phone* потребует много ресурсов, не только в связи с необходимостью перенести большой объем существующих данных, но и потому, что все SQL в приложениях необходимо изменить, чтобы сопоставить с новой таблицей.

В левой части рисунка показано, как то же самое можно сделать с помощью XML. Если сотрудник *christine* также имеет номер мобильного телефона, для ввода этой информации можно добавить новый тег. Если сотрудник *michael* не имеет номера мобильного телефона, просто оставляем всё без изменений.

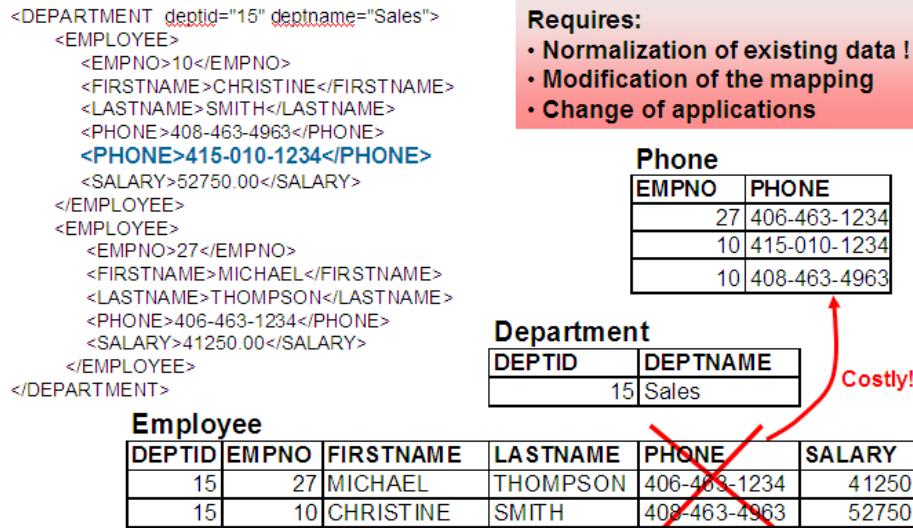


Рисунок 15.6. Повышенная гибкость данных при использовании XML

4. Можно повысить производительность XML-приложений. Проверки, проведенные с применением технологии pureXML, показали значительное повышение производительности XML-приложений. В Таблице 15.1 показаны результаты таких проверок для компаний, которая перешла на pureXML с более старых технологий. Во втором столбце содержатся результаты применения старого метода работы с XML через другую реляционную базу данных, а в третьем столбце приведены результаты использования DB2 с технологией pureXML.

| Задача | Другая реляционная БД | DB2 с pureXML |
|--|---------------------------|------------------------|
| Разработка бизнес-процессов для поиска и извлечения данных | CLOB: 8 ч Нарезка: 2 ч | 30 мин. |
| Относительное число строк кода для ввода/вывода | 100 | 35 (сокращение на 65%) |
| Добавление поля в схему | 1 неделя | 5 мин |
| Запросы | 24—36 ч | 20 с – 10 мин |

Таблица 15.1. Повышенная производительность с использованием технологии pureXML

15.3.2 Основы XPath

XPath — это язык, применяемый для создания запросов в XML-документах. В листинге 15.1 показан XML-документ, а на рис. 15.7 проиллюстрирован тот же документ, представленный в формате иерархии с синтаксическим анализом (также называемом форматом «узла» или «листа»). Мы воспользуемся форматом иерархии с синтаксическим анализом, чтобы объяснить принцип работы XPath.

```
<dept bldg="101">
  <employee id="901">
    <name>John Doe</name>
    <phone>408 555 1212</phone>
    <office>344</office>
  </employee>
  <employee id="902">
    <name>Peter Pan</name>
    <phone>408 555 9918</phone>
    <office>216</office>
  </employee>
</dept>
```

Листинг 15.1. XML-документ

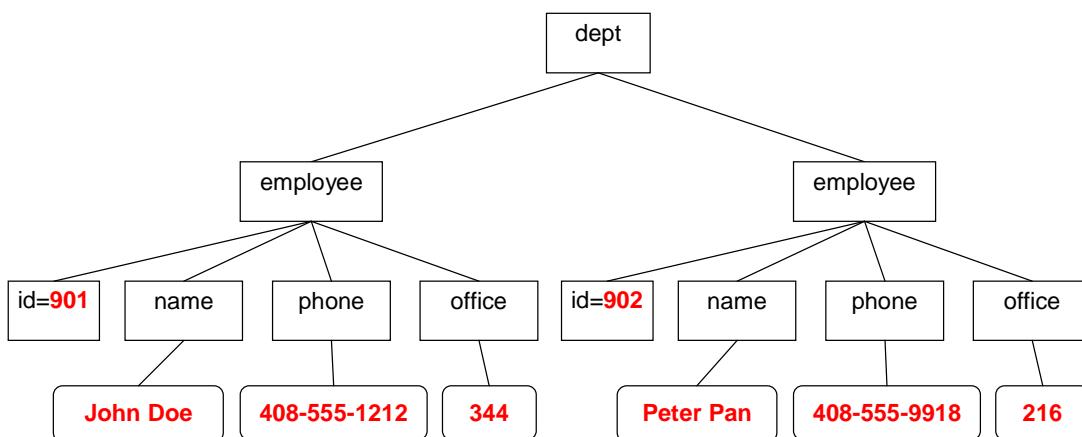


Рисунок 15.7. Представление XML-документа с листинга 15.1 в формате иерархии с синтаксическим анализом

XPath можно быстро изучить, сравнив этот язык с командой `change directory (cd)` в MS-DOS или Linux/UNIX. С помощью команды `cd` мы перемещаемся по дереву каталогов следующим образом:

```
cd /directory1/directory2/...
```

Аналогично, в XPath для перехода от одного элемента XML-документа к другому используется косая черта. К примеру, используя документ из листинга 15.1 в XPath можно получить имена всех сотрудников с помощью такого запроса:

```
/dept/employee/name
```

15.3.2.1 Выражения XPath

Выражения XPath используют полностью классифицированные пути для определения элементов и атрибутов. Для указания на атрибут используется знак «@». Чтобы извлечь только значение (текстовый узел) элемента, воспользуйтесь функцией `text()`. В табл. 15.2 показаны запросы XPath и соответствующие результаты для XML-документа из листинга 15.1.

| XPath | Результат |
|----------------------------|---|
| /dept/@bldg | 101 |
| /dept/employee/@id | 901 902 |
| /dept/employee/name | <name>Peter Pan</name> <name>John Doe</name> |
| /dept/employee/name/text() | Peter Pan John Doe |

Таблица 15.2. Примеры выражений XPath

15.3.2.2 Специальные символы XPath

В XPath используются два основных специальных символов:

- «*» соответствует любому имени тега
- «//» — это специальный символ «descendant-or-self» (указывающий на сам элемент и всех его потомков)

В табл. 15.3 представлено больше примеров для XML-документа из листинга 15.1.

| XPath | Результат |
|--------------------------|---|
| /dept/employee/* /text() | John Doe 408 555 1212 344 Peter Pan 408 555 9918 216 |
| /dept/* /@id | 901 902 |
| //name/text() | Peter Pan John Doe |
| /dept//phone | <phone>408 555 1212</phone> <phone>408 555 9918</phone> |

Таблица 15.3. Примеры специальных символов XPath

15.3.2.3 Предикаты XPath

Предикаты записываются в квадратных скобках []. Их можно рассматривать как аналог оператора WHERE в SQL. К примеру, `[@id="902"]` читается как: «WHERE (где) идентификатор атрибута (id) равен 902». Одно выражение XPath может содержать несколько предикатов. Чтобы задать позиционный предикат, используйте `[n]` — будет выбран $n^{\text{ый}}$ дочерний элемент. К примеру, `employee[2]` означает, что будет выбран второй сотрудник. В табл. 15.4 представлены дополнительные примеры.

| XPath | Результат |
|---|---|
| <code>/dept/employee[@id="902"]/name</code> | <code><name>Peter Pan</name></code> |
| <code>/dept[@bldg="101"]/employee[office >"300"]/name</code> | <code><name>John Doe</name></code> |
| <code>//employee[office="344" OR office="216"]/@id</code> | 901 902 |
| <code>/dept/employee[2]/@id</code> | 902 |

Таблица 15.4. Примеры предикатов XPath

15.3.2.4 Родительская ось

Как в MS-DOS или Linux/UNIX, можно использовать «.» (точку) для обозначения выражения, ссылающегося на текущий контекст, и «..» (двойную точку), чтобы сослаться на родительский контекст. В табл. 15.5 представлены дополнительные примеры.

| XPath | Результат |
|---|---|
| <code>/dept/employee/name[../../@id="902"]</code> | <code><name>Peter Pan</name></code> |
| <code>/dept/employee/office[.>"300"]</code> | <code><office>344</office></code> |
| <code>/dept/employee[office > "300"]/office</code> | <code><office>344</office></code> |
| <code>/dept/employee[name="John Doe"]/.../@bldg</code> | 101 |
| <code>/dept/employee/name[.= "John Doe"]/.../@bldg</code> | 101 |

Таблица 15.5. Родительская ось XPath

15.3.3 Основы XQuery

XQuery — это язык запросов, созданный для XML. XQuery поддерживает выражения маршрута для навигации в иерархической структуре XML. Фактически, XPath — сокращенная версия XQuery; соответственно, все сказанное выше об XPath также относится и к XQuery. XQuery поддерживает как типизированные, так и нетипизированные данные. В XQuery нет пустых значений, поскольку в XML-

документах недостающие или неизвестные данные пропускаются. Выражения XQuery и XPath различают регистр букв, и XQuery возвращает последовательности XML-данных.

XQuery поддерживает выражение FLWOR. Если сравнивать с SQL, это выражение эквивалентно выражению SELECT-FROM-WHERE. В следующем разделе выражение FLWOR рассмотрено более подробно.

15.3.3.1 XQuery: выражение FLWOR

FLWOR означает следующее:

- FOR: проходит по последовательности, привязывает переменную к элементу;
- LET: привязывает переменную к последовательности;
- WHERE: устраниет элементы итерации;
- ORDER: заново упорядочивает элементы итерации;
- RETURN: конструирует результаты запроса.

Это выражение позволяет манипулировать XML-документами, давая возможность получить другое выражение. К примеру, предположим, что есть таблица с таким определением:

```
CREATE TABLE dept(deptID CHAR(8),deptdoc XML);
```

XML-документ, показанный в листинге 15.2, вставляется в столбец *deptdoc*

```
<dept bldg="101">
  <employee id="901">
    <name>John Doe</name>
    <phone>408 555 1212</phone>
    <office>344</office>
  </employee>
  <employee id="902">
    <name>Peter Pan</name>
    <phone>408 555 9918</phone>
    <office>216</office>
  </employee>
</dept>
```

Листинг 15.2. Образец XML-документа

Затем можно выполнить показанный в листинге 15.3 оператор XQuery с выражением FLWOR:

```
xquery
for $d in db2-fn:xmlcolumn('dept.deptdoc')/dept
```

```

let $emp := $d//employee/name
where $d/@bldg > 95
order by $d/@bldg
return
<EmpList>
{$d/@bldg, $emp}
</EmpList>
```

Листинг 15.3. Образец оператора XQuery с выражением FLWOR

На выходе получим результат, показанный в листинге 15.4.

```

<EmpList bldg="101">
<name>
    John Doe
</name>
<name>
    Peter Pan
</name>
</EmpList>
```

Листинг 15.4. Результат выполнения оператора XQuery из листинга 15.3**15.3.4 Вставка XML-документов**

Документы XML можно вставить в базу данных DB2 с помощью оператора SQL INSERT или утилиты IMPORT. XQuery нельзя использовать с такой целью, поскольку в стандарте это еще не определено.

Рассмотрим показанный в листинге 15.5 ниже сценарий table_creation.txt, который можно запустить из командного окна DB2 или командного процессора Linux с помощью такого оператора:

```

db2 -tvf table_creation.txt

-- (1)
drop database mydb
;

-- (2)
create database mydb using codeset UTF-8 territory US
;

-- (3)
connect to mydb
;

-- (4)
create table items (
    id      int primary key not null,
```

```
brandname  varchar(30),
itemname   varchar(30),
sku        int,
srp        decimal(7,2),
comments   xml
);

-- (5)
create table clients(
id      int primary key not null,
name    varchar(50),
status  varchar(10),
contact  xml
);

-- (6)
insert into clients values (77, 'John Smith', 'Gold',
'<addr>111 Main St., Dallas, TX, 00112</addr>')
;

-- (7)
IMPORT FROM "D:\Raul\clients.del" of del xml from "D:\Raul" INSERT INTO
CLIENTS (ID, NAME, STATUS, CONTACT)
;

-- (8)
IMPORT FROM "D:\Raul\items.del" of del xml from "D:\Raul" INSERT INTO
ITEMS (ID, BRANDNAME, ITEMNAME, SKU, SRP, COMMENTS)
;
```

Листинг 15.5. Содержимое файла table_creation.txt

Обратите внимание, что файл сценария и связанные файлы содержатся в скжатом файле **Expressc_book_exercises_9.7.zip**, прилагаемом к этой книге. Рассмотрим по отдельности каждую строку сценария, представленного в листинге 15.5.

1. Удалить базу данных *mydb*. Обычно это действие выполняется в файлах сценариев для очистки оперативной памяти. Если *mydb* не существует, появится сообщение об ошибке, но это нормально.
2. Создать базу данных *mydb*, используя кодовый набор (codeset) UTF-8. В результате будет создана база данных Unicode. Технология pureXML поддерживается как в базах данных Unicode, так и в других базах данных.
3. Подключиться к новосозданной базе данных *mydb*. Это необходимо для создания объектов в базе данных.

4. Создать таблицу *items*. Обратите внимание, что последний столбец таблицы (столбец *comments*) определяется как столбец XML, использующий новый тип данных XML.
5. Мы создаем таблицу *clients*. Обратите внимание, что последний столбец таблицы (столбец *contact*) также определяется с новым типом данных XML.
6. Используя этот SQL-оператор INSERT, можно вставить XML-документ в столбец XML. В оператор INSERT XML-документ вводится как строка в одинарных кавычках.
7. С помощью команды IMPORT можно вставить или импортировать в базу данных несколько XML-документов наравне с реляционными данными. В пункте (7) мы импортируем данные из файла *clients.del* (файл ASCII с разделителями), а также указываем, где расположены XML-данные, на которые ссылается этот файл *clients.del* (например, в каталоге D:\Raul).

Мы рассмотрим файл *clients.del* более тщательно, но для начала ознакомимся с содержимым каталога D:\Raul. Эта информация показана на рис. 15.8.

| Name | Size | Type |
|-----------------|------|--------------|
| Client3227.xml | 1 KB | XML Document |
| Client4309.xml | 1 KB | XML Document |
| Client5681.xml | 1 KB | XML Document |
| Client8877.xml | 1 KB | XML Document |
| Client9077.xml | 1 KB | XML Document |
| Client9177.xml | 1 KB | XML Document |
| ClientInfo.xsd | 2 KB | XML Schema |
| clients.del | 1 KB | DEL File |
| Comment3926.xml | 1 KB | XML Document |
| Comment4023.xml | 1 KB | XML Document |
| Comment4272.xml | 1 KB | XML Document |
| items.del | 1 KB | DEL File |

Рисунок 15.8. Содержимое каталога D:\Raul с XML-документами

В листинге 15.6 показано содержимое текстового файла *clients.del*.

```
3227,Ella Kimpton,Gold,<XDS FIL='Client3227.xml' />,
8877,Chris Bontempo,Gold,<XDS FIL='Client8877.xml' />,
9077,Lisa Hansen,Silver,<XDS FIL='Client9077.xml' />,
9177,Rita Gomez,Standard,<XDS FIL='Client9177.xml' />,
5681,Paula Lipenski,Standard,<XDS FIL='Client5681.xml' />,
4309,Tina Wang,Standard,<XDS FIL='Client4309.xml' />
```

Листинг 15.6. Содержимое файла *clients.del*

В файле clients.del "XDS FIL=" используется для указания на определенный файл документа XML.

На рис. 15.9 показан Центр управления после выполнения описанного выше сценария.

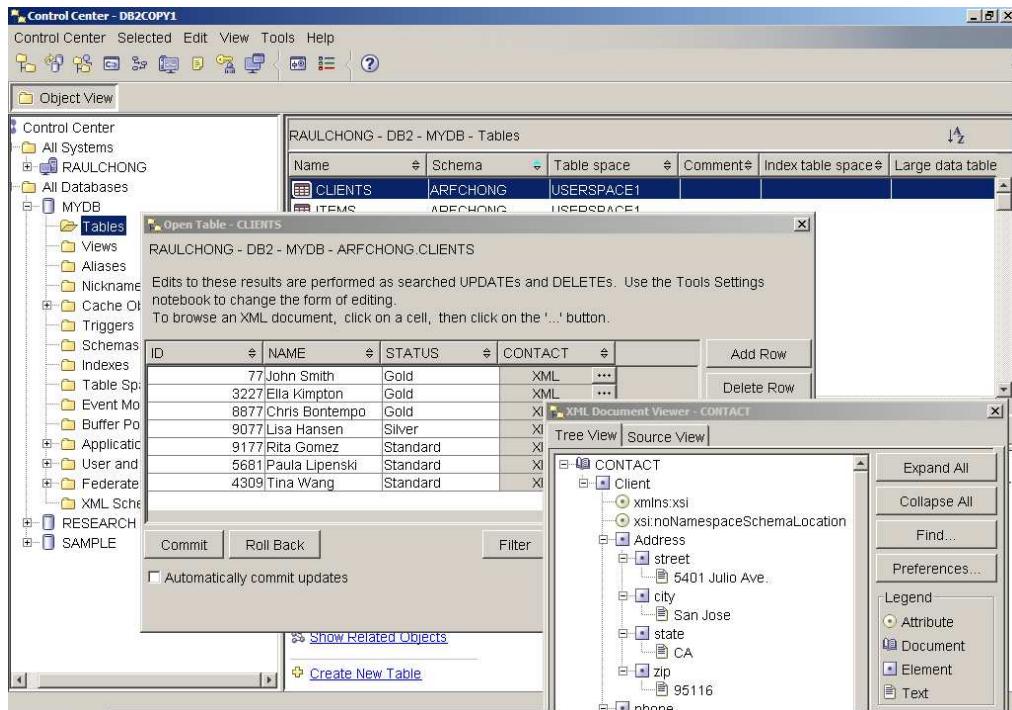


Рисунок 15.9. Центр управления после выполнения сценария table_creation.txt

Обратите внимание, что на рисунке показано содержимое таблицы CLIENTS. Последний столбец contact является столбцом XML. Если нажать кнопку с троеточием, откроется окно с содержимым XML-документа. Это окно показано в нижнем правом углу рис. 15.9.

15.3.5 Запрос XML-данных

В DB2 есть два способа запросить XML-данные:

- с помощью SQL с расширениями XML (SQL/XML);
- с помощью Xquery.

В обоих случаях DB2 придерживается международных стандартов XML.

15.3.5.1 Запрос XML-данных с помощью SQL/XML

Применение стандартных операторов SQL дает возможность работать со строками и столбцами. Оператор SQL можно использовать для работы с полными XML-документами; однако это не поможет при попытке извлечь только часть такого документа. В таких случаях следует использовать SQL с расширениями XML (SQL/XML).

В табл. 15.6 описаны некоторые функции SQL/XML, доступные по стандарту SQL 2006.

| Имя функции | Описание |
|--------------|--|
| XMLPARSE | Обрабатывает двоичные данные символьных или больших объектов, создает значение XML |
| XMLSERIALIZE | Конвертирует значение XML в двоичные данные символьного или большого объекта |
| XMLVALIDATE | Проверяет значение XML, сравнивая его со схемой XML, и отмечает тип значения XML |
| XMLEXISTS | Определяет, возвращает ли XQuery результаты (т. е. последовательность одного или нескольких элементов) |
| XMLQUERY | Выполняет XQuery и возвращает последовательность результатов |
| XMLTABLE | Выполняет XQuery и возвращает последовательность результатов в виде реляционной таблицы (по возможности) |
| XMLCAST | Преобразование в тип XML или из типа XML |

Таблица 15.6. Функции SQL/XML

Представленные ниже примеры можно протестировать на созданной ранее базе данных *mydb*.

Пример 1

Предположим, что необходимо найти имена всех клиентов, проживающих в зоне действия определенного почтового кода. Адреса клиентов, в том числе почтовые коды, хранятся в столбце XML таблицы *clients* (клиенты). С помощью XMLEXISTS можно выполнить в столбце XML поиск целевого почтового индекса, а затем установить соответствующие ограничения на возвращаемые результаты. В листинге 15.7 ниже показан необходимый запрос.

```
SELECT name FROM clients
WHERE xmlexists(
    '$c/Client/Address[zip="95116"]'
    passing clients.contact as "c"
)
```

Листинг 15.7. Пример использования XMLEXISTS

Первая строка в листинге 15.7 — это выражение SQL, указывающее, что вы хотите извлечь информацию из столбца *name* таблицы *clients*.

Выражение WHERE вызывает функцию XMLEXISTS, задавая выражение XPath, которое указывает DB2 перейти к элементу `zip` и проверить наличие значения 95116.

Выражение `$c/Client/Address` обозначает путь в иерархии XML-документа, по которому DB2 может найти элемент `zip`. Знак доллара (\$) используется для обозначения переменной; соответственно, «`c`» — это переменная. Затем эта переменная определяется такой строкой: `passing clients.contact as "c"`. Тут `clients` — это имя таблицы, а `contact` — имя столбца с типом данных XML. Иными словами, мы передаем XML-документ в переменную «`c`».

DB2 проверяет XML-данные, содержащиеся в столбце `contact`, переходит от корневого узла `client` вниз к узлу `Address`, затем к узлу `zip`, и наконец определяет, проживает ли клиент (`customer`) в зоне действия целевого почтового кода (`zip`). Функция XMLEXISTS фиксирует значение «`true`» (истина), и DB2 возвращает имя клиента, ассоциируемое с этой строкой.

Начиная с DB2 9.5, вышепредставленный запрос можно упростить, как показано в листинге 15.8 ниже.

```
SELECT name FROM clients
WHERE xmlexists(
  '$CONTACT/Client/Address[zip="95116"]'
)
```

Листинг 15.8. Упрощенная версия запроса, показанного в листинге 15.7

DB2 автоматически создает переменную с таким же именем, как у столбца XML. В примере выше DB2 автоматически создает переменную `CONTACT`. Её имя совпадает с именем XML-столбца `CONTACT`.

Пример 2

Попробуем создать отчет со списком адресов электронной почты клиентов со статусом «Gold». Для этого можно выполнить запрос, показанный в листинге 15.9 ниже.

```
SELECT xmlquery('$c/Client/email' passing contact as "c")
  FROM clients
 WHERE status = 'Gold'
```

Листинг 15.9. Пример использования XMLQUERY

Первая строка обозначает, что мы хотим получить адрес электронной почты, являющийся элементом XML-документа (а не реляционного столбца). Как и в предыдущем примере, «`$c`» является переменной, содержащей XML-документ. В этом примере мы используем только функцию XMLQUERY, которую можно применить после SELECT; с другой стороны, функция XMLEXISTS применяется после выражения WHERE.

Пример 3

Бывают случаи, когда данные XML необходимо представить в виде таблиц. Это можно сделать с помощью функции XMLTABLE, как показано в листинге 15.10 ниже.

```
SELECT t.comment#, i.itemname, t.customerID, Message
  FROM items i,
       xmltable('$c/Comments/Comment' passing i.comments as "c"
                columns Comment# integer path 'CommentID',
                        CustomerID integer path 'CustomerID',
                        Message varchar(100) path 'Message') AS t
```

Листинг 15.10. Пример использования XMLTABLE

В первой строке указываются столбцы, которые будут включены в набор результатов. Столбцы с переменной «t» в качестве префикса основаны на значениях элемента XML.

Третья строка вызывает функцию XMLTABLE, чтобы указать на столбец XML DB2 с целевыми данными (*i.comments*) и путь в XML-документах этой строки, по которому расположены интересующие нас элементы.

Выражение *columns*, записанное в строках 4—6, обозначает определенные XML-элементы, которые будут преобразованы в выходные столбцы в наборе результатов SQL, заданном в строке 1. Такое преобразование частично включает обозначение типов данных, в которые будут конвертироваться значения элемента XML. В этом примере все XML-данные конвертируются в стандартные типы данных SQL.

Пример 4

Теперь рассмотрим простой пример, в котором выражение XQuery FLWOR будет включено в функцию XMLQUERY SQL/XML. Это показано в Листинге 15.11.

```
SELECT name, xmlquery(
  'for $e in $c/Client/email[1] return $e'
  passing contact as "c"
)
  FROM clients
 WHERE status = 'Gold'
```

Листинг 15.11. Пример использования XMLQUERY и FLWOR

Первая строка обозначает, что имена клиентов и выход функции XMLQUERY будут включены в набор результатов. Вторая строка указывает на то, что на выходе должен быть первый подэлемент *email* элемента *Client*. Третья строка определяет источник XML-данных (столбец *contact*). Четвертая строка показывает, что этот столбец расположен в таблице *clients*, а пятая указывает, что нас интересуют только клиенты *Gold*.

Пример 5

Показанный в листинге 15.12 пример снова демонстрирует функцию XMLQUERY, в которой содержится выражение XQuery FLWOR; однако обратите внимание, что на этот раз на выходе получаем не только XML, но и HTML.

```
SELECT xmlquery('for $e in $c/Client/email[1]/text()
    return <p>{$e}</p>'
    passing contact as "c")
FROM clients
WHERE status = 'Gold'
```

Листинг 15.12. Пример возврата XML и HTML

Выражение возврата XQuery позволяет трансформировать результаты XML по мере необходимости. Использование функции `text()` в первой строке означает, что нас интересует только текстовое представление первого адреса электронной почты подходящих клиентов. Вторая строка указывает, что данная информация должна быть должна с двух сторон выделяться HTML-тегами абзаца.

Пример 6

В представленном ниже примере функция XMLEMENT используется для создания серии элементов, каждый из которых содержит подэлементы для значений идентификатора, товарной марки и единицы учета запасов (SKU), получаемых из соответствующих столбцов таблицы `items`. По сути, функцию XMLEMENT можно использовать, если нужно преобразовать реляционные данные в данные XML. Это показано в листинге 15.13.

```
SELECT
    xmlelement (name "item", itemname,
    xmlelement (name "id", id),
    xmlelement (name "brand", brandname),
    xmlelement (name "sku", sku)
FROM items
WHERE srp < 100
```

Листинг 15.13. Пример использования XMLEMENT

Результаты запроса из листинга 15.13 показаны в листинге 15.14.

```
<item>
    <id>4272</id>
    <brand>Classy</brand>
    <sku>981140</sku>
</item>
...
<item>
    <id>1193</id>
    <brand>Natural</brand>
    <sku>557813</sku>
</item>
```

Листинг 15.14. Результат запроса из листинга 15.13

15.3.5.2 Запрос XML-данных с помощью XQuery

В предыдущем разделе мы рассматривали запросы XML-данных при помощи SQL с расширениями XML. SQL всегда был основным методом создания запросов, а XPath или XQuery встраивались в SQL. В этом разделе мы рассмотрим запросы XML-данных с помощью XQuery. На этот раз основным методом запроса будет XQuery, а в некоторых случаях будем использовать SQL, встроенный в XQuery (с помощью функции `db2-fn:sqlquery`). Используя XQuery, мы запустим несколько функций и воспользуемся выражением FLWOR.

Пример 1

Это простой пример XQuery для вывода контактных данных клиентов. В этом примере `CONTACT` — это имя столбца XML, а `CLIENTS` — имя таблицы.

```
xquery db2-fn:xmlcolumn('CLIENTS.CONTACT')
```

Всегда выделяйте все выражения XQuery командой `xquery` в качестве префикса, чтобы сообщить DB2 о необходимости использовать анализатор XQuery; в противном случае DB2 предполагает, что используется выражение SQL. Функция `db2-fn:xmlcolumn` — это функция, извлекающая XML-документы из столбца, указанного в качестве параметра. Для следующего оператора SQL эта функция является обязательной, поскольку выполняется извлечение содержимого всего столбца:

```
SELECT contact FROM clients
```

Пример 2

В примере, показанном в листинге 15.15, мы используем выражение FLWOR для вывода данных о номерах факса клиентов.

```
xquery
for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/fax
return $y
```

Листинг 15.15. XQuery и выражение FLWOR

В первой строке инициируется анализатор XQuery. Вторая строка дает DB2 указание выполнить итерацию для всех подэлементов «fax», содержащихся в столбце `CLIENTS.CONTACT`. Каждый элемент «fax» привязан к переменной `$y`. В третьей строке указано, что для каждой итерации возвращается значение `«$y»`.

Результаты этого запроса показаны в листинге 15.16 (мы пропустили в выводе область имен, поскольку она может занимать несколько строк, и листинг было бы неудобно читать):

```
<fax>4081112222</fax>
<fax>5559998888</fax>
```

Листинг 15.16. Результат запроса из листинга 15.15

Пример 3

В примере из листинга 15.17 запрашивает XML-данные и возвращает результаты в формате HTML.

```
xquery
<ul>
  for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/Address
  order by $y/zip
  return <li>{$y}</li>
}
</ul>
```

Листинг 15.17. Оператор XQuery с выражением FLWOR, HTML на выходе

Пример HTML-результатов показан в листинге 15.18.

```
<ul>
<li>
<address>
  <street>9407 Los Gatos Blvd.</street>
  <city>Los Gatos</city>
  <state>ca</state>
  <zip>95302</zip>
</address>
</li>
<address>
  <street>4209 El Camino Real</street>
  <city>Mountain View</city>
  <state>CA</state>
  <zip>95302</zip>
</address>
</li>
...
</ul>
```

Листинг 15.18. Результат запроса, выполненного в листинге 15.17

Пример 4

В этом примере показано, как встроить SQL в XQuery с помощью функции `db2-fn:sqlquery`. Функция `db2-fn:sqlquery` выполняет SQL-запрос и возвращает только выбранные XML-данные. Передаваемый на `db2-fn:sqlquery` SQL-запрос должен возвращать только XML-данные. Дальнейшая обработка этих XML-данных выполняется в XQuery. Это показано в листинге 15.19.

```
xquery
for $y in
```

```
db2-fn:sqlquery(
  'select comments from items where srp > 100'
) /Comments/Comment
where $y/ResponseRequested='Yes'
return (
  <action>
  {$y/ProductID
   $y/CustomerID
   $y/Message}
  </action>
)
)
```

Листинг 15.19. Пример функции db2-fn:sqlquery, которая встраивает SQL в XQuery

В нашем примере SQL-запрос фильтрует строки на основании условия, что столбец `srp` имеет значение больше 100. Из отфильтрованных строк формируется столбец `comments`, являющийся столбцом XML. Затем применяется XQuery (или XPath) для перехода к подэлементам.

Примечание.

SQL нечувствителен к регистру, и по умолчанию DB2 сохраняет все имена таблиц и столбцов в верхнем регистре. С другой стороны, XQuery чувствителен к регистру. Показанные выше функции — это функции интерфейса XQuery, поэтому все имена таблиц и столбцов для этих функций должны указываться в верхнем регистре. Указание имен объектов в нижнем регистре может привести к ошибке неопределенного имени объекта.

15.3.6 Операции соединения с SQL/XML

В этом разделе описано, как выполнять операции JOIN между двумя столбцами XML разных таблиц, а также между одним столбцом XML и одним столбцом реляционных данных. Предположим, вы создали две таблицы, воспользовавшись операторами, показанными в листинге 15.20.

```
CREATE TABLE dept (unitID CHAR(8), deptdoc XML)
CREATE TABLE unit (unitID CHAR(8) primary key not null,
  name CHAR(20),
  manager VARCHAR(20),
  ...
)
```

Листинг 15.20. DDL таблиц для примеров операций JOIN

Операцию JOIN можно выполнить одним из двух способом. Первый способ показан в листинге 15.21.

```
SELECT u.unitID
  FROM dept d, unit u
 WHERE XMLEXISTS (
```

```
'$e//employee[name = $m]'  
passing d.deptdoc as "e", u.manager as "m")
```

Листинг 15.21. Первый способ выполнения операции JOIN с SQL/XML

Четвертая строка оператора в листинге выше показывает, что операция JOIN выполняется между элементом `name`, который является подэлементом XML-столбца `deptdoc` таблицы `dept`, и реляционным столбцом `manager` таблицы `unit`.

В листинге 15.22 показан второй способ выполнения операции JOIN.

```
SELECT u.unitID  
FROM dept d, unit u  
WHERE u.manager = XMLCAST(  
    XMLQUERY(''$e//employee/name '  
        passing d.deptdoc as "e"  
    AS char(20))
```

Листинг 15.22. Второй способ выполнения операции JOIN с SQL/XML

При использовании второго способа реляционный столбец расположен по левую сторону JOIN. Если по левую сторону уравнения расположен реляционный столбец, реляционный индекс может использоваться вместо XML-индекса.

15.3.7 Операции соединения с XQuery

Предположим, были созданы следующие таблицы:

```
CREATE TABLE dept (unitID CHAR(8), deptdoc XML)  
CREATE TABLE project(projectDoc XML)
```

Если использовать SQL/XML, операция JOIN будет выглядеть, как показано в листинге 15.23.

```
SELECT XMLQUERY (  
    '$d/dept/employee' passing d.deptdoc as "d")  
FROM dept d, project p  
WHERE XML EXISTS (  
    '$e/dept[@deptID=$p/project/deptID]'  
        passing d.deptdoc as "e", p.projectDoc as "p")
```

Листинг 15.23. Операция JOIN с SQL/XML

Эквивалентная операция JOIN с использованием XQuery показана в листинге 15.24.

```
xquery  
for $dept in db2-fn:xmlcolumn("DEPT.DEPTDOC")/dept  
for $proj in db2-fn:xmlcolumn("PROJECT.PROJECTDOC")/project  
where $dept/@deptID = $proj/deptID  
return $dept/employee
```

Листинг 15.24. Операция JOIN с XQuery

Второй способ более наглядный — переменная `$dept` содержит XML-документ столбца XML `deptdoc` таблицы `dept`. Переменная `$proj` содержит XML-документ

столбца XML `projectdoc` таблицы `project`. Затем четвертая строка выполняет операцию JOIN между атрибутом первого XML-документа и элементом второго XML-документа.

15.3.8 Операции обновления и удаления

Есть два способа выполнения операций обновления и удаления XML-данных:

- с помощью SQL-операторов UPDATE и DELETE;
- с помощью выражения TRANSFORM.

При использовании SQL-операторов UPDATE и DELETE обновление или удаление происходит на уровне документа; иными словами, целый XML-документ заменяется обновленным. К примеру, при использовании показанного в [листинге 15.25](#) ниже оператора UPDATE, даже если необходимо изменить элемент `<state>`, будет заменен весь XML-документ.

```
UPDATE clients SET contact=(
  xmlparse(document
    '<Client>
      <address>
        <street>5401 Julio ave.</street>
        <city>San Jose</city>
        <state>CA</state>
        <zip>95116</zip>
      </address>
      <phone>
        <work>4084633000</work>
        <home>4081111111</home>
        <cell>4082222222</cell>
      </phone>
      <fax>4087776666</fax>
      <email>newemail@someplace.com</email>
    </Client>' )
)
WHERE id = 3227
```

Листинг 15.25. Пример SQL-оператора UPDATE

Используя второй способ, можно выполнять обновление поддокументов с помощью выражения TRANSFORM, что значительно эффективнее. Этот способ позволяет заменять, вставлять, удалять или переименовывать узлы XML-документа. Также можно изменить значение узла, не заменяя сам узел. Обычно это служит для изменения значения элемента или атрибута, что является очень распространенным типом обновления. Поддержка этого способа была представлена в DB2 9.5.

Выражение TRANSFORM является частью языка XQuery. Его можно использовать во всех случаях, в которых обычно используется XQuery, к примеру в выражении FLWOR или функции XMLQUERY в операторе SQL/XML. Это выражение чаще всего применяется в SQL-операторе UPDATE для модификации XML-документа в столбце XML.

В листинге 15.26 показана синтаксическая структура выражения TRANSFORM.

```
>>-transform--| copy clause |--| modify clause |--| return clause |-><

Раздел copy (копирование)

.---,-----.
  V      |
| --copy---$VariableName---CopySourceExpression-----|  
  
Раздел modify (модификация)

| --modify--ModifyExpression-----|  
  
Раздел return (возврат)

| --return--ReturnExpression-----|
```

Листинг 15.26. Синтаксическая структура выражения TRANSFORM

Раздел copy используется, чтобы назначить для переменной XML-документы, которые необходимо обработать.

В разделе modify можно инициировать выражение insert, delete, rename или replace. Эти выражения дают возможность обновить XML-документ.

Например:

- Если необходимо добавить в документ новые узлы, воспользуйтесь выражением insert.
- Чтобы удалить узлы из XML-документа, воспользуйтесь выражением delete.
- Чтобы переименовать элемент или атрибут XML-документа, воспользуйтесь выражением rename.
- Чтобы заменить существующий узел новым узлом или последовательностью узлов, воспользуйтесь выражением replace. Значение замены в выражении может использоваться только для изменения значения элемента или атрибута.

Раздел «return» возвращает результат выражения TRANSFORM.

В листинге 15.27 показан пример оператора UPDATE с выражением TRANSFORM.

```
(1)-- UPDATE customers
(2)-- SET contactinfo = xmlquery( 'declare default element namespace
(3)--           "http://posample.org";
(4)-- transform
(5)--   copy $newinfo := $c
(6)--     modify do insert <email2>my2email.gm.com</email2>
(7)--       as last into $newinfo/customerinfo
(8)--   return $newinfo' passing contactinfo as "c")
(9)-- WHERE id = 100
```

Листинг 15.27. Оператор UPDATE с выражением TRANSFORM

В приведенном выше примере строки (1), (2) и (9) являются частью синтаксической структуры SQL UPDATE. В строке (2) инициируется функция XMLQUERY, вызывающая выражение TRANSFORM в строке (4). Блок выражения TRANSFORM переходит из строки (4) в строку (8) и используется для вставки нового узла в XML-документ, содержащий элемент `email2`. Обратите внимание, что обновление элементов XML-документа через представление не поддерживается.

Удаление целых XML-документов из таблиц выполняется прямолинейно, как при использовании оператора SELECT в SQL/XML. Воспользуйтесь SQL-оператором DELETE и укажите все необходимые предикаты WHERE.

15.3.9 Индексирование XML

В XML-документе можно создать индексы для элементов, атрибутов или значений (тековых узлов). Ниже представлено несколько примеров. Предположим, что была создана таблица ниже:

```
CREATE TABLE customer(info XML)
```

Также предположим, что представленный в листинге 15.28 XML-документ — один из документов, хранящихся в таблице.

```
<customerinfo Cid="1004">
  <name>Matt Foreman</name>
  <addr country="Canada">
    <street>1596 Baseline</street>
    <city>Toronto</city>
    <state>Ontario</state>
    <pcode>M3Z-5H9</pcode>
  </addr>
  <phone type="work">905-555-4789</phone>
  <phone type="home">416-555-3376</phone>
  <assistant>
    <name>Peter Smith</name>
    <phone type="home">416-555-3426</phone>
  </assistant>
</customerinfo>
```

Листинг 15.28. XML-документ для примеров об индексах XML

Показанный в листинге 15.29 оператор создает индекс в атрибуте `Cid`

```
CREATE UNIQUE INDEX idx1 ON customer(info)
  GENERATE KEY USING
  xmlpattern '/customerinfo/@Cid'
  AS sql DOUBLE
```

Листинг 15.29 — Индекс в атрибуте Cid

Показанный в листинге 15.30 оператор создает индекс в элементе `name`

```
CREATE INDEX idx2 ON customer(info)
  GENERATE KEY USING
  xmlpattern '/customerinfo/name'
  AS sql VARCHAR(40)
```

Листинг 15.30. Индекс в элементе name

Показанный в листинге 15.31 оператор создает индекс во всех элементах `name`

```
CREATE INDEX idx3 ON customer(info)
  GENERATE KEY USING
    xmlpattern '//name'
    AS sql VARCHAR(40);
```

Листинг 15.31. Индекс во всех элементах name

Показанный в Листинге 15.32 оператор создает индекс во всех текстовых узлах (всех значениях). Не рекомендуется использовать этот вариант, поскольку поддержание индекса для операций обновления, удаления и вставки будет слишком ресурсоемким, а сам индекс будет слишком большим.

```
CREATE INDEX idx4 ON customer(info)
  GENERATE KEY USING
    xmlpattern '//text()'
    AS sql VARCHAR(40);
```

Листинг 15.32. Индекс во всех текстовых узлах (не рекомендуется)

15.4 Работа со схемами XML

DB2 дает возможность вставить правильно построенный XML-документ в базу данных. Если документ построен неправильно, во время вставки появится сообщение об ошибке. С другой стороны, в DB2 не обязательно проверять XML-документ. Чтобы проверить XML-документ, можно воспользоваться одним из нескольких описанных в этом разделе способов.

15.4.1 Регистрация XML-схем

Схемы XML хранятся в базах данных DB2 в так называемом репозитории схем XML. Для добавления схем XML в репозиторий используется команда REGISTER XMLSCHEMA.

К примеру, предположим, что в файле `order.xml` хранится XML-документ (см. рис. 15.10).

```
<?xml version="1.0" encoding="UTF-8"?>
<po:PurchaseOrder xmlns:po="http://www.test.com/po">
    <Header>
        <Id>1</Id>
        <date>2004-01-29</date>
        <description>purchase order</description>
        <value>20</value>
        <status>shipped</status>
    </Header>
    <Items>
        <Item>
            <ItemDescription color="red" weight="5">
                <Name>Widget C</Name>
                <SKU>1</SKU>
                <Price>30</Price>
                <Comment>no comment</Comment>
            </ItemDescription>
            <NumberOrdered>1</NumberOrdered>
        </Item>
    </Items>
    <Customer type="regualar">
        <Name>Manoj K Sardana</Name>
        <Address>ring road, bangalore</Address>
        <Phone>918051055109</Phone>
        <email>msardana@in.ibm.com</email>
    </Customer>
</po:PurchaseOrder>
```

Рисунок 15.10. Файл order.xml, содержащий XML-документ

Теперь предположим, что в файле order.xsd хранится документ схемы XML (см. Рисунок 15.11).

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.test.com/po"
  xmlns:po="http://www.test.com/po"
  xmlns:head="http://www.test.com/header"
  xmlns:prod = "http://www.test.com/product"
  xmlns:cust = "http://www.test.com/customer">

  <xsd:import namespace="http://www.test.com/product" schemaLocation="product.xsd" />
  <xsd:import namespace="http://www.test.com/customer" schemaLocation="customer.xsd" />
  <xsd:import namespace="http://www.test.com/header" schemaLocation="header.xsd" />
  <xsd:complexType name="itemType">
    <xsd:sequence>
      <xsd:element name="Item" minOccurs="1" maxOccurs="unbounded" >
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="ItemDescription" type="prod:prodType" />
            <xsd:element name="NumberOrdered" type="xsd:integer" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="potype">
    <xsd:sequence>
      <xsd:element name="Header" type="head:headerType" />
      <xsd:element name="Items" type="po:itemType" />
      <xsd:element name="Customer" type="cust:customerType" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="PurchaseOrder" type="po:potype" />
</xsd:schema>

```

Рисунок 15.11. Файл order.xsd, содержащий XML-схему

В этом документе схемы XML овалом выделено следующее:

- <xsd:schema>: указывает, что это документ схемы XML
- <xsd:import ...>: мы импортируем другие файлы xsd (кроме схем XML), которые станут частью общей схемы XML
- minOccurs="1": пример «правила» схемы XML, в котором для элемента **Item** мы утверждаем появление хотя бы один раз, или, иными словами, что должен быть хотя бы один элемент **Item**

Затем для регистрации схемы XML в базе данных можно воспользоваться таким сценарием, как показано ниже в листинге 15.33. Этот сценарий включает комментарии, а потому не требует дополнительных объяснений.

```

-- ПОДКЛЮЧЕНИЕ К БАЗЕ ДАННЫХ
CONNECT TO SAMPLE;

-- РЕГИСТРАЦИЯ ГЛАВНОЙ СХЕМЫ XML
REGISTER XMLSCHEMA http://www.test.com/order FROM D:\example3\order.xsd AS
order;

```

```
-- ДОБАВЛЕНИЕ ДОКУМЕНТА СХЕМЫ XML К ГЛАВНОЙ СХЕМЕ
ADD XMLSCHEMA DOCUMENT TO order ADD http://www.test.com/header FROM
D:\example3\header.xsd;

-- ДОБАВЛЕНИЕ ДОКУМЕНТА СХЕМЫ XML К ГЛАВНОЙ СХЕМЕ
ADD XMLSCHEMA DOCUMENT TO order ADD http://www.test.com/product FROM
D:\example3\product.xsd;

-- ДОБАВЛЕНИЕ ДОКУМЕНТА СХЕМЫ XML К ГЛАВНОЙ СХЕМЕ
ADD XMLSCHEMA DOCUMENT TO order ADD http://www.test.com/customer FROM
D:\example3\customer.xsd;
```

```
-- ЗАВЕРШЕНИЕ РЕГИСТРАЦИИ СХЕМЫ
COMPLETE XMLSCHEMA order;
```

Листинг 15.33. Образец сценария, показывающий шаги регистрации схемы XML

Чтобы просмотреть эту информацию позже, можно выбрать (SELECT) информацию из таблиц каталога, как показано в листинге 15.34 ниже.

```
SELECT CAST(OBJECTSCHEMA AS VARCHAR(15)), CAST(OBJECTNAME AS VARCHAR(15))
FROM syscat.xsrobjects
WHERE OBJECTNAME='ORDER';
```

Листинг 15.34. Извлечение информации о схеме XML из таблиц каталога DB2

15.4.2 Подтверждение схемы XML

После регистрации схем XML в DB2, можно подтвердить XML-документы двумя способами:

- воспользуйтесь функцией XMLVALIDATE во время операции INSERT;
- воспользуйтесь триггером BEFORE.

На рис. 15.12 показан пример, в котором XML-документ с рис. 15.10 подтверждается в соответствии со схемой XML с рис. 15.11.

```

DROP TABLE t1;
CREATE TABLE t1 (po xml);

INSERT INTO t1 VALUES(xmlvalidate(xmlparse(document('<?xml version="1.0" encoding="UTF-8"?>
<po:PurchaseOrder xmlns:po="http://www.test.com/po">
<Header>
<Id>1</Id>
<date>2004-01-29</date>
<description>purchase order</description>
<value>20</value>
<status>shipped</status>
</Header>
<Items>
<Item>
<ItemDescription color="red" weight="5">
<Name>Widget C</Name>
<SKU>1</SKU>
<Price>30</Price>
<Comment>no comment</Comment>
</ItemDescription>
<NumberOrdered>1</NumberOrdered>
</Item>
</Items>
<Customer type="regular">
<Name>Manoj K Sardana</Name>
<Address>ring road, bangalore</Address>
<Phone>918051055109</Phone>
<email>msardana@in.ibm.com</email>
</Customer>
</po:PurchaseOrder>')) ACCORDING TO XMLSCHEMA ID order));

```

Рисунок 15.12. Подтверждение схемы XML с помощью XMLVALIDATE

Чтобы узнать, был ли подтвержден XML-документ, можно воспользоваться предикатом «IS VALIDATED» в ограничении CHECK (CHECK constraint).

XML-документы в столбце можно подтвердить с помощью различных схем XML. Это важно для простой миграции с XML-схемы версии 1 до версии 2. В том же столбце XML также можно найти XML-документы, для которых подтверждение не выполняется. Это полезно, если документы получены из надежных и ненадежных источников, и подтверждение схемы требуется только для последних.

15.4.3 Прочая поддержка XML

Небольшие XML-документы теперь могут быть в основной таблице. Это означает, что данные XML хранятся там же, где и реляционные данные, и могут воспользоваться такими же механизмами сжатия, как обычные реляционные данные. Более крупные XML-документы хранятся в отдельном внутреннем объекте, который также можно сжать.

DB2 также поддерживает эволюцию схем XML. Это означает, что при изменении схемы XML её можно легко обновить с помощью команды UPDATE XMLSCHEMA. Если выполнялись радикальные изменения схемы XML, могут появиться сообщения об ошибках.

В DB2 также поддерживается декомпозиция или «нарезка» XML. Это «старый» метод хранения XML в базах данных, который используется для хранения XML другими поставщиками. DB2 всё ещё поддерживает этот метод, если необходимо им воспользоваться; однако мы рекомендуем pureXML. DB2 также поддерживает XML Extender, использующий старый метод сохранения XML, однако это расширение больше не будет улучшаться.

В DB2 9.7 все преимущества pureXML расширены до разделов базы данных, обычно используемых в хранилищах данных. DPF входит в состав редакции DB2 Enterprise.

**new in
V9.7**

15.6 Краткий обзор

В этой главе мы познакомились с XML и технологией pureXML. Использование XML-документов стремительно развивается благодаря инструментам и методикам Web 2.0, а также SOA. Сохраняя XML-документы в базе данных DB2, можно воспользоваться преимуществами безопасности, производительности и гибкости кодирования с помощью pureXML. pureXML — это технология, позволяющая сохранять XML-документы в формате иерархии с синтаксическим анализом, в виде дерева. Сохранение происходит во время вставки в базу данных. Выполняя запросы, не нужно анализировать XML-документ, чтобы построить дерево до обработки. Дерево XML-документа уже построено и сохранено в базе данных. Кроме того, технология pureXML использует ядро XML в исходном формате, которое понимает XQuery; соответственно нет необходимости преобразовывать XQuery в SQL, а именно это происходит в других СУРБД.

В этой главе мы также рассмотрели операции вставки, удаления, обновления и создания запроса XML-документов с помощью SQL/XML и XQuery. Мы также обсудили индексы XML, схемы XML и другие функции, такие как сжатие и эволюция схем XML.

15.7 Упражнения

В этой главе мы изучили несколько примеров синтаксиса SQL/XML и XQuery, а также ознакомились с Редактором команд DB2 и IBM Data Studio. В этом упражнении мы проверим ваши знания об SQL/XML и XQuery, упражняясь в использовании этих инструментов. Будет использоваться база данных `mydb`, созданная с помощью файла сценария `table_creation.txt`, рассмотренного в этой главе ранее ([Листинг 15.5](#)).

Процедура

- Создайте базу данных `mydb` и загрузите XML-данные, как было описано ранее в этой главе. Файл `table_creation.txt` содержится в прилагаемом

файле **Expressc_book_exercises_9.7.zip** в папке Chapter 2. Выполните файл сценария `table_creation.txt` в командном окне DB2 или командном процессоре Linux следующим образом:

```
db2 -tvf table_creation.txt
```

2. Если на одном из шагов произойдет сбой сценария, попытайтесь определить проблему, ознакомившись с сообщениями об ошибках. Распространенная проблема при выполнении сценария заключается в том, что может понадобиться смена путей к файлам, которые могут храниться в других каталогах. В любой момент можно удалить и снова создать базу данных, выполнив следующую команду в командном окне DB2 или командном процессоре Linux:

```
db2 drop database mydb
```

3. Если при попытке удалить базу данных появится сообщение об ошибке в связи с наличием активных подключений, выполните следующую команду:

```
db2 force applications all
```

4. После успешного выполнения сценария воспользуйтесь Центром управления DB2 или IBM Data Studio, чтобы подтвердить создание таблиц **items** и **clients** и наличие в них 4 и 7 строк, соответственно.
5. Создав базу данных **mydb** и загрузив две таблицы, можно подключиться к ней и выполнить запросы, показанные в листингах 15.7—15.19.

A

Приложение А. Устранение неисправностей

В этом Приложении рассмотрены способы решения проблем, которые могут возникать при работе с DB2. На рис. A.1 проиллюстрирован краткий обзор действий в случае возникновения проблемы.

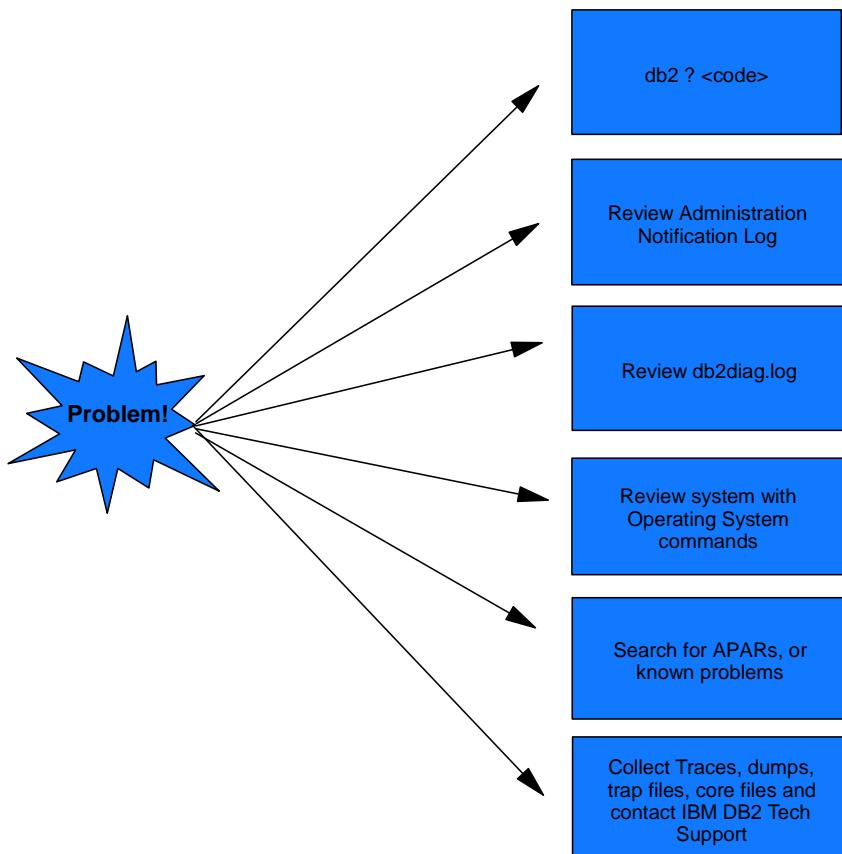


Рисунок А.1. Обзор устранения неисправностей

Примечание.

Чтобы получить более подробную информацию об устранении неисправностей, просмотрите видео:

<http://www.channeldb2.com/video/video/show?id=807741:Video:4462>

A.1 Получение более подробной информации о кодах ошибок

Чтобы более подробно узнать о полученном коде ошибки, введите код со знаком вопроса в качестве префикса в область ввода Редактора команд и нажмите кнопку *Выполнить*. Это показано на рис. А.2.

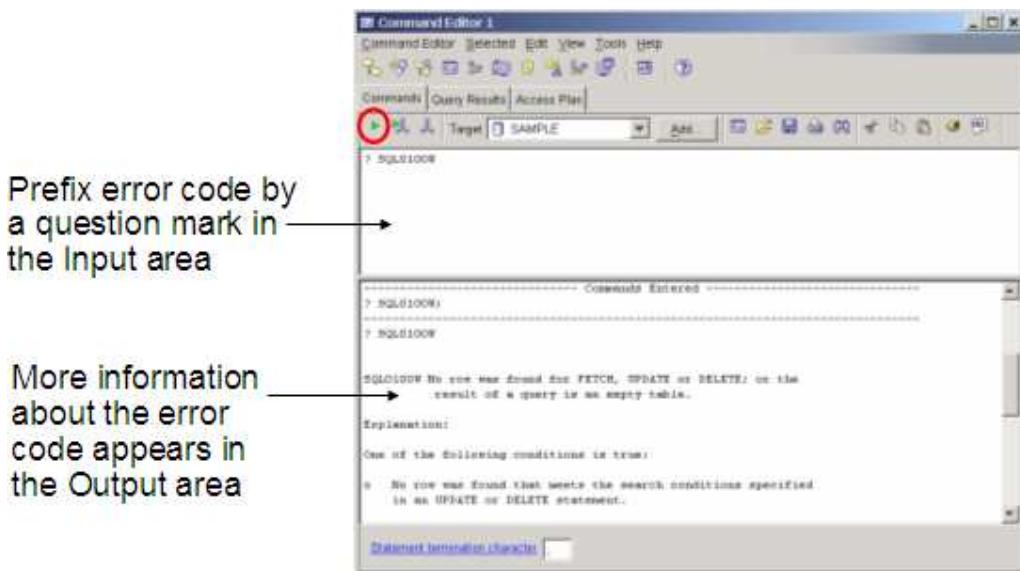


Рисунок А.2. Получение более подробной информации о кодах ошибок DB2

Вопросительный знак (?) вызывает команду справки DB2. Ниже приведено несколько примеров вызова справки при получении, к примеру, кода ошибки SQL «-104». Все примеры ниже являются равнозначными.

```
db2 ? SQL0104N
db2 ? SQL104N
db2 ? SQL-0104
db2 ? SQL-104
db2 ? SQL-104N
```

A.2 SQLCODE и SQLSTATE

SQLCODE — это код, получаемый после выполнения каждого оператора SQL. Значения подытожены ниже:

- SQLCODE = 0; команда прошла успешно
- SQLCODE > 0; команда прошла успешно, но вернула предупреждение
- SQLCODE < 0; команда прошла неудачно и вернула код ошибки

SQLSTATE — это строка из пяти символов, соответствующая стандарту ISO/ANSI SQL92. Первые два символа известны как код класса SQLSTATE:

- Код класса 00 означает, что команда выполнена успешно
- Код класса 01 указывает на предупреждение
- Код класса 02 указывает на то, что состояние не найдено
- Все остальные коды классов считаются ошибками.

A.3 Журнал административных уведомлений DB2

В журнале административных уведомлений DB2 содержится диагностическая информация об ошибках в точках сбоя. На платформах Linux и UNIX журнал административных уведомлений — это текстовый файл с именем <имя_экземпляра>.nfy (например, «db2inst.nfy»). В Windows все административные уведомления записываются в Журнал событий Windows.

Параметр конфигурирования DBM `notifylevel` дает администраторам возможность задать уровень записываемой информации:

- 0 — административные уведомления не записываются (не рекомендуется);
- 1 — критические или неисправимые ошибки;
- 2 — требуется немедленное действие;
- 3 — важная информация, немедленные действия не требуются (по умолчанию);
- 4 — информационные сообщения.

A.4 db2diag.log

В журнале db2diag.log содержится более подробная информация, чем в журнале административных уведомлений DB2. Обычно он используется службой технической поддержки DB2 компании IBM или опытными администраторами баз данных. Журнал db2diag.log содержит следующую информацию:

- местоположение кода ошибки DB2;
- идентификаторы приложения, которые позволяют сопоставить записи, относящиеся к конкретному приложению, в журналах db2diag.log серверов и клиентов;
- диагностическое сообщение (начинающееся с символов «DIA»), объясняющее причину ошибки;
- все доступные вспомогательные данные, такие как структуры данных SQLCA и указатели на местоположение дополнительных файлов дампа или прерываний (trap).

В Windows (кроме Vista) журнал db2diag.log по умолчанию расположен в каталоге:

```
C:\Documents and Settings\All Users\Application  
Data\IBM\DB2\DB2COPY1\<instance name>
```

В Windows Vista журнал db2diag.log по умолчанию расположен в каталоге:

```
C:\ProgramData\IBM\DB2\DB2COPY1\<instance name>
```

В Linux/UNIX журнал db2diag.log по умолчанию расположен в каталоге:

```
/home/<instance_owner>/sql1ib/db2dump
```

Количество деталей и диагностического текста определяется параметром конфигурации dbm cfg DIAGLEVEL. Диапазон значений — от 0 до 4, где 0 — наименьший уровень детализации деталей, 4 — наивысший. По умолчанию установлен уровень 3.

A.5 Трассировка CLI

Для устранения неисправностей в приложениях на CLI, Java, PHP и Ruby on Rails можно включить службу трассировки CLI. Для этого необходимо внести изменения в файл db2cli.ini на сервере, на котором работает приложение. В листинге A.1 ниже показаны стандартные записи файла db2cli.ini.

```
[common]  
trace=0  
tracerefreshinterval=300  
tracepathname=/path/to/writeable/directory  
traceflush=1
```

Листинг A.1. Записи в файле db2cli.ini для включения трассировки CLI

Также доступна трассировка низкого уровня (db2trc), но обычно она полезна только для службы технической поддержки DB2.

A.6 Дефекты и исправления DB2

Причиной некоторых возникающих проблем могут быть дефекты DB2. Компания IBM регулярно выпускает пакеты исправлений, содержащие исправления программного кода для устранения дефектов (APAR). Документация пакета исправлений содержит список всех включенных в пакет исправлений. Мы всегда рекомендуем использовать для разработки новых приложений последние пакеты исправлений, чтобы избежать возникновения уже решенных проблем. Чтобы просмотреть текущую версию и уровень пакета исправлений: в Центре управления выберите пункт **О программе** в меню **Справка**; или введите команду db2level в командном окне. Обратите внимание, что пакеты исправлений и официальная техническая поддержка DB2 от компании IBM не предоставляются для DB2 Express-C. Исправления для DB2 Express-C встраиваются в сам образ, а не применяются отдельно как пакеты исправлений.

B

Приложение В. Справочные материалы и ресурсы

B.1 Справочные материалы

[1] ZIKOPOULOS, P. *IBM DB2 Universal Database and the Microsoft Excel Application Developer... for Beginners*, dbazine.com article, April 2005

<http://www.dbazine.com/db2/db2-disarticles/zikopoulos15>

[2] ZIKOPOULOS, P. *DB2 9 and Microsoft Access 2007 Part 1: Getting the Data...*, Database Journal article, May 2008

<http://www.databasejournal.com/features/db2/article.php/3741221>

[3] BHOGAL, K. *Use Microsoft Access to interact with your DB2 data*, developerWorks article, May 2006. <http://www.ibm.com/developerworks/db2/library/techarticle/dm-0605bhogal/>

[4] SARACCO, C. et all. IBM Redbook *DB2 9: pureXML overview and fast start* July 2006. <http://www.redbooks.ibm.com/abstracts/sg247298.html>

B.2 Веб-сайты:

1. Веб-сайт DB2 Express-C: www.ibm.com/db2/express

На этом веб-сайте можно загрузить образы серверов DB2 Express-C, клиенты DB2 и драйверы DB2, просмотреть руководства пользователя, получить доступ к блогу команды, подписаться на рассылку и пр.

2. Форум DB2 Express:

www.ibm.com/developerworks/forums/dw_forum.jsp?forum=805&cat=19

На этом форуме можно задать технические вопросы, ответы на которые не удается найти в справочниках самостоятельно.

3. Информационный центр DB2

Информационный центр предоставляет доступ к руководствам пользователя в сети. Это источник наиболее актуальной информации. Для каждой версии DB2 существует соответствующий информационный центр DB2:

- DB2 9.1: <http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp>
- DB2 9.5: <http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp>
- DB2 9.7: <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>

4. developerWorks: <http://www.ibm.com/developerworks/db2>

Этот веб-сайт является отличным ресурсом для разработчиков и администраторов баз данных и предоставляет доступ к бесплатным учебным руководствам, актуальным статьям и другим материалам.

5. alphaWorks: <http://www.alphaworks.ibm.com/>

Этот веб-сайт предоставляет непосредственный доступ к развивающимся технологиям компании IBM. Здесь можно ознакомиться с последними технологиями от IBM Research.

6. planetDB2: www.planetDB2.com

Это агрегатор блогов многих членов сообщества, которые пишут о DB2.

7. Техническая поддержка DB2: http://www.ibm.com/software/data/db2/support/db2_9/

Тут можно найти отчеты о дефектах и проблемах (APAR), а также прочую техническую информацию.

8. ChannelDB2: <http://www.ChannelDB2.com/>

ChannelDB2 — это социальная сеть сообщества DB2. Здесь можно найти такое содержимое, как видео, демоверсии, подкасты, блоги, обсуждения, ресурсы и пр. материалы о DB2 для Linux, UNIX, Windows, z/OS и i5/OS.

B.3 Книги

1. Free Redbook: DB2 Express-C: The Developer Handbook for XML, PHP, C/C++, Java and .NET
Wei-Jen Chen, John Chun, Naomi Ngan, Rakesh Ranjan, Manoj K. Sardana,
August 2006 - SG24-7301-00
<http://www.redbooks.ibm.com/abstracts/sg247301.html?Open>
2. Free Redbook: DB2 pureXML Guide
Wei-Jen Chen, Art Sammartino, Dobromir Goutev, Felicity Hendricks, Ippei Komi,
Ming-Pang Wei, Rav Ahuja, Matthias Nicola. August 2007
<http://www.redbooks.ibm.com/abstracts/sg247315.html?Open>
3. Free Redbook: Developing PHP Applications for IBM Data Servers.
Wei-Jen Chen, Holger Kirstein, Daniel Krook, Kiran H Nair, Piotr Pietrzak
May 2006 - SG24-7218-00
<http://www.redbooks.ibm.com/abstracts/sg247218.html?Open>
4. Understanding DB2 – Learning Visually with Examples V9.5
Raul F. Chong, et all. January 2008
ISBN-10: 0131580183
5. DB2 SQL PL: Essential Guide for DB2 UDB on Linux, UNIX, Windows, i5/OS and z/OS, 2nd Edition
Zamil Janmohamed, Clara Liu, Drew Bradstock, Raul Chong, Michael Gao, Fraser McArthur, Paul Yip
ISBN: 0-13-100772-6
6. DB2 9: pureXML overview and fast start
Cynthia M. Saracco, Don Chamberlin, Rav Ahuja June 2006 SG24-7298
<http://www.redbooks.ibm.com/abstracts/sg247298.html?Open>
7. Information on Demand - Introduction to DB2 9 New Features
Paul Zikopoulos, George Baklarz, Chris Eaton, Leon Katsnelson
ISBN-10: 0071487832
ISBN-13: 978-0071487832

B.4 Контактные адреса электронной почты

Общий почтовый ящик DB2 Express-C (для административных вопросов):
db2x@ca.ibm.com

Общий почтовый ящик программы DB2 on Campus: db2univ@ca.ibm.com

Начать работу с DB2 9.7 проще простого. Прочтите эту книгу чтобы:

- Познакомиться с сервером данных DB2 используя его бесплатную редакцию — Express-C
- Понять архитектуру DB2, инструменты, вопросы безопасности
- Научиться администрировать базы данных DB2
- Создавать запросы SQL, XQuery и хранимые процедуры
- Разрабатывать приложения баз данных для DB2
- Поупражняться в выполнении практических упражнений

DB2 Express-C от компании IBM — это бесплатная редакция сервера данных DB2 для удобного управления реляционными и XML-данными. «Бесплатная» означает, что DB2 Express-C можно свободно загрузить, свободно использовать для создания ваших приложений, свободно разворачивать в промышленных системах и даже свободно встраивать и распространять с вашими решениями. Более того, DB2 не устанавливает искусственных ограничений на размеры баз данных, их количество или число пользователей.

DB2 Express-C работает в системах Windows и Linux, а также предоставляет драйвера приложений для множества языков программирования, в том числе C/C++, Java, .NET, PHP, Perl и Ruby. Доступна недорогая подписка, предоставляющая техническую поддержку и дополнительные возможности. Если требуется еще больший уровень масштабируемости или более широкая функциональность, построенные в DB2 Express-C приложения можно беспрепятственно развернуть в других редакциях DB2, таких как Workgroup и Enterprise.

Данная бесплатная редакция DB2 идеально подходит разработчикам, консультантам, независимым поставщикам ПО, администраторам баз данных, студентам и всем, кто планирует разрабатывать, тестировать, развертывать или распространять приложения баз данных. Присоединяйтесь к растущему сообществу DB2 Express-C прямо сейчас и сами испытайте возможности продукта DB2 Express-C. Сделайте первые шаги навстречу созданию приложений нового поколения и внедрению инновационных решений!

Получить более подробную информацию о DB2 Express-C или загрузить продукт можно по адресу:

ibm.com/db2/express

Пообщаться с участниками сообщества DB2, а также просмотреть посвященные DB2 видео и блоги можно по адресу:

Channeldb2.com



9 780986 628351

Цена: 24,99 долл. США