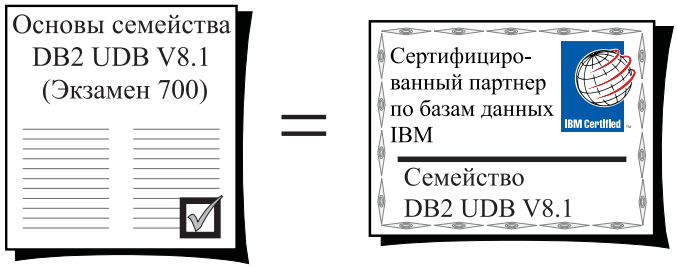
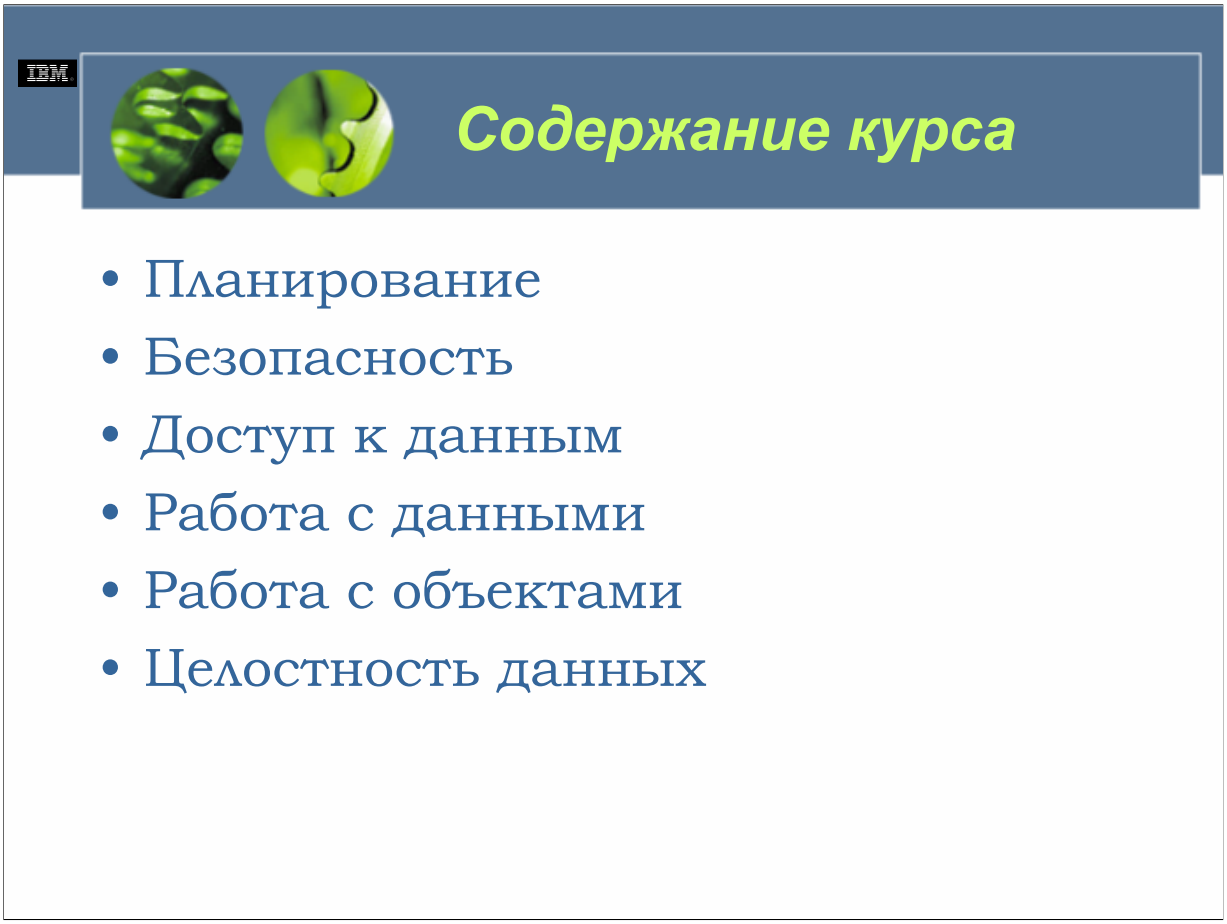
WCF03.0

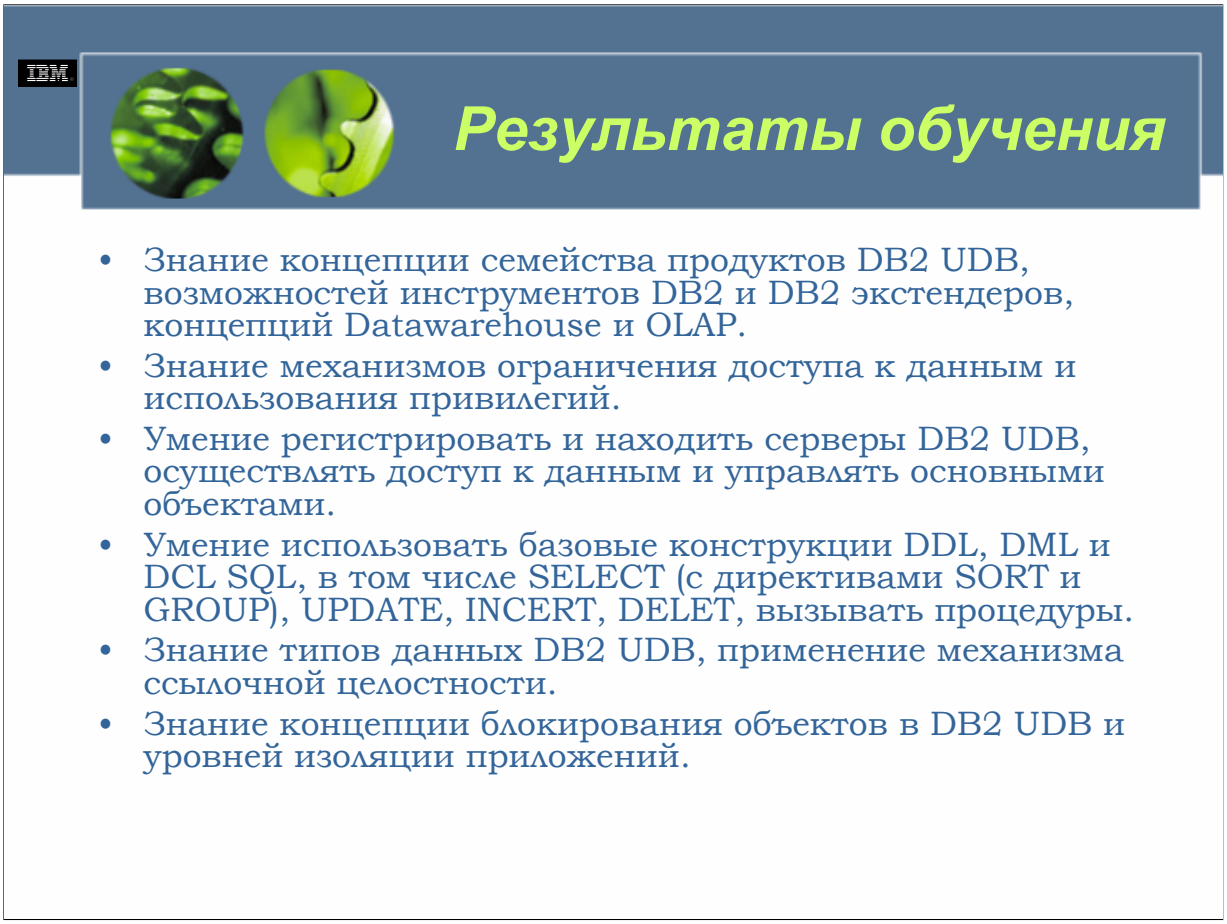
WCF03.0

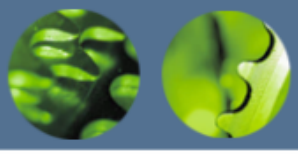
О курсе

- Продолжительность курса 1 день
- Курс завершается сдачей сертификационного экзамена
  700 *IBM DB2 UDB V8.1 Family Fundamentals*









Расскажите о себе

- Представьтесь
- и расскажите о себе
  - *Ваша кампания*
  - *Опыт работы с DB2*
  - *Что Вы хотели бы получить от курса*

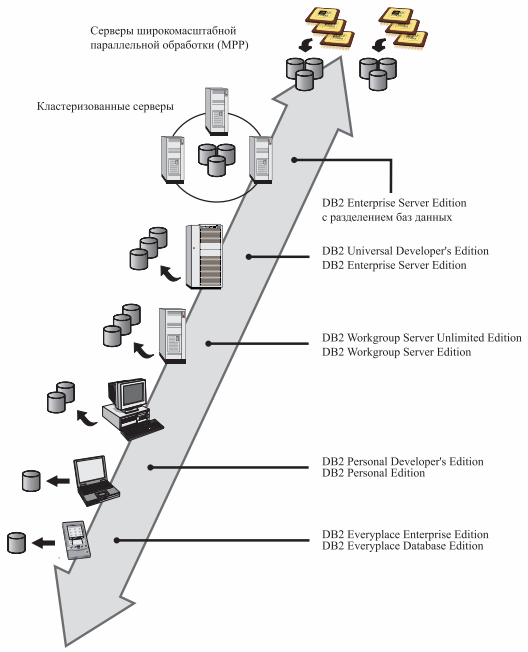
Вперед!

- Учиться, учиться и учиться…

WCF03.0

Редакции универсальной базы данных DB2

- DB2 Everyplace Database Edition
- DB2 Everyplace Enterprise Edition
- DB2 Universal Database Personal Edition
- DB2 Workgroup Server Edition
- DB2 Workgroup Server Unlimited Edition
- DB2 Enterprise Server Edition
- DB2 Personal Developer's Edition
- DB2 Universal Developer's Edition



WCF03.0

Универсальная база данных DB2, версия 8.1, является последним выпуском популярного программного обеспечения управления данными, которое впервые было представлено в 1989 г. Подобно предыдущим версиям, DB2 UDB V8.1 работает на широком множестве платформ (AIX, HP-UX, Linux, Solaris, Windows NT, Windows 2000 и Windows XP), и доступно несколько редакций – каждая из которых была спроектирована для удовлетворения определенных потребностей бизнеса. Эти редакции, вместе с широким набором добавочных продуктов, предоставляющих дополнительные возможности хранения и расширенные возможности соединений, совместно известны в виде *семейства универсальной базы данных DB2*.

Основной компонент семейства универсальной базы данных DB2 составлен из шести различных редакций DB2 UDB, предназначенных для поддержки среды база данных/пользователь все возрастающей сложности, и двух редакций разработки, содержащих богатый набор инструментов, которые могут использоваться для разработки приложений, взаимодействующих с базами данных DB2 UDB независимо от того, где они находятся. Этими редакциями являются:

•DB2 Everyplace Database Edition

•DB2 Everyplace Enterprise Edition

•DB2 Universal Database Personal Edition

•DB2 Workgroup Server Edition

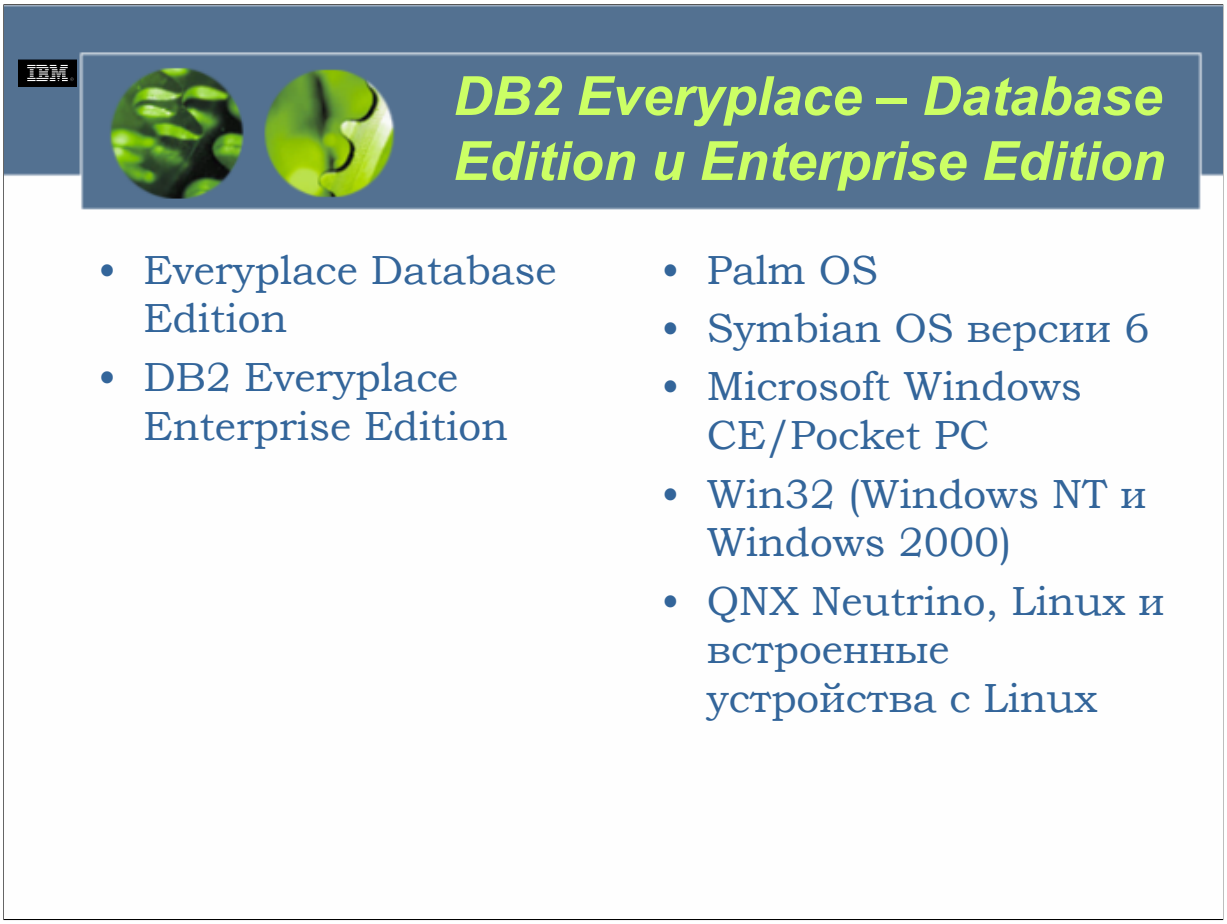
•DB2 Workgroup Server Unlimited Edition

•DB2 Enterprise Server Edition

•DB2 Personal Developer's Edition

•DB2 Universal Developer's Edition

Каждая редакция DB2 UDB использует один и тот же движок базы данных, распознает язык структурированных запросов (SQL) ANSI и предоставляет изобилие графических (GUI) инструментов, которые можно использовать как для управления, так и для взаимодействия с базами данных DB2 UDB и их объектами.



WCF03.0

DB2 Everyplace является как системой управления реляционными базами данных, так и сервером синхронизации, который позволяет получать доступ к корпоративным приложениям и данным из мобильных устройств, таких как карманные компьютеры (PDA) и портативные компьютеры (HPC). С занимаемым дисковым пространством примерно в 180 Кбайт, DB2 Everyplace можно запустить на любой из следующих операционных систем:

•Palm OS

•Symbian OS версии 6

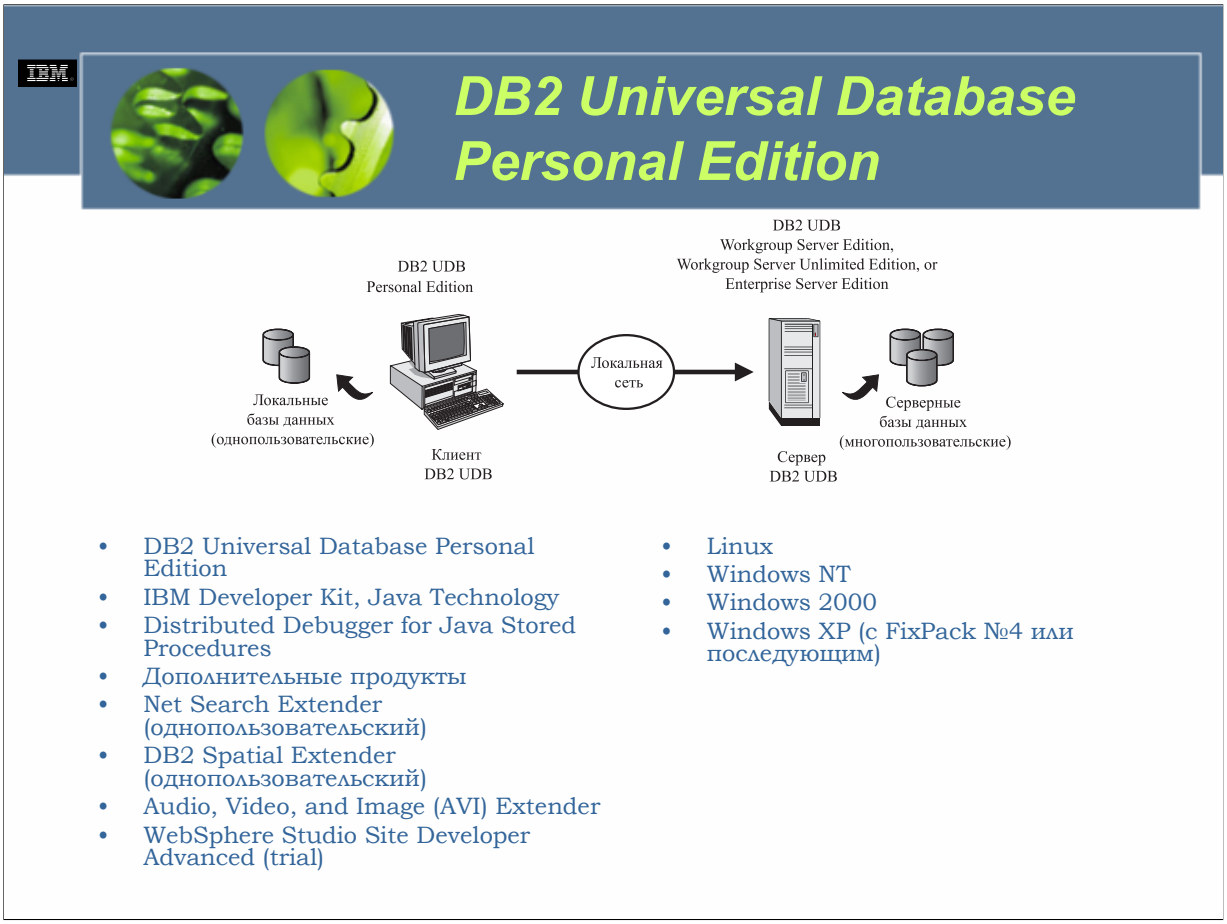
•Microsoft Windows CE/Pocket PC

•Win32 (Windows NT и Windows 2000)

•QNX Neutrino, Linux и встроенные устройства с Linux

DB2 Everyplace доступна в двух редакциях: DB2 Everyplace Database Edition и DB2 Everyplace Enterprise Edition. DB2 Everyplace Database Edition предназначена для использования производителями независимого ПО (ISV) и разработчиками приложений, желающими создавать мощные мобильные и встроенные приложения, которые работают с данными базы данных DB2 Everyplace, хранящимися непосредственно на мобильном устройстве. С другой стороны, DB2 Everyplace Enterprise Edition спроектирована в качестве полностью центрированного вокруг данных мобильного сервера синхронизации. Этот безопасный сервер отвечает за управление распределением и синхронизацией данных между пользователями мобильных устройств и серверными источниками данных, такими как DB2 UDB, Informix, Oracle, Sybase и Microsoft SQL Server. (Синхронизация осуществляется каждый раз при обнаружении подключения к серверному источнику данных).

Процессор командной строки (CLP) доступен как в DB2 Everyplace Database Edition, так и в DB2 Everyplace Enterprise Edition. Он может использоваться для выполнения стандартных операторов SQL; таким образом, с помощью DB2 Everyplace пользователи могут создавать или уничтожать объекты баз данных (такие как таблицы и индексы), а также вводить, обновлять, удалять или получать определенные значения данных. Кроме того, обе редакции DB2 Everyplace поставляются с полным набором инструментов разработки, которые могут использоваться для построения, развертывания и поддержки приложений DB2 UDB, работающих как с базами данных DB2 Everyplace, так и с корпоративными данными, хранящимися на любом поддерживаемом сервере баз данных.



WCF03.0

DB2 Universal Database Personal Edition (PE) является системой управления базами данных для единственного пользователя, которая предназначена для использования на персональных компьютерах (PC), работающих с любой из следующих операционных систем:

•Linux

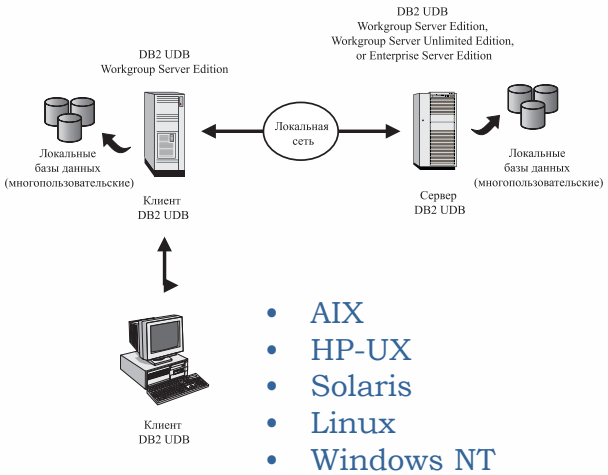
•Windows NT

•Windows 2000

•Windows XP (с FixPack №4 или последующим)

С помощью DB2 Universal Database Personal Edition пользователь может создавать, изменять и администрировать любое число баз данных; однако каждая созданная база данных должна находиться на запоминающем устройстве, управляемом PC, на котором установлено ПО DB2 UDB. Удаленные клиенты не могут получить доступ к базам данных, находящихся под управлением DB2 Universal Database Personal Edition, но PC с запущенным DB2 Universal Database Personal Edition могут действовать в качестве удаленных клиентов и получать доступ к данным, хранящимся на других серверах DB2 UDB. Более того, DB2 Universal Database Personal Edition дает базам данных под ее контролем управляться удаленными серверами DB2 UDB, что делает ее отличной редакцией для автономных или удаленных офисных реализаций, которым не требуются возможности многопользовательской работы. На рис. 2-2 показана среда базы данных, которая существует при использовании DB2 Universal Database Personal Edition.

DB2 Workgroup Server Edition



- AIX
- HP-UX
- Solaris
- Linux
- Windows NT
- Windows 2000
- Windows XP

- DB2 Administration Client
- DB2 Run-Time Client
- IBM Developer Kit, Java Technology
- Distributed Debugger for Java Stored Procedures
- Дополнительные продукты
- Net Search Extender (для пяти пользователей)
- DB2 Spatial Extender (для пяти пользователей)
- Audio, Video, and Image (AVI) Extender
- WebSphere Studio Site Developer Advanced (trial)
- WebSphere MQ
- QMF for Windows (trial)

WCF03.0

DB2 Universal Database Workgroup Server Edition (DB2 WSE) является полнофункциональной клиент/серверной системой управления базами данных, предназначенной для использования на микрокомпьютерах, имеющих до 4 центральных процессоров и работающих со следующими операционными системами:

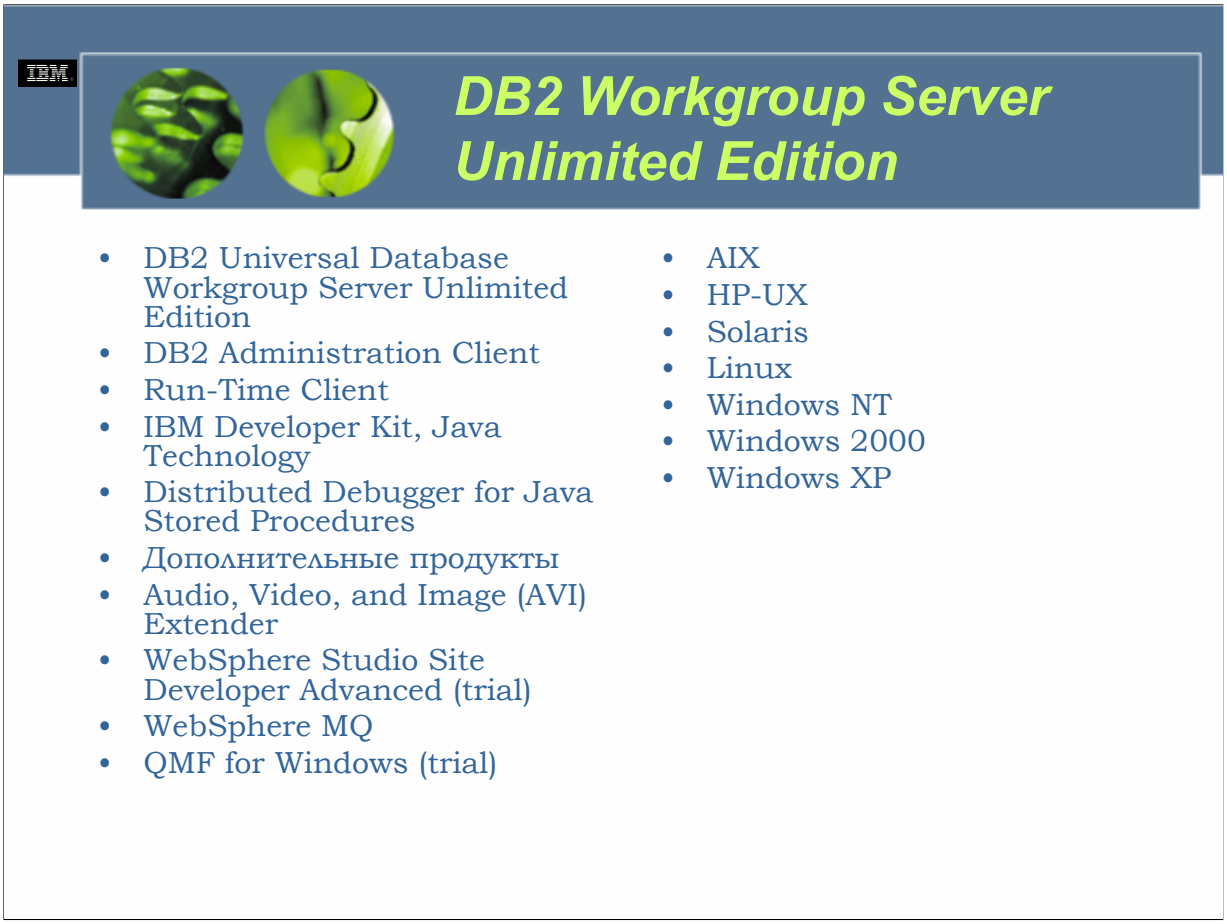
•AIX
•HP-UX
•Solaris
•Linux
•Windows NT
•Windows 2000
•Windows XP

В значительной степени DB2 Universal Database Workgroup Server Edition функционально эквивалентна DB2 Universal Database Enterprise Server Edition. Однако DB2 Universal Database Workgroup Server Edition не предоставляет интегрированной возможности соединения с мейнфреймами, и набор ее возможностей несколько ограничен (например, 64-разрядные вычисления, DataLinks и внешний основанный на веб доступ не поддерживаются). На рис. 2-3 показана среда базы данных, которая существует при установке DB2 Universal Database Workgroup Server Edition.

Удаленные клиенты могут получать доступ к базам данных, управляемым DB2 Universal Database Workgroup Server Edition, а компьютеры с запущенной DB2 Universal Database Workgroup Server Edition могут действовать в качестве удаленных клиентов для других серверов DB2 UDB. В обоих случаях клиент/серверное взаимодействие осуществляется с использованием одного из следующих коммуникационных протоколов:

•Transmission Control Protocol/Internet Protocol (TCP/IP)
•NetBIOS
•Internet Packet Exchange/Sequence Packet Exchange (IPX/SPX)
•Advanced Program-To-Program Communications (APPC)
•Именованные каналы (Named Pipes)

DB2 Universal Database Workgroup Server Edition идеальна для бизнес-окружений малого и среднего размеров, которым нужен полнофункциональный сервер баз данных, являющийся масштабируемым и доступным через локальную (LAN) или глобальную (WAN) сеть. Она полезна также для организаций, состоящих из небольшого числа внутренних пользователей, которым нужно реляционное хранилище данных.



WCF03.0

DB2 Universal Database Workgroup Server Unlimited Edition (DB2 WSUE) является в сущности DB2 Universal Database Workgroup Server Edition с упрощенной моделью лицензирования по числу процессоров в отличие от модели цен по объему (зарегистрированным пользователям) , êîòîðóþ èñïîëüçóåò DB2 Universal Database Workgroup Server Edition. И подобно DB2 Universal Database Workgroup Server Edition, DB2 Universal Database Workgroup Server Unlimited Edition может использоваться на микрокомпьютерах, работающих с любой из следующих операционных систем:

•AIX

•HP-UX

•Solaris

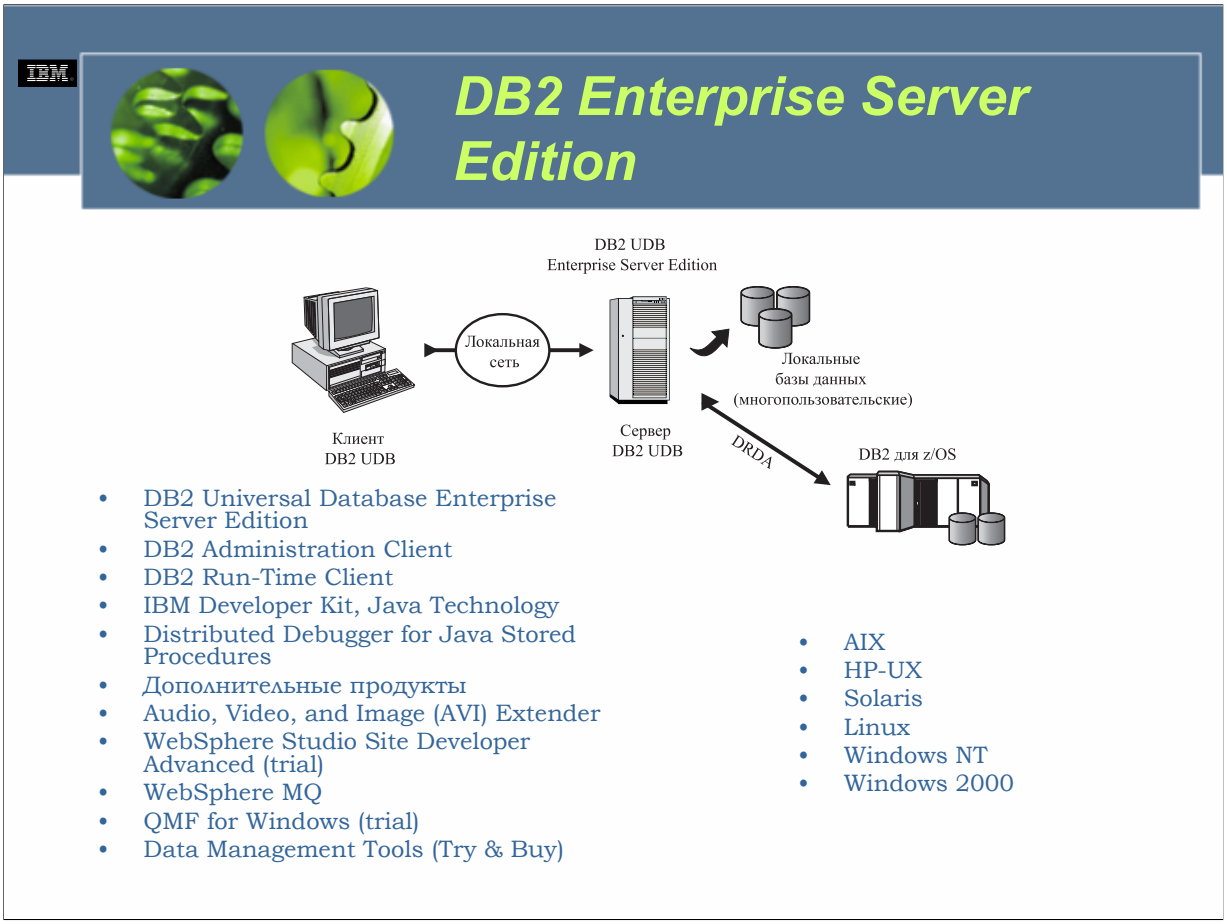
•Linux

•Windows NT

•Windows 2000

•Windows XP

Однако поскольку ее модель лицензирования другая, DB2 Universal Database Workgroup Server Unlimited Edition предоставляет внешний основанный на веб доступ . Это делает ее отличным выбором для организаций или малого бизнеса, когда нужен основанный на веб доступ к данным, и для отделов и предприятий, абсолютное число пользователей которых делает лицензирование на основе отдельных процессоров более привлекательным, чем модель лицензирования на основе объема, используемую DB2 Universal Database Workgroup Server Edition.



WCF03.0

DB2 Universal Database Enterprise Server Edition (DB2 ESE) является полнофункциональной клиент/серверной системой управления базами данных с возможностью управления через веб, спроектированной для использования на сервере любого размера, на котором работает одна из следующих операционных систем:

•AIX

•HP-UX

•Solaris

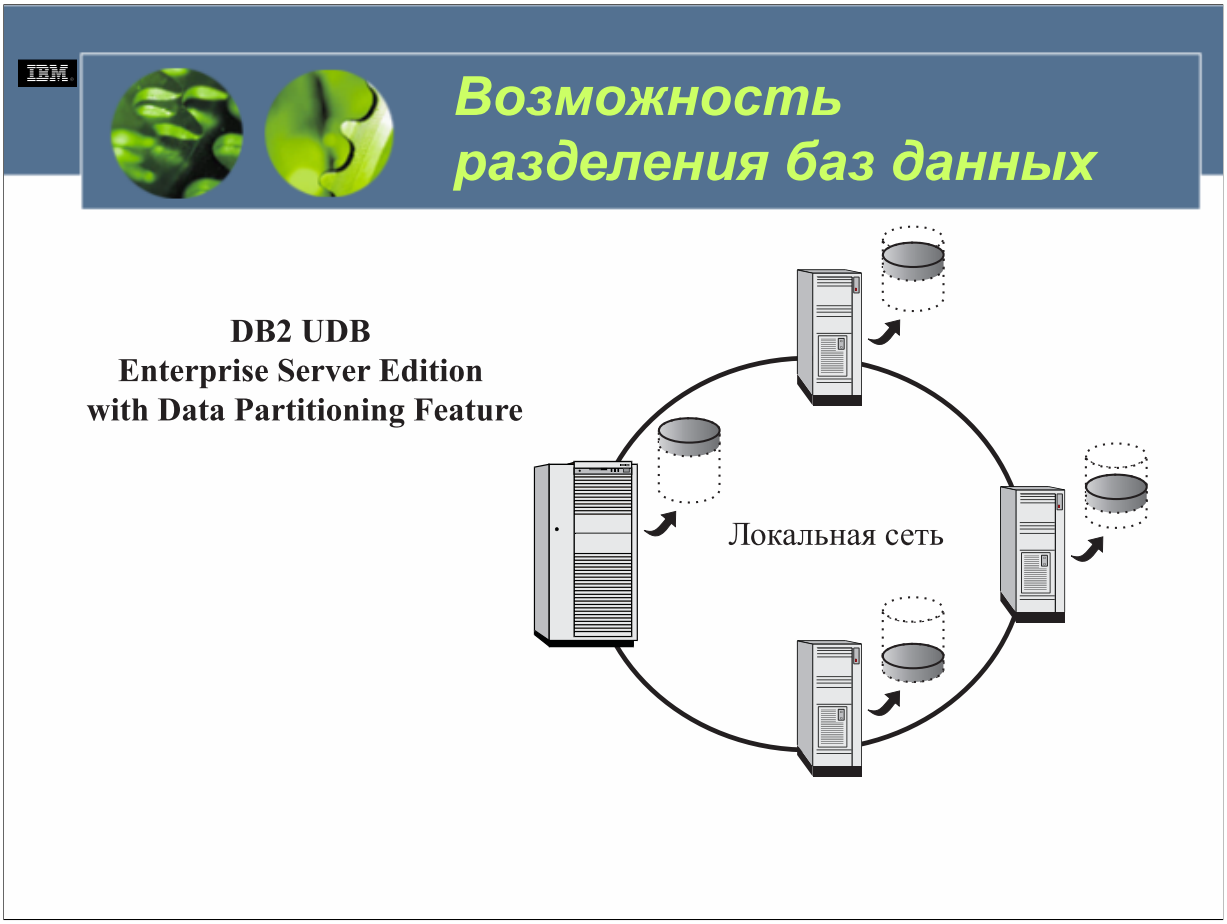
•Linux

•Windows NT

•Windows 2000

Помимо предоставления всех функциональных возможностей DB2 Universal Database Workgroup Server Edition, DB2 Universal Database Enterprise Server Edition включает в пакет тесно интегрированный продукт поддержки соединений (DB2 Connect), который позволяет ему работать в гетерогенных сетях, использующих протокол архитектуры распределенных реляционных баз данных (DRDA). Эта возможность позволяет до пяти пользователям DB2 Universal Database Enterprise Server Edition взаимодействовать с базами данных DB2 на основе iSeries и zSeries, а также ресурсами хостов, не являющимися базами данных, такими как CICS, VSAM и IMS. (Если необходима возможность подключения большего числа пользователей, вы можете приобрести дополнительные лицензии пользователей DB2 Connect). На рис. 2-4 показана среда базы данных, которая существует при установке DB2 Universal Database Enterprise Server Edition.

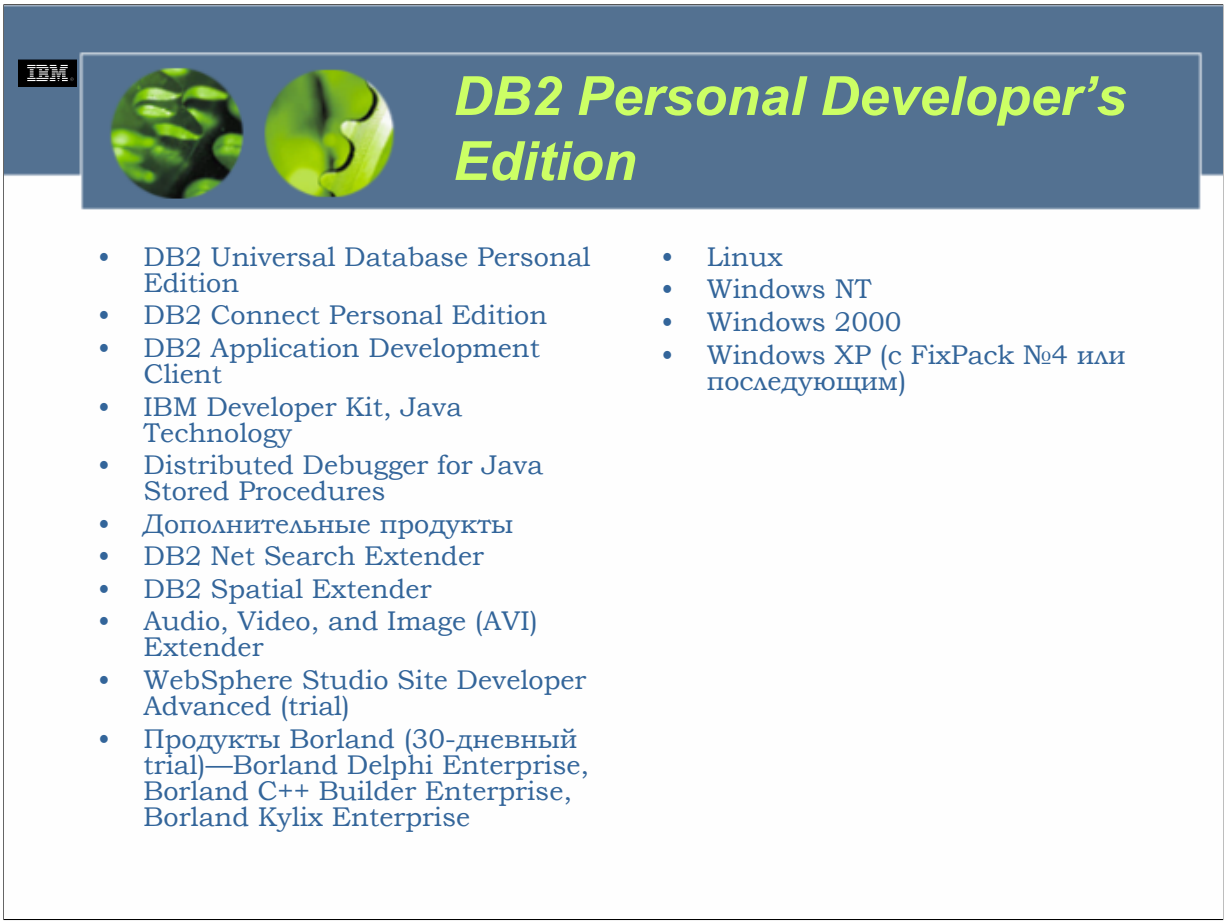
DB2 Universal Database Enterprise Server Edition предназначен для удовлетворения потребности в серверах баз данных для средних и крупных предприятий, особенно когда важна возможность соединения с Интернетом и/или сетью предприятия; его масштабируемость и надежность в сочетании с развитым набором возможностей делают его идеальным основанием для построения хранилищ данных, онлайн-систем обработки транзакций или решений на основе веб, а также в качестве отличного сервера для комплексных решений наподобие ERP, CRM или SCM.



WCF03.0

В версии 7.1 нужно было использовать специальную редакцию DB2 UDB, известную как DB2 Universal Database Enterprise Extended Edition (EEE), если вы хотели разделить единую базу данных на два или более раздела в рамках большой рабочей станции SMP или нескольких рабочих станций, работающих с одной операционной системой. В версии 8.1 возможность разделения базы данных предоставляется в виде особой возможности DB2 Universal Database Enterprise Server Edition, известной как Database Partitioning Feature (DPF).

Когда она включена, DPF предоставляет пользователям множество преимуществ, включая возможность поддержки очень больших баз данных или объемов загрузки и усиленный параллелизм для административных задач. И в то время, как DB2 Universal Database Enterprise Extended Edition был полностью самостоятельным продуктом, Database Partitioning Feature является активируемым лицензией и не требует установки дополнительных продуктов перед тем, как может быть осуществлено разделение базы данных. Поэтому если у вас уже установлена DB2 Universal Database Enterprise Server Edition и вы определяете, что было бы лучше разделить одну или более контролируемых баз данных, не нужно удалять текущую инсталляцию и устанавливать новое издание. Вместо этого вы просто приобретаете лицензию Database Partitioning Feature для каждого сервера, на котором вы планируете создать разделы баз данных.



WCF03.0

DB2 Personal Developer's Edition содержит как систему управления базами данных для отдельного пользователя, которая предоставляется DB2 Universal Database Personal Edition, так и инструментарий разработчика (SDK), который может использоваться для разработки приложений, которые взаимодействуют с однопользовательскими базами данных , управляемыми DB2 Universal Database Personal Edition. Используя инструменты, предоставляемые DB2 Personal Developer's Edition, разработчик может создавать приложения, взаимодействующие с базами данных DB2 UDB, используя широкое разнообразие доступных методов:

•Встроенный язык структурированных запросов (SQL)

•Интерфейс уровня вызовов (CLI) IBM, который совместим с интерфейсом Microsoft ODBC

•Богатый набор интерфейсов прикладного программирования (API) DB2 UDB

•Java Database Connectivity (JDBC)

•SQLJ

Инструментарий, предоставляемый DB2 Personal Developer's Edition, содержит набор библиотек и заголовочных файлов для каждого поддерживаемого языка программирования (COBOL, FORTRAN, C, C++ и Java), набор образцов программ, предназначенных для помощи в ваших собственных разработках, и прекомпилятор/компоновщик, который используется для обработки файлов исходного кода, содержащих встроенный SQL, таким образом, чтобы их можно было компилировать и компоновать с помощью обычного компилятора. Подобно DB2 Universal Database Personal Edition, DB2 Personal Developer's Edition предназначен для использования на системах персональных компьютеров (PC), на которых работает любая из следующих операционных систем:

•Linux

•Windows NT

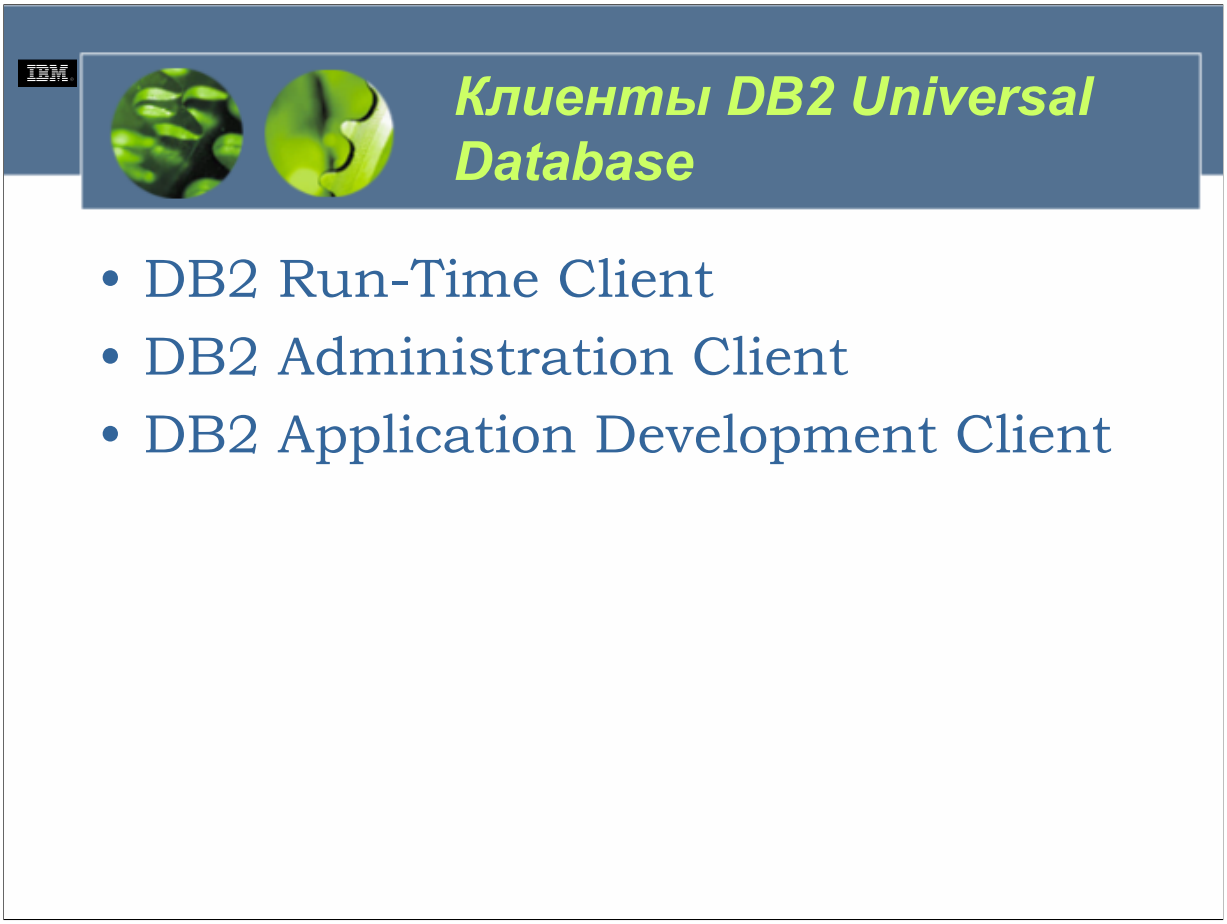
•Windows 2000

•Windows XP (с FixPack №4 или последующим)

Более того, приложения, разработанные с помощью инструментария, предоставляемого DB2 Personal Developer's Edition, могут работать на любом PC, на котором были установлены DB2 Personal Developer's Edition или DB2 Universal Database Personal Edition.

WCF03.0

DB2 Universal Developer's Edition предназначен для использования производителями независимого ПО и разработчиками приложений, которые хотят создавать решения, использующие последние доступные технологии универсальных баз данных DB2. Этот разнообразный пакет разработчика включает каждую редакцию DB2 Universal Database, за исключением DB2 Everyplace, всех трех клиентов DB2 UDB, обе редакции DB2 Connect, все модули расширений DB2, DB2 Warehouse Manager, DB2 Intelligent Miner и тот же самый инструментарий разработчика, который поставляется вместе с DB2 Personal Developer's Edition. Однако поскольку DB2 Universal Developer's Edition предоставляет разработчику приложений пакет, содержащий все инструменты, необходимые для проектирования, построения и создания опытных образцов приложений для развертывания их на любой доступной платформе клиента или сервера DB2, за сравнительно низкую цену, ни одна часть ПО, предоставленного в данном пакете, не может использоваться для установления производственной системы.



WCF03.0

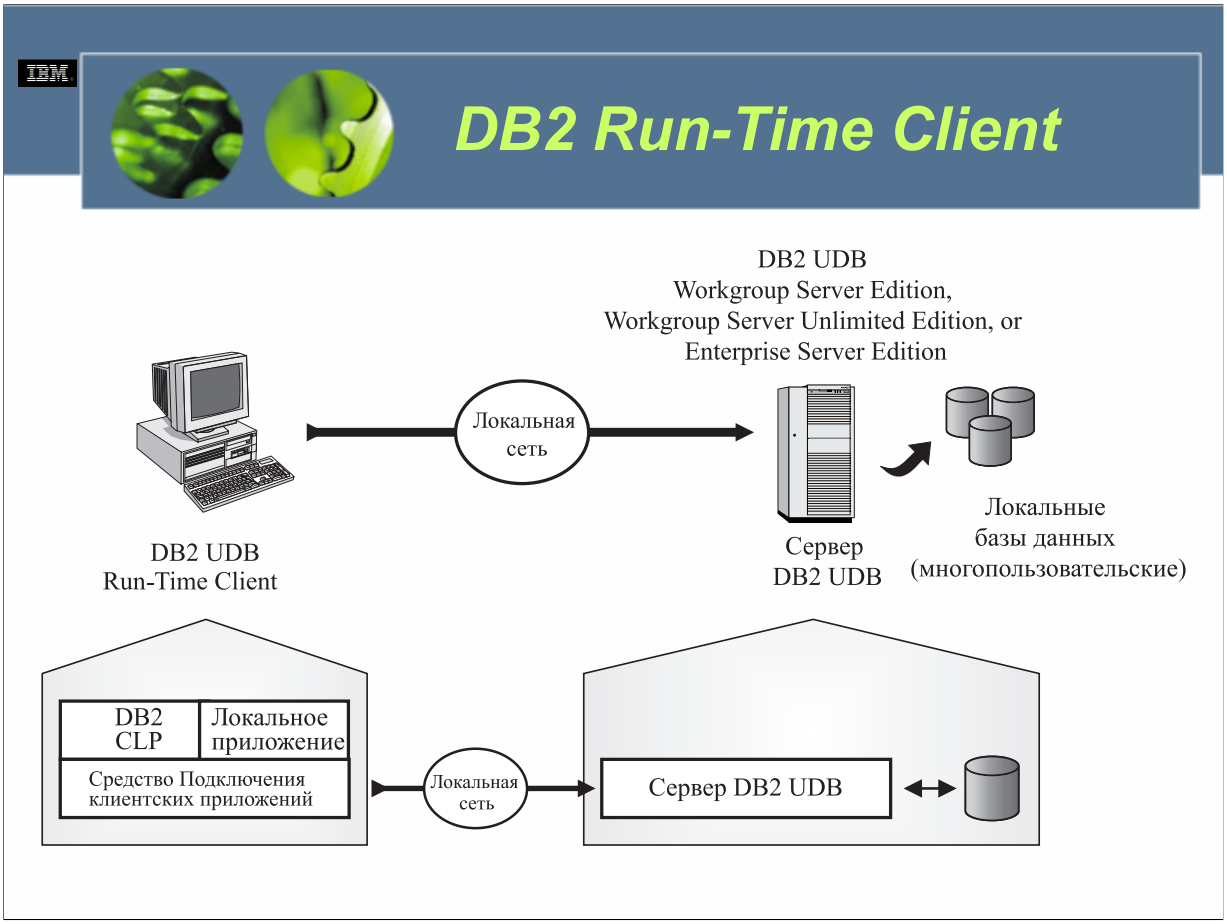
Для того, чтобы создавать клиент/серверное окружение, у вас должна быть установлена некоторая разновидность клиентского ПО на рабочей станции, которая будет служить в качестве клиента, до того, как может быть установлено соединение с сервером. Поэтому большинство редакций DB2 Universal Database содержат ПО, необходимое для установки двух или более видов клиентов. Доступны три вида клиентов DB2 Universal Database:

•DB2 Run-Time Client

•DB2 Administration Client

•DB2 Application Development Client

Любой из этих трех клиентов может быть создан (путем установки соответствующего клиентского ПО) на любом числе рабочих станций; однако вид клиента, который вы выберите для создания на данной рабочей станции, должен определяться требованиями к данной рабочей станции. Например, если определенная рабочая станция будет использоваться лишь для выполнения приложения базы данных, которое взаимодействует с базой данных, хранящейся на сервере DB2 UDB, она должна быть настроена как DB2 Run-Time Client (посредством установки ПО DB2 Run-Time Client для соответствующей операционной системы, установленной на рабочей станции). Как вы можете видеть из этого примера, чтобы знать, какого клиента создавать в данной ситуации, нужно понимать, чем отличаются каждый из трех данных видов клиента и когда каждый из них должен использоваться.



WCF03.0

DB2 Run-Time Client предоставляет рабочим станциям, работающим с различными операционными системами, возможность получать доступ к базам данных DB2 UDB, управляемым одним или более серверами DB2 UDB и/или DB2 Connect. Этот клиент является облегченным клиентом, предоставляющим базовую возможность взаимодействия, что дает пользователям клиентской рабочей станции возможность использовать интерактивные операторы SQL, которые получают доступ к данным, хранящимся на сервере DB2 UDB/DB2 Connect. Аналогичным образом DB2 Run-Time Client позволяет приложениям, использующим ODBC/CLI, JDBC, SQLJ и OLE DB, работать на клиентской рабочей станции и взаимодействовать с данными, хранящимися на сервере DB2 UDB/DB2 Connect.

ПО DB2 Run-Time Client доступно для следующих операционных систем:

•AIX

•HP-UX

•Solaris

•Linux

•Windows NT

•Windows 2000

WCF03.0

Подобно DB2 Run-Time Client, DB2 Administration Client предоставляет рабочим станциям, работающим с широким разнообразием операционных систем, возможность получить доступ к базам данных DB2 UDB, управляемым одним или более сервером DB2 UDB и/или DB2 Connect. Однако если DB2 Run-Time Client предоставляет базовые возможности взаимодействия, DB2 Administration Client предоставляет базовую возможность соединения и возможность выполнения административных операций с базами данных DB2 UDB, которые находятся на сервере DB2 UDB/DB2 Connect, с клиентской рабочей станции. Для осуществления этого DB2 Administration Client поставляется с богатым набором административных инструментов (центром управления и ассистентом конфигурирования), а также с поддержкой для тонких клиентов.

ПО DB2 Administration Client доступно для следующих операционных систем:

•AIX

•HP-UX

•Solaris

•Linux

•Windows NT

•Windows 2000

WCF03.0

Аналогично другим клиентам, DB2 Application Development Client предоставляет рабочим станциям, работающим с разнообразными операционными системами, возможность получать доступ к базам данных DB2 UDB, управляемым одним или более серверами DB2 UDB и/или DB2 Connect. Однако в отличие от других клиентов DB2 Application Development Client предоставляет также наборы как графических, так и не графических инструментов и компонентов, которые можно использовать для разработки, тестирования и запуска приложений, предназначенных для взаимодействия с базами данных, находящимися на серверах DB2 UDB/DB2 Connect. Как можно было бы представить, инструменты разработки приложений, предоставляемые DB2 Application Development Client, те же самые, которые предоставляются редакциями DB2 Developer (центр разработки, набор библиотек и заголовочных файлов для каждого поддерживаемого языка программирования, набор примеров программ в качестве образца для ваших собственных разработок и прекомпилятор/компоновщик, который используется для обработки файлов исходного кода, содержащих встроенный SQL, чтобы их можно было компилировать и компоновать обычным компилятором). Application Development Client включает также инструменты и компоненты, предоставляемые продуктом DB2 Administration Client.

ПО DB2 Application Development Client доступно для следующих операционных систем:

•AIX

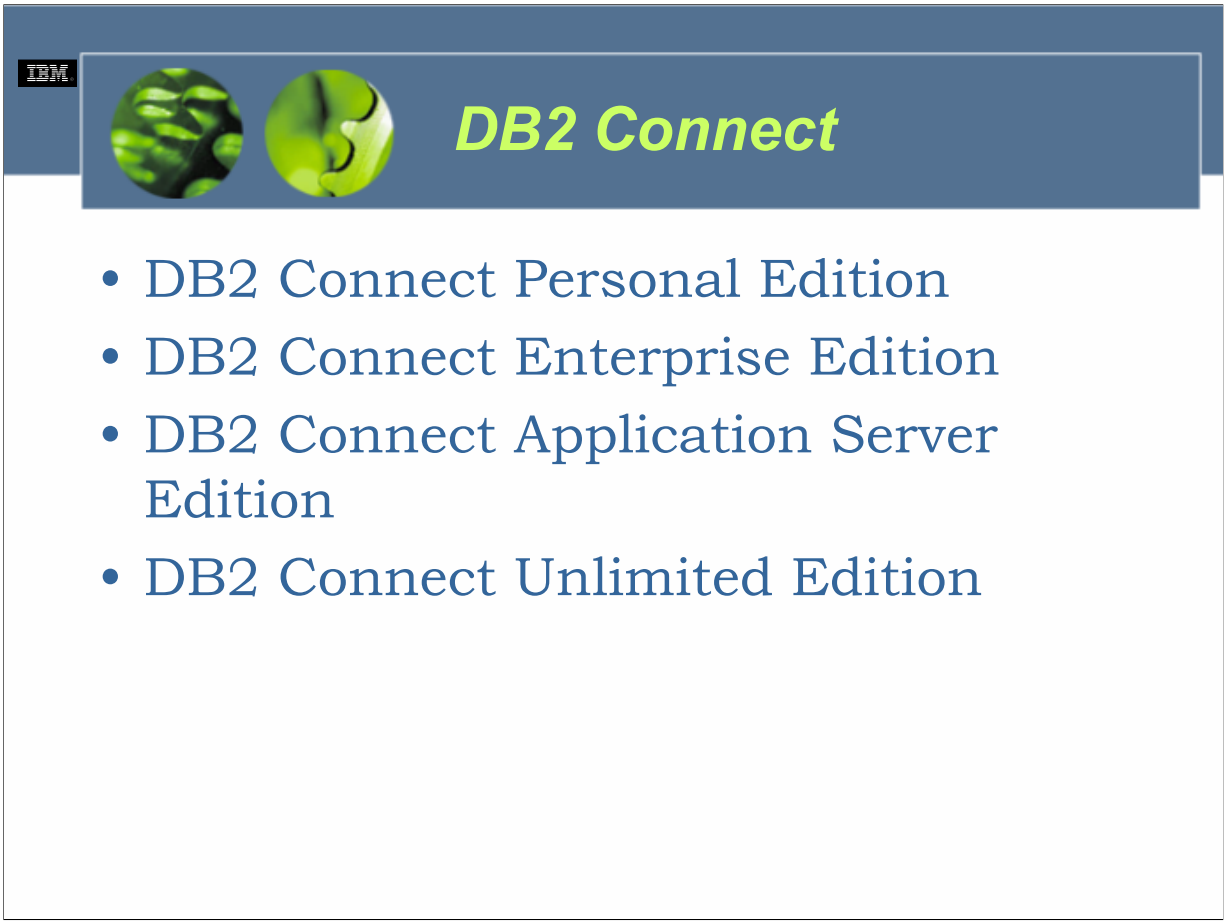
•HP-UX

•Solaris

•Linux

•Windows NT

•Windows 2000



WCF03.0

DB2 Connect Personal Edition (PE). DB2 Connect Personal Edition предоставляет непосредственную возможность соединения между рабочими станциями Linux и Windows и серверами баз данных мейнфреймов/iSeries. Она спроектирована для традиционного двухуровневого клиент-серверного окружения, в котором каждое клиентское приложение непосредственно соединяется с нужной базой данных на сервере. Данная редакция DB2 Connect не принимает входящие запросы данных и не может использоваться с многопользовательскими системами и многопользовательскими серверами приложений.

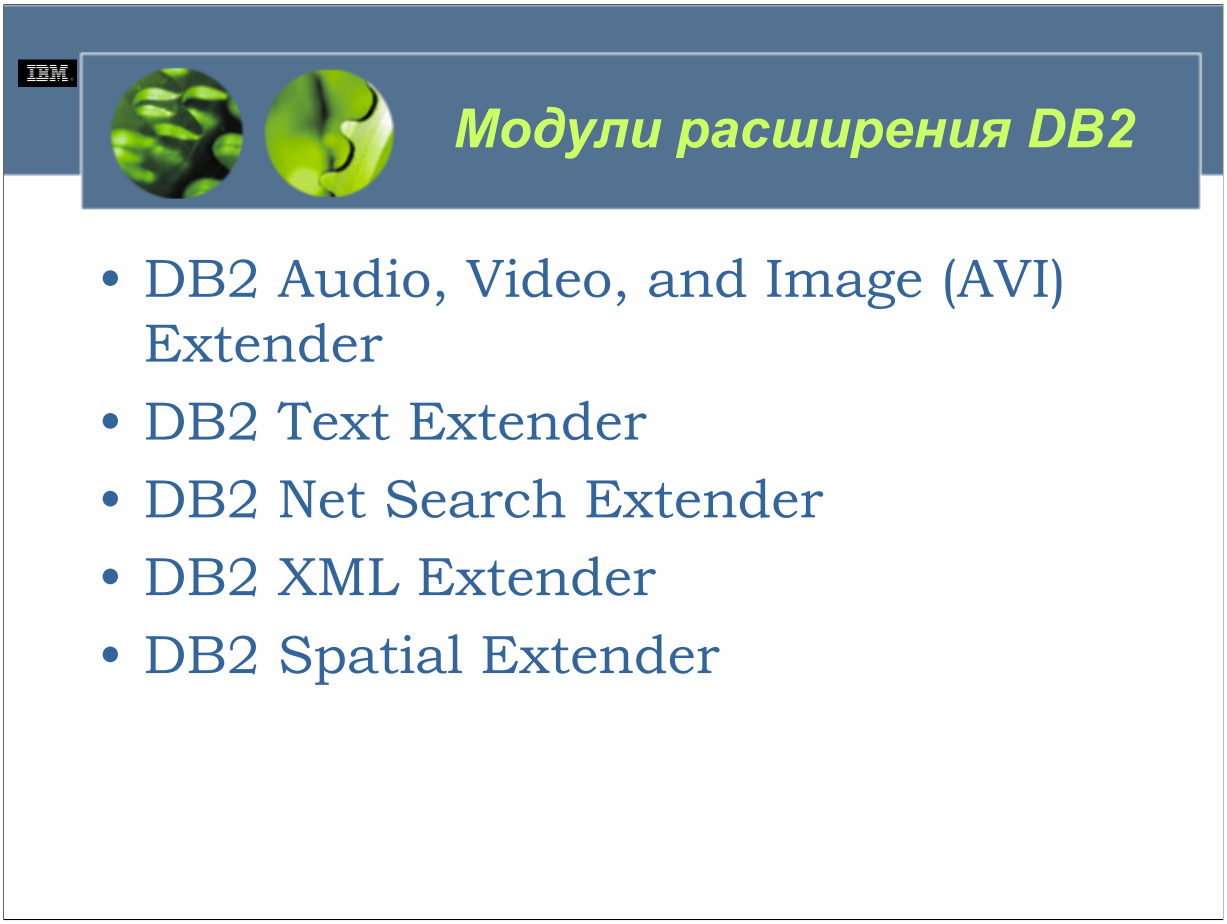
DB2 Connect Enterprise Edition (EE). DB2 Connect Enterprise Edition предоставляет возможность непосредственного соединения между рабочими станциями Linux, UNIX и Windows и серверами баз данных мейнфреймов/iSeries. Она спроектирована для двухуровневых клиент-серверных приложений, где желательно использование промежуточного сервера шлюза. В отличие от DB2 Connect Personal Edition, DB2 Connect Enterprise Edition принимает входящие клиентские запросы и подходит для использования в окружении, в котором легко можно определить число одновременно работающих пользователей или когда число зарегистрированных пользователей сравнительно невелико. Однако эта редакция DB2 Connect не подходит для окружения, в котором используются многоуровневые клиент-серверные приложения или приложения на основе веб, поскольку определение числа одновременно работающих пользователей в такой среде непрактично, а лицензирование каждого зарегистрированного пользователя может быть экономически невыгодно.

DB2 Connect Application Server Edition (ASE). DB2 Connect Application Server предоставляет те же самые функциональные возможности, что и DB2 Connect Enterprise Edition; данная редакция предназначена для предоставления более дешевой альтернативы DB2 Connect Unlimited Edition для окружения, в котором одно или более многоуровневых приложений с регулярной частотой будут получать доступ к корпоративным данным, а рост числа пользователей этих приложений с течением времени ожидается медленный или минимальный.

DB2 Connect Unlimited Edition (UE). DB2 Connect Unlimited Edition по существу предоставляет неограниченное число лицензий как DB2 Connect Personal Edition, так и DB2 Connect Enterprise Edition; данная редакция предназначена для предоставления ценовой альтернативы для окружения, в котором либо большой объем доступа к корпоративным данным необходим на сегодня, либо большой объем доступа к мейнфреймам будет необходим когда-нибудь в будущем.

WCF03.0

DB2 Relational Connect работает в сочетании с DB2 Universal Database Enterprise Server Edition для предоставления собственного доступа для чтения к базам данных Informix IDS, Oracle, Sybase и Microsoft SQL Server. По существу DB2 Relational Connect дает вам возможность комбинировать данные, хранящиеся в других СУБД, с данными, которые управляются DB2 UDB; различия между функциями и типами данных DB2 UDB отображаются таким образом, что набор различных баз данных можно видеть и обрабатывать так, как если бы они управлялись одним ресурсом. DB2 Relational Connect является добавочным продуктом, который должен приобретаться отдельно. Обычно он используется для создания баз данных объединения или для обеспечения доступности данных, хранящихся в другом продукте СУБД из центра хранилищ данных DB2 .



WCF03.0

Как предполагает название, DB2 Audio, Video, and Image Extender содержит набор пользовательских типов данных и функций, которые позволяют хранить и обрабатывать в базе данных DB2 UDB нетрадиционные данные, такие как аудио клипы, фильмы и изображения. Типы данных и функции, предоставляемые DB2 Audio, Video, and Image Extender могут использоваться в операторах SQL точно так же, как любые из встроенных типов данных и функций. А поскольку SQL может использоваться для построения запросов данных нескольких типов, этот модуль расширения обеспечивает большую гибкость при поиске информации. Например, может быть написан запрос для обнаружения определенного фильма путем поиска его описания, даты записи или общего времени проигрывания. Кроме того, возможность запроса по содержанию изображения (QBIC), предоставляемая данным модулем расширения, может использоваться для обнаружения изображений, имеющих определенную комбинацию цветов или имеющих цвета и/или текстуры, сходные с таковыми для другого изображения.

WCF03.0

DB2 Text Extender содержит набор пользовательских типов данных, которые могут хранить в базе данных DB2 UDB сложные текстовые документы, и набор пользовательских функций, которые могут извлекать ключевую информацию из таких документов независимо от того, где они хранятся (текстовые документы могут храниться либо в базе данных DB2 UDB, либо в файловой системе, к которой может получить доступ менеджер баз данных DB2). Сила этого модуля расширения происходит от мощной технологии IBM по лингвистическому поиску и анализу текстов; эта технология позволяет пользователям конструировать запросы, которые будут осуществлять поиск в любом виде текстового документа, включая документы большинства текстовых процессоров, следующих элементов:

•Определенное слово.

•Определенную фразу.

•Определенную последовательность слов.

•Разновидности слова (такие, как формы множественного числа слова или слово в другом времени).

•Синонимы определенного слова.

•Сходно звучащие слова.

•Слова со сходным написанием.

•Слова, имеющие определенный паттерн (например, все слова, начинающиеся с букв “data”).

WCF03.0

DB2 Net Search Extender предоставляет разработчикам приложений, использующим Net.Data, Java или интерфейс уровня вызовов (CLI) DB2, способ интеграции в свои приложения возможностей поиска, предоставляемые DB2 Text Extender. Поскольку DB2 Net Search Extender аналогичен, но более производителен по сравнению с DB2 Text Extender, он может быть особенно полезен при использовании с Интернет-приложениями, в которых может быть критичной производительность поиска для больших индексов и необходима возможность масштабирования обработки одновременных запросов. Ключевые особенности, которые предоставляет DB2 Net Search Extender, включают следующее:

•Возможность создавать несколько индексов для одного столбца (индексирование продолжается без запроса блокировок на уровне строк).

•Возможность создавать индексы на нескольких процессорах.

•Возможность поиска определенного слова или фразы.

•Возможность поиска слов, имеющих сходное написание.

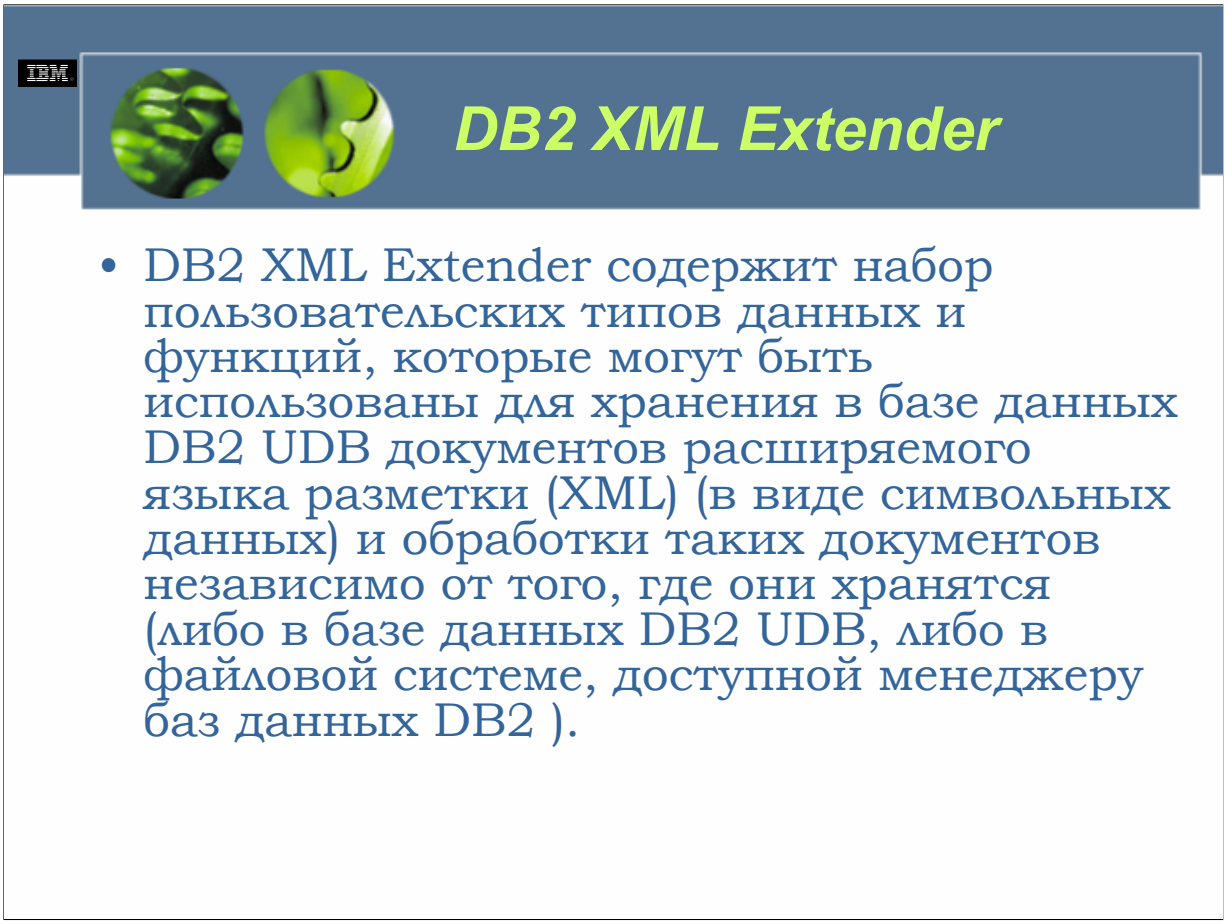
•Возможность осуществления подстановочных поисков (например, поиска всех слов, начинающихся с букв “net”).

•Возможность контролировать то, как сортируются результаты поиска.

•Возможность ограничения числа возвращаемых результатов поиска.

•Возможность поиска тэгов или разделов (с использованием булевых операторов или без них).

Важно отметить, что в отличие от других модулей расширения DB2, которые предоставляют свои функциональные возможности посредством набора пользовательских типов данных и пользовательских функций, DB2 Net Search Extender предоставляет свои возможности посредством набора хранимых процедур.



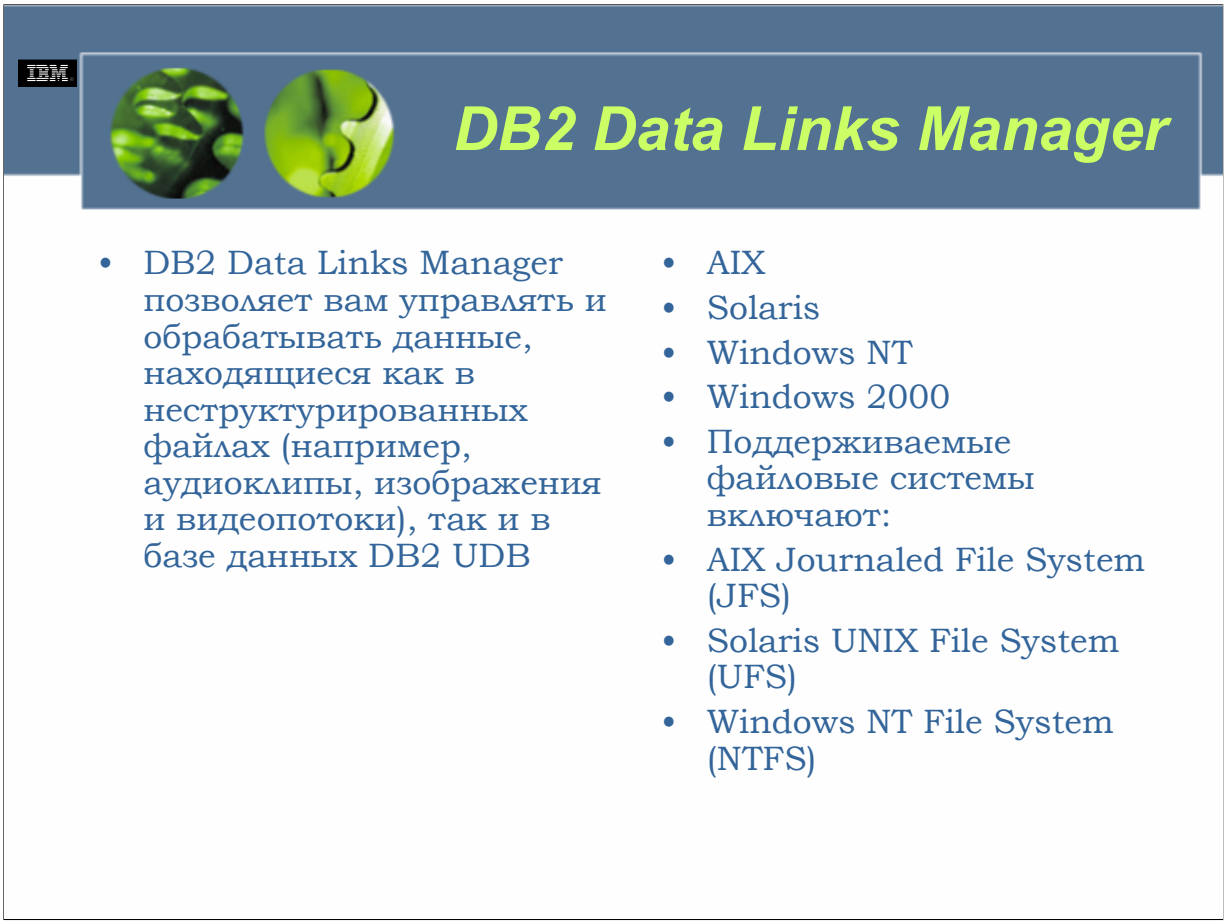
WCF03.0

DB2 XML Extender содержит набор пользовательских типов данных и функций, которые могут быть использованы для хранения в базе данных DB2 UDB документов расширяемого языка разметки (XML) (в виде символьных данных) и обработки таких документов независимо от того, где они хранятся (либо в базе данных DB2 UDB, либо в файловой системе, доступной менеджеру баз данных DB2). DB2 XML Extender может использоваться для анализа (извлечения) из документа элементов XML и сохранения их в столбцах и таблицах; он может также составлять (создавать) новые документы XML из существующих символьных и числовых данных или ранее извлеченных данных XML. И поскольку DB2 XML Extender доступны те же самые мощные возможности поиска, которые предоставляются DB2 Text Extender, внутри набора XML документов можно быстро обнаруживать специфические элементы.

WCF03.0

Традиционно геопространственные данные управлялись специализированными Географическими информационными системами (GIS), пространственные данные которых из-за их дизайна было невозможно интегрировать с производственными данными, хранящимися в других системах управления реляционными базами данных и/или источниках данных. Однако вскоре после того, как была выпущена версия 5.0 DB2 Universal Database, IBM совместно с Исследовательским институтом систем окружающей среды (ESRI), ведущим производителем систем пространственных баз данных, создали набор пользовательских типов данных для описания пространственных данных (например, точек, линий и многоугольников) и набор пользовательских функций для запросов пространственных объектов (например, для нахождения области, конечных точек и пересечений). Этот набор пользовательских типов данных и функций составляют DB2 Spatial Extender.

DB2 Spatial Extender может использоваться для хранения, создания и анализа пространственной информации о географических особенностях. С помощью данного модуля расширения пространственные данные можно хранить вместе с непространственной производственной информацией в той же самой базе данных DB2 UDB и представлять в трехмерном формате. Эта возможность позволяет предприятиям принимать геопространственные business-intelligence решения без необходимости физического перемещения данных из одного места в другое.



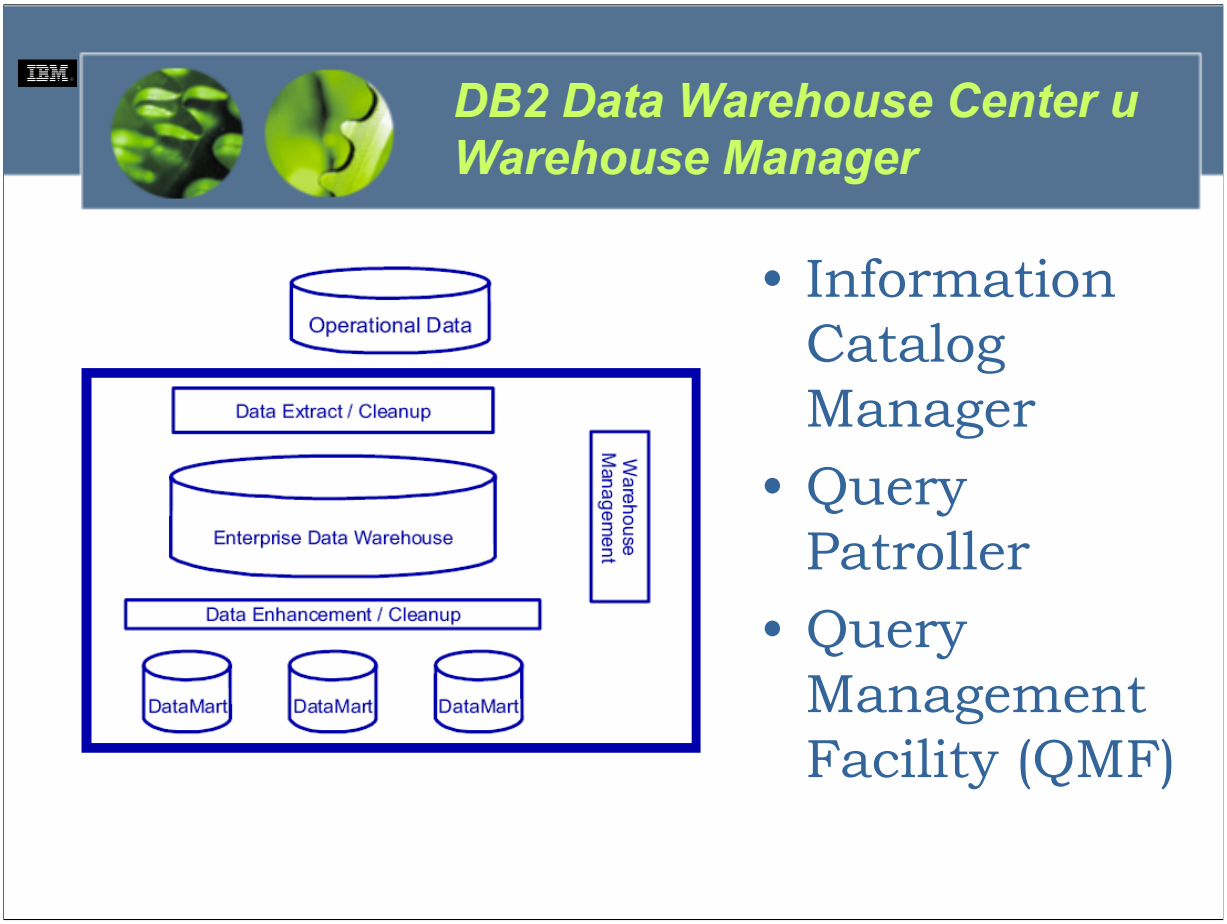
WCF03.0

DB2 Data Links Manager позволяет вам управлять и обрабатывать данные, находящиеся как в неструктурированных файлах (например, аудиоклипы, изображения и видеопотоки), так и в базе данных DB2 UDB. Файлы, которые хранятся вне базы данных DB2 UDB, находятся в файловых системах, доступных менеджеру баз данных DB2 , и управляются таким образом, как если бы они хранились внутри базы данных; DB2 Data Links Manager берет управление над файловой системой и позволяет DB2 UDB обеспечить расширенный контроль доступа над всеми файлами, находящимися в файловой системе, гарантируя поддержание реляционной целостности и то, что файлы становятся частью операций резервирования и восстановления (которые являются решающими для управления данными в транзакционном окружении). И поскольку файлы, связанные с базой данных DB2 менеджером DB2 Data Links, резервируются асинхронно каждый раз, когда ссылка на файл сохраняется в столбце DATALINK, созданный резервный образ данных меньше и создается быстрее, чем если бы данные файла сохранялись непосредственно в самой базе данных (например, в виде LOB).

DB2 Data Links Manager может также максимизировать производительность приложения и уменьшить сетевой трафик, стратегически сохраняя внешние файлы вблизи места, где они будут нужны. Более того, в существующих приложениях DB2 UDB требуются незначительные изменения или они вовсе не требуются; часто они могут использовать преимущества DB2 Data Links Manager, как только он будет установлен.

DB2 Data Links Manager является добавочным продуктом, который должен приобретаться отдельно. Он доступен для следующих операционных систем:

•AIX
•Solaris
•Windows NT
•Windows 2000
•Поддерживаемые файловые системы включают:
•AIX Journaled File System (JFS)
•Solaris UNIX File System (UFS)
•Windows NT File System (NTFS)



WCF03.0

Накопление и хранение данных является процессом, который осуществляется почти каждым предприятием. Однако для принятия производственных решений, основывающихся на накопленных данных, у вас должны быть нужные инструменты, а что еще важнее, данные должны быть в формате, подходящем для анализа. Часто это означает, что данные должны извлекаться из системы, в которой находятся, очищаться, преобразовываться, а затем загружаться в одно или более хранилищ данных (или рынки данных), которые сами должны регулярно обновляться и управляться.

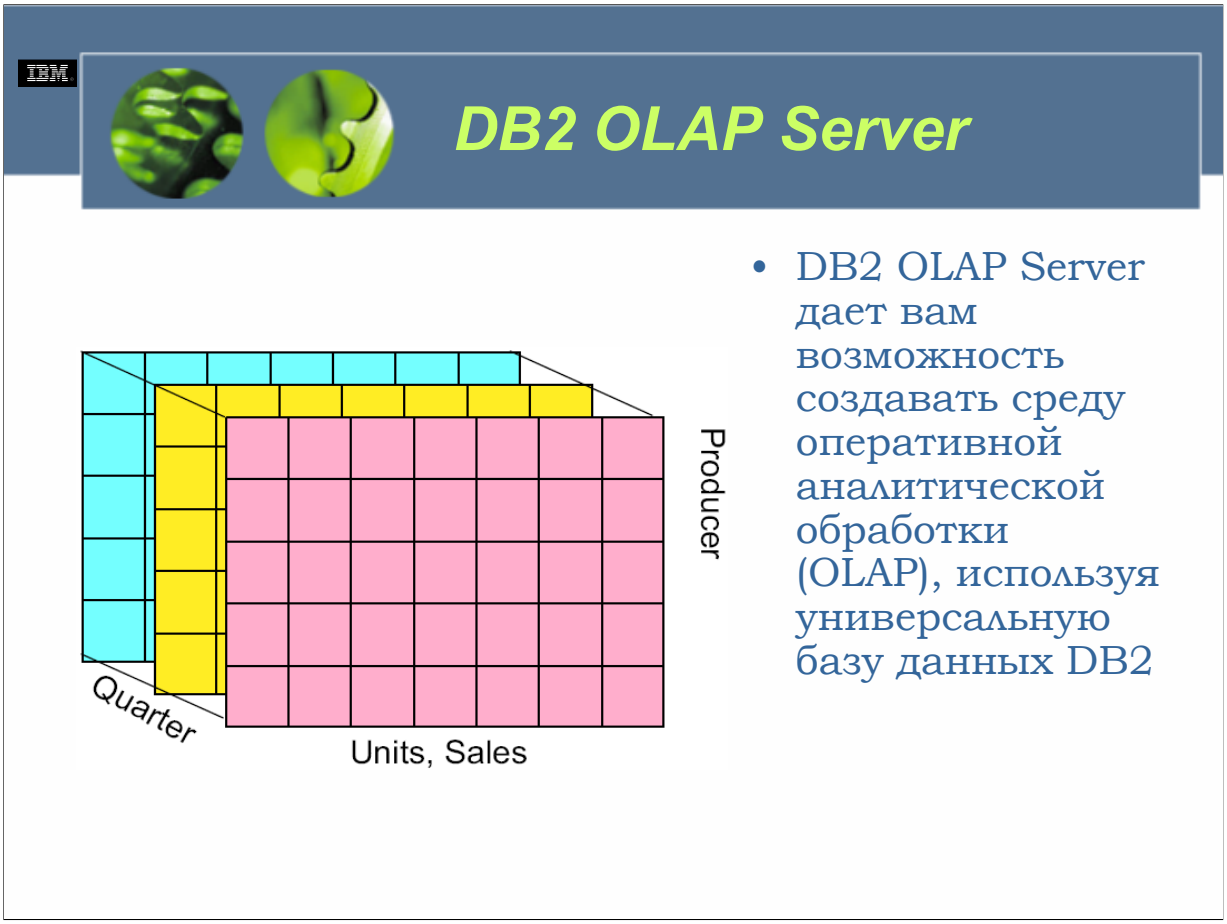
DB2 Data Warehouse Center (центр хранилищ данных) и DB2 Warehouse Manager предназначены, чтобы помочь вам создавать и поддерживать хранилища данных DB2 UDB. DB2 Data Warehouse Center, который является добавочным продуктом, который должен приобретаться отдельно, если не установлена DB2 Universal Database Enterprise Server Edition, имеет возможность извлекать, преобразовывать и загружать данные в хранилище данных. Более того, DB2 Data Warehouse Center имеет возможность перемещать данные между несколькими исходными и целевыми системами без необходимости прохождения их через централизованный сервер. Это дает возможность конструировать хранилища данных более эффективным образом.

DB2 Warehouse Manager предоставляет расширенные возможности по извлечению, преобразованию и загрузке (ETL), которых нет в DB2 Data Warehouse Center. (Фактически DB2 Warehouse Manager содержит одну из самых мощных в промышленности распределенных систем планирования заданий ETL). Он предоставляет также опции перемещения данных с полным и постепенным обновлением, и он может использовать преимущества интегрированных функций репликации данных IBM. DB2 Warehouse Manager является добавочным продуктом, который должен приобретаться отдельно, он комплектуется тремя дополнительными инструментами, которые предназначены, чтобы помочь упростить управление данными хранилища. Эти инструменты:

•Information Catalog Manager

•Query Patroller

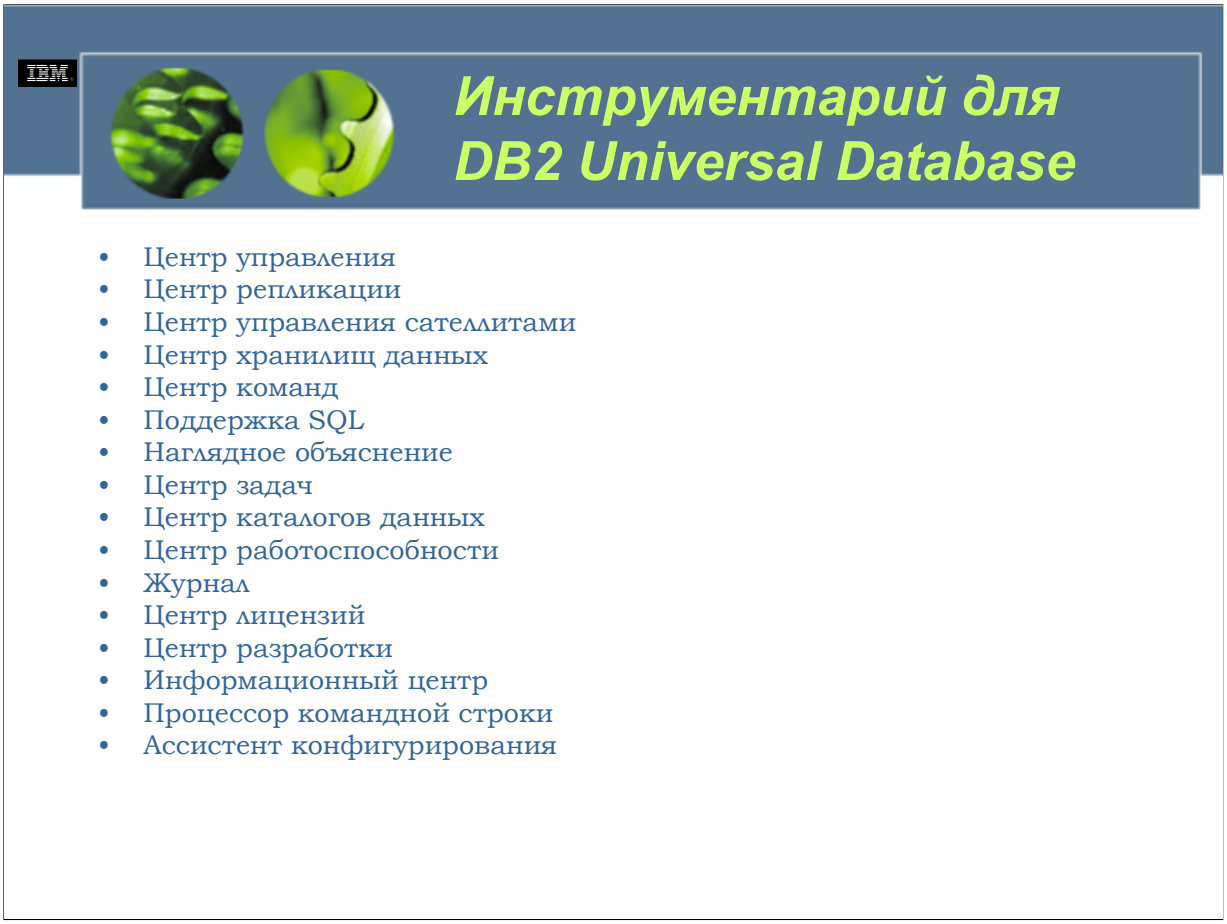
•Query Management Facility (QMF)



WCF03.0

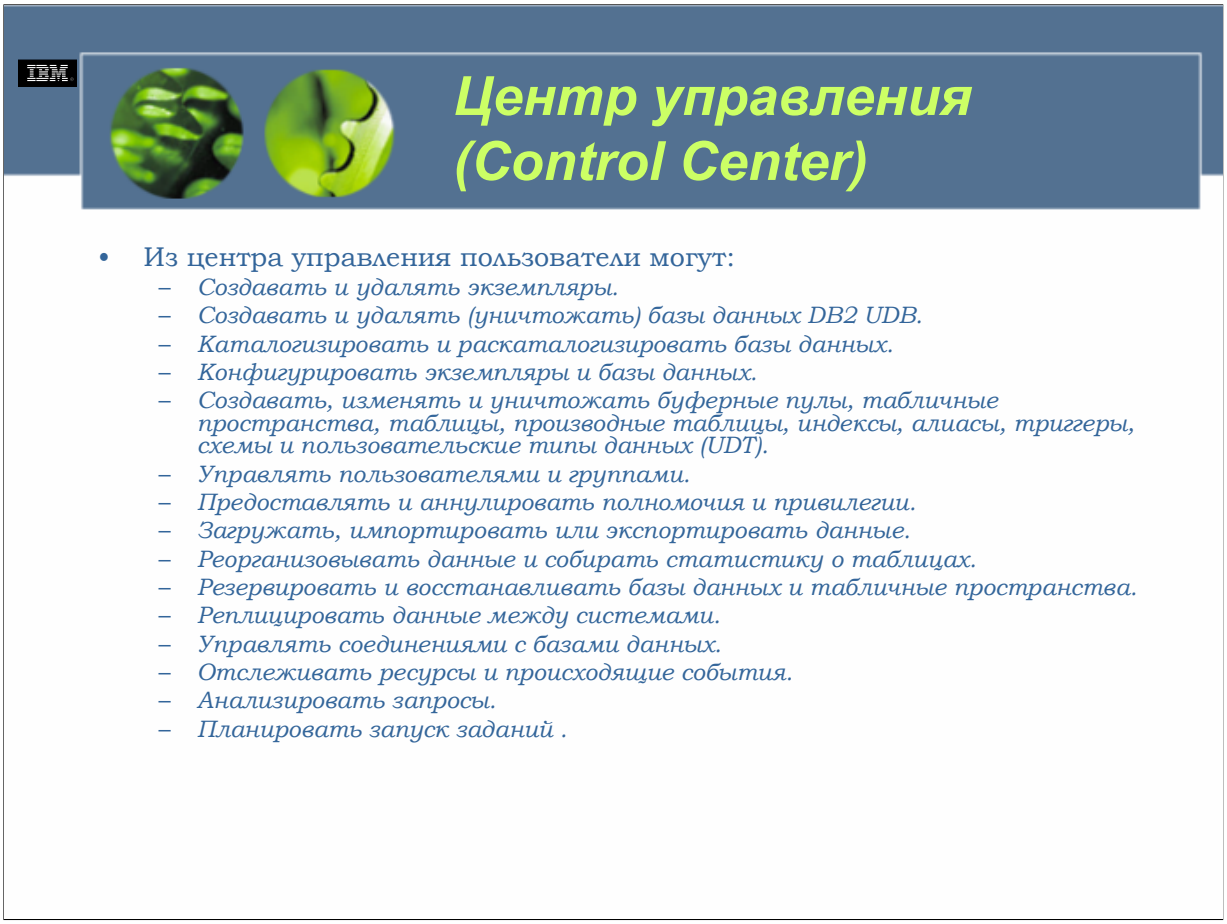
DB2 OLAP Server дает вам возможность создавать среду оперативной аналитической обработки (OLAP), используя универсальную базу данных DB2. Этот продукт основан на технологии OLAP, которая была разработана Hyperion Solutions Corporation (которая выступает на рынке как Hyperion Essbase) и которая может использоваться для создания широкого разнообразия приложений для многомерного планирования, анализа и создания отчетов, взаимодействующих с хранилищами данных. DB2 OLAP Server содержит свыше 100 встроенных функций, включая финансовые, статистические и математические функции, и имеет возможность хранения многомерных баз данных в виде набора таблиц реляционных баз данных. И поскольку DB2 OLAP Server построен на технологии Hyperion Essbase, он предоставляет те же функциональные возможности, что и Hyperion Essbase, и поддерживает широко распространенный интерфейс прикладного программирования (API) Hyperion Essbase, который доступен через широкое разнообразие интерфейсных аналитических инструментов и промышленных приложений, а также для стандартных инструментов запросов SQL. У вас есть также возможность использовать для создания приложения DB2 OLAP и связанных с ним баз данных Essbase Application Manager и команды Essbase.

DB2 OLAP Server является добавочным продуктом, который должен приобретаться отдельно; однако большинство редакций DB2 Universal Database поставляются с упрощенной версией DB2 OLAP Server, известной как DB2 OLAP Starter Kit. (DB2 OLAP Starter Kit предоставляет те же самые функциональные возможности, что и DB2 OLAP Server, но лишь для трех параллельно работающих пользователей).



WCF03.0

За исключением DB2 Everyplace, каждая редакция DB2 Universal Database, наряду с DB2 Administration Client, укомплектовывается набором разнообразных инструментов, предназначенных для содействия в администрировании и управлении экземплярами DB2 UDB, базами данных и их объектами. Большинство этих инструментов имеют графический интерфейс (GUI); однако многие задачи, которые можно выполнить с помощью предоставленных графических инструментов, можно осуществить также с помощью эквивалентных команд DB2 UDB из командной строки операционной системы. В следующих разделах описываются чаще всего использующиеся из доступных инструментов.



WCF03.0

Из всех доступных графических инструментов DB2 UDB центр управления (Control Center) является самым важным и многосторонним. Центр управления дает ясный, точный вид на всю систему и служит в качестве центральной точки для управления системой и осуществления обычных административных задач. Из центра управления пользователи могут:

•Создавать и удалять экземпляры.

•Создавать и удалять (уничтожать) базы данных DB2 UDB.

•Каталогизировать и раскаталогизировать базы данных.

•Конфигурировать экземпляры и базы данных.

•Создавать, изменять и уничтожать буферные пулы, табличные пространства, таблицы, производные таблицы, индексы, алиасы, триггеры, схемы и пользовательские типы данных (UDT).

•Управлять пользователями и группами.

•Предоставлять и аннулировать полномочия и привилегии.

•Загружать, импортировать или экспортировать данные.

•Реорганизовывать данные и собирать статистику о таблицах.

•Резервировать и восстанавливать базы данных и табличные пространства.

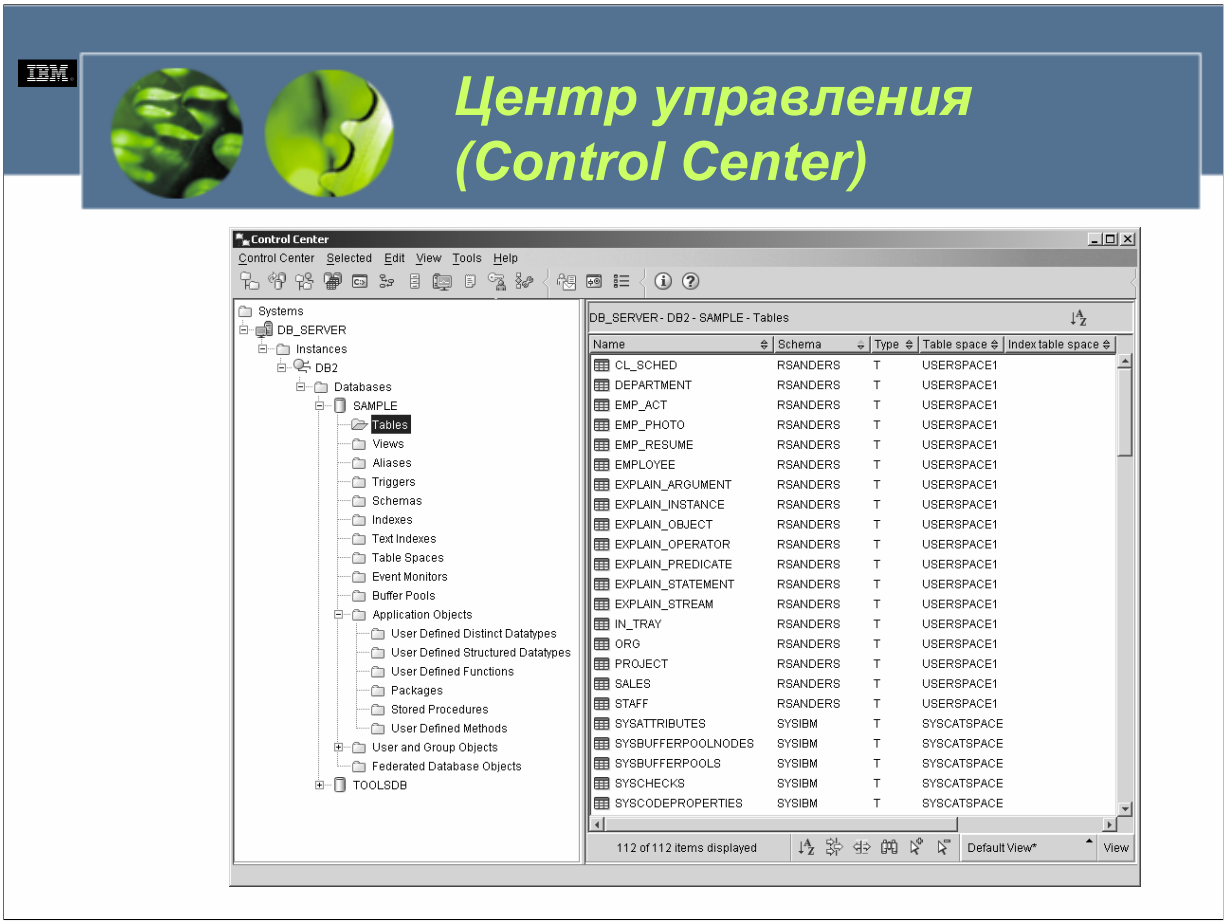
•Реплицировать данные между системами.

•Управлять соединениями с базами данных.

•Отслеживать ресурсы и происходящие события.

•Анализировать запросы.

•Планировать запуск заданий .



WCF03.0

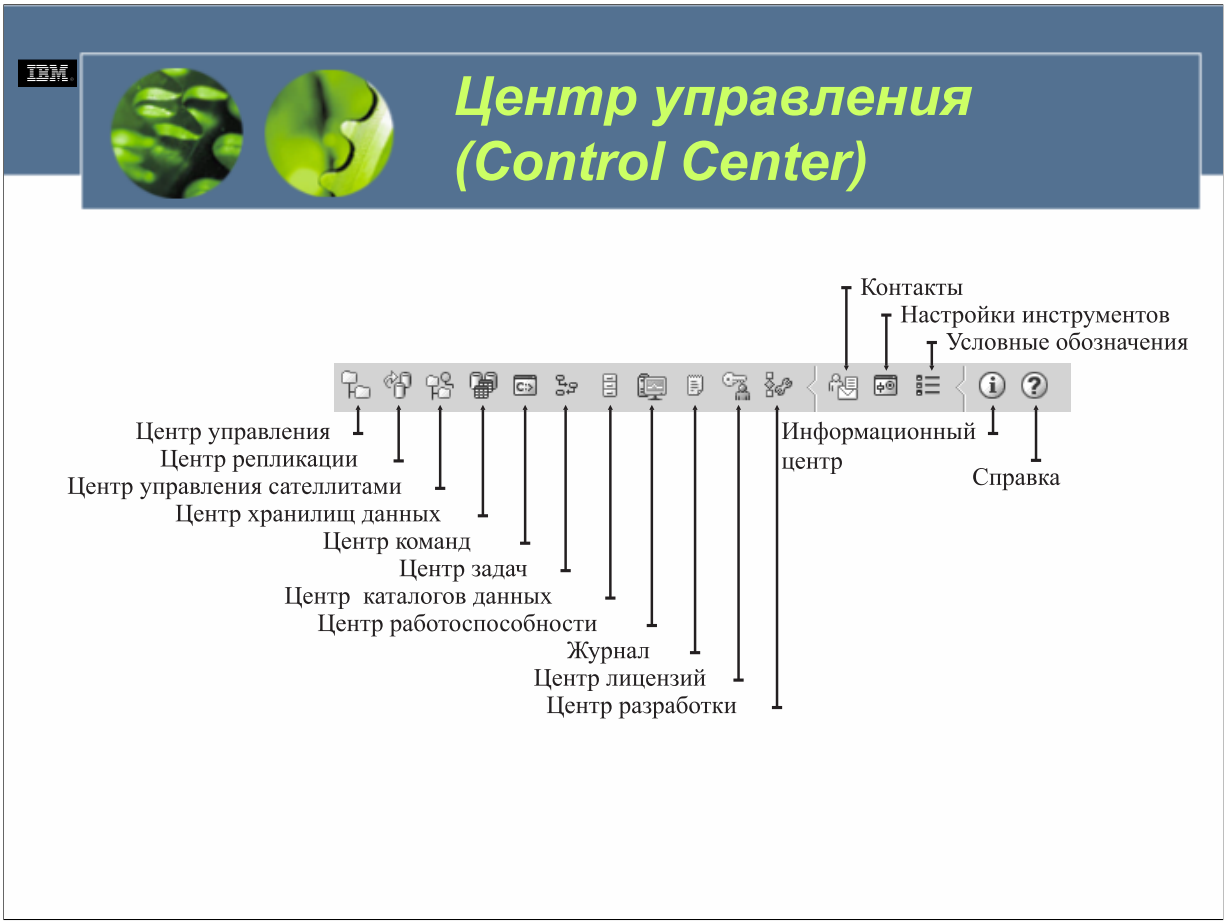
Центр управления состоит из следующих элементов:

Строки меню, которое позволяет пользователям выполнять все доступные функции центра управления.

Панели инструментов, которую можно использовать для запуска других доступных графических инструментов DB2 UDB. На рис. 2-8 показаны инструменты, которые могут быть запущены непосредственно из панели инструментов центра управления. Важно отметить, что каждый инструмент, который можно вызвать из панели инструментов центра управления, можно вызвать также из меню.

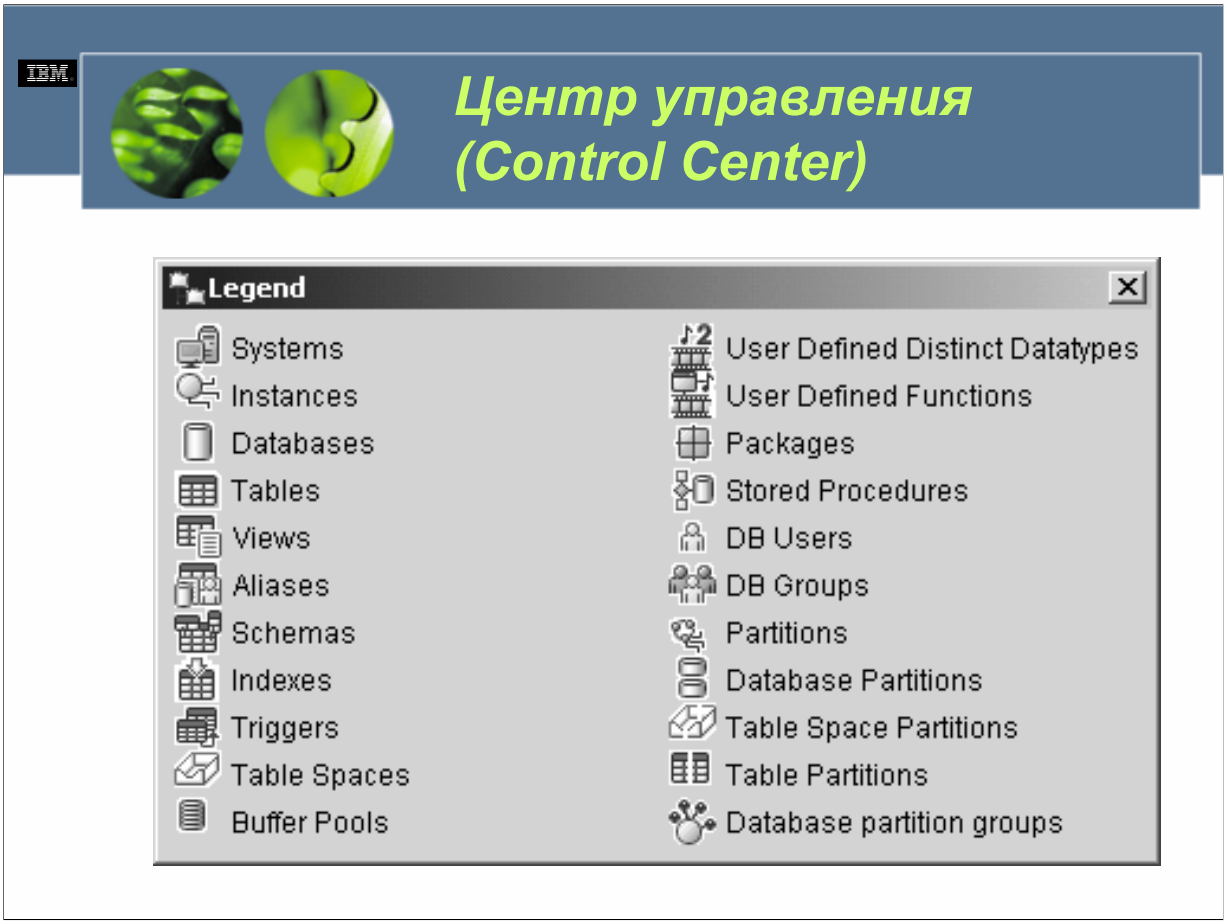
Панель объектов (расположенная в левой части центра управления), которая содержит иерархическое представление каждого типа объекта, который может управляться из центра управления.

Панель содержания (расположенная в правой части центра управления), которая содержит список существующих объектов, соответствующих выбранному в панели объектов типу объекта. (Например, если в панели объектов был бы выбран тип объекта Таблицы, в панели содержания были бы перечислены все доступные таблицы).



WCF03.0

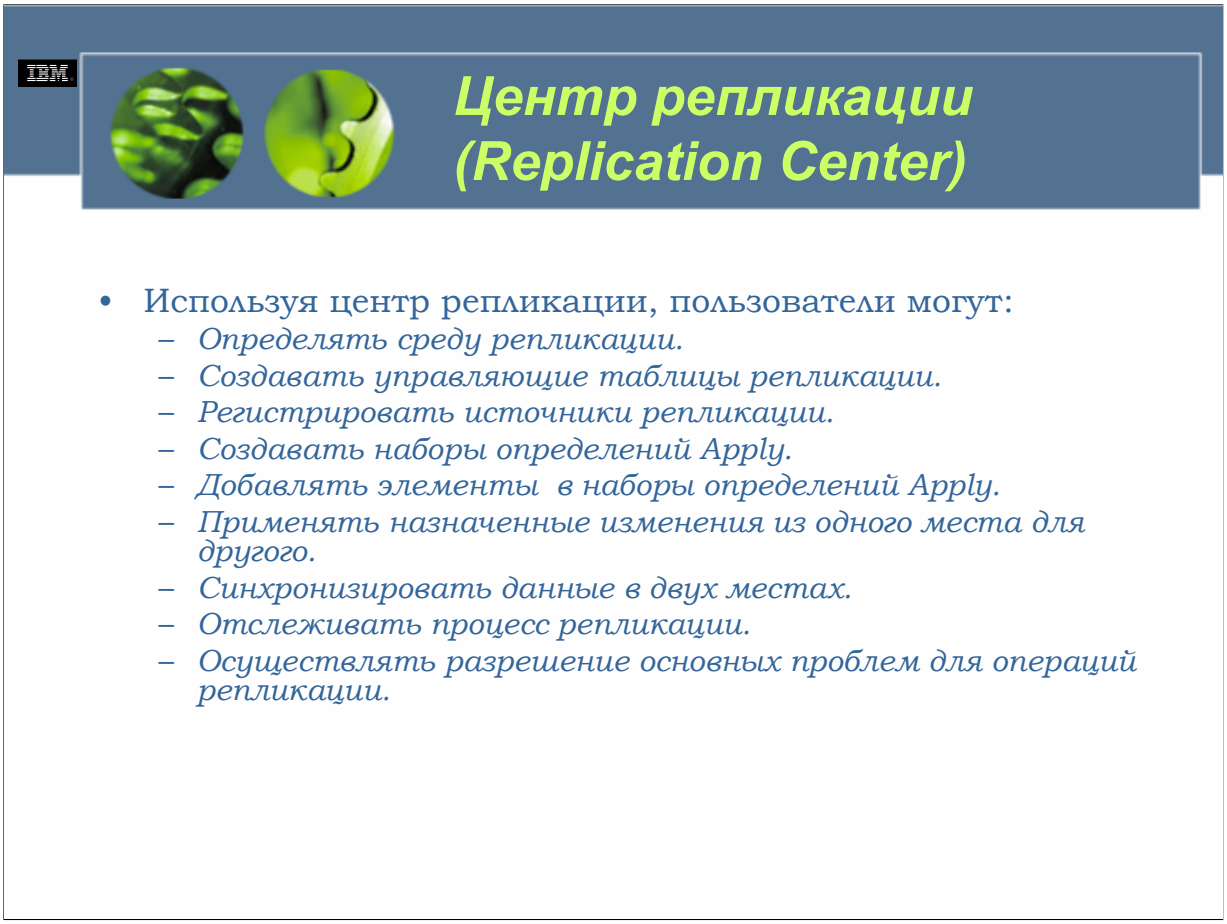
Из центра управления пользователи могут также открывать другие центры управлений, запускать другие инструменты, выполнять задачи управления хранением данных и работать с командами DB2.



WCF03.0

Каждому перечисленному в панели содержания объекту предшествует значок, предназначенный для идентификации типа описываемого в списке объекта. Используется широкое разнообразие значков, а список всех доступных значков вместе с соответствующими типами объектов можно увидеть в диалоговом окне условных обозначений (Legend), который можно вызвать из меню центра управления.

Пользователи могут выполнить определенные задачи с любым объектом, выбрав его из списка и щелкнув правой клавишей мыши; будет отображено всплывающее окно с перечислением всех возможных действий для данного конкретного объекта, и пользователь просто выбирает нужное действие из меню.



WCF03.0

Центр репликации (Replication Center) является интерактивным графическим приложением, которое позволяет пользователям администрировать реплицирование данных между базой данных DB2 UDB и любой другой реляционной базой данных, будь то база данных DB2 или другая. Используя центр репликации, пользователи могут:

•Определять среду репликации.

•Создавать управляющие таблицы репликации.

•Регистрировать источники репликации.

•Создавать наборы определений Apply.

•Добавлять элементы в наборы определений Apply.

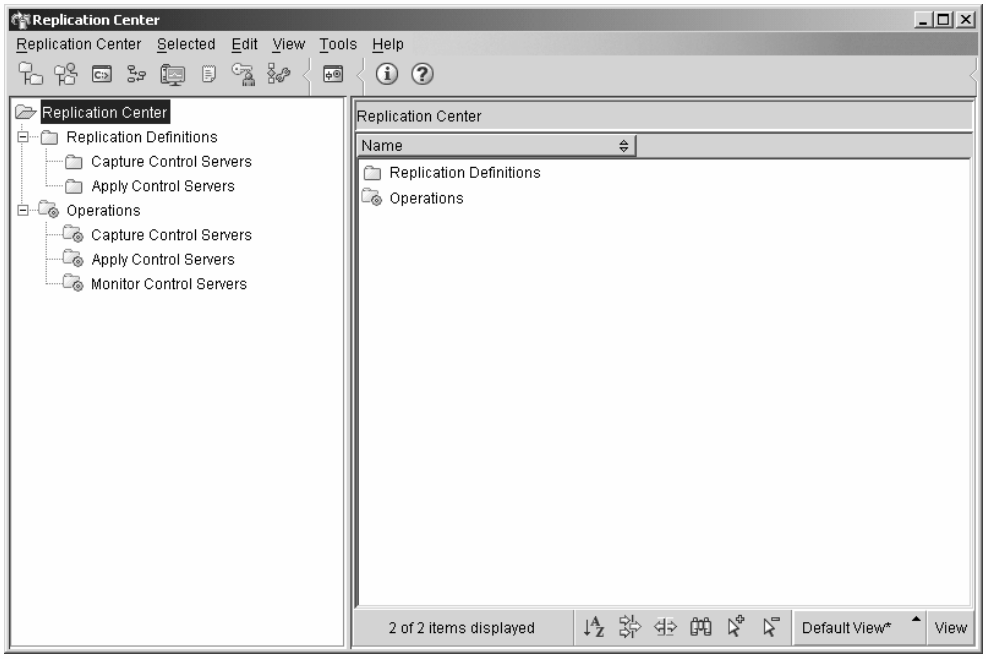
•Применять назначенные изменения из одного места для другого.

•Синхронизировать данные в двух местах.

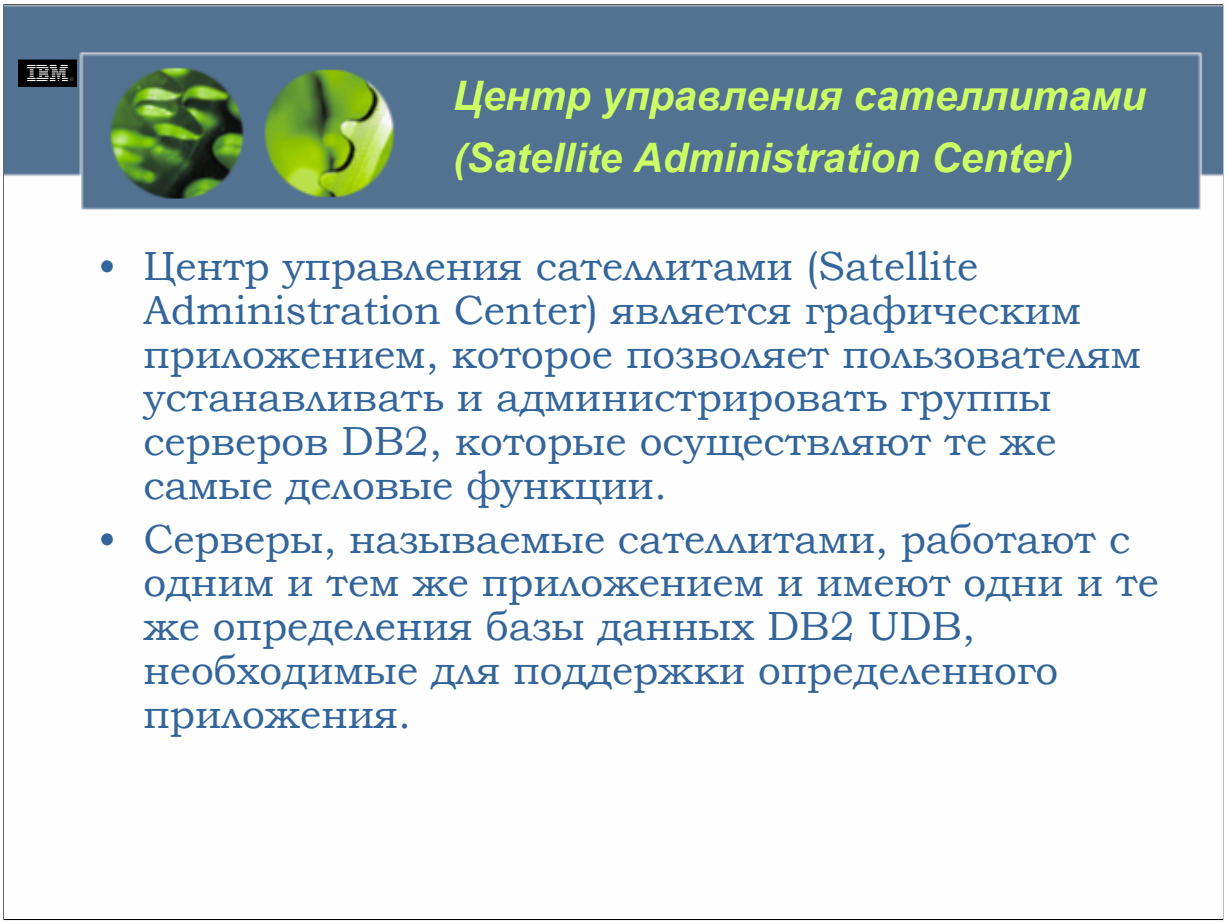
•Отслеживать процесс репликации.

•Осуществлять разрешение основных проблем для операций репликации.

Центр репликации (Replication Center)



WCF03.0



WCF03.0

Центр управления сателлитами (Satellite Administration Center) является графическим приложением, которое позволяет пользователям устанавливать и администрировать группы серверов DB2, которые осуществляют те же самые деловые функции. Эти серверы, называемые сателлитами, работают с одним и тем же приложением и имеют одни и те же определения базы данных DB2 UDB, необходимые для поддержки определенного приложения. С помощью центра управления сателлитами пользователи создают группы, а затем определяют сателлиты в качестве членов этих групп. Затем этой группой сателлитов можно управлять как единым целым, в отличие от необходимости отдельного администрирования каждым сателлитом. Если позже запрашиваются дополнительные серверы DB2, осуществляющие те же самые деловые функции, они просто добавляются в группу в качестве дополнительных сателлитов.

Информация о среде сателлитов хранится в центральной базе данных, которая называется управляющей базой данных сателлитов. В данной базе данных, помимо прочего, записывается, какие сателлиты находятся в среде, группа, к которой принадлежат сателлиты, с какой версией бизнес-прилложения конечного пользователя в данный момент сателлит работает. Эта база данных находится на сервере DB2 UDB, который называется управляющим сервером DB2, он должен быть каталогизирован и сделан доступным центру управления до того, как центр управления сателлитами сможет с ним взаимодействовать.

Группы сателлитов администрируются путем создания пакетных сценариев для настройки и поддержания определения баз данных, поддерживающих бизнес-приложение, на каждом сателлите в группе. Затем каждый сателлит регулярно подключается к управляющему серверу сателлитов и загружает все сценарии, которые к нему относятся. Сателлит выполняет эти сценарии локально и загружает результаты обратно в управляющую базу данных сателлитов. Этот процесс загрузки пакетных сценариев, их выполнения и сообщения о результатах исполнения пакета обратно в управляющую базу данных сателлитов называется синхронизацией. Сателлит синхронизируется для поддержания своей согласованности с другими сателлитами, принадлежащими к его группе.

WCF03.0

Центр хранилищ данных (Data Warehouse Center) представляет собой набор графических инструментов, которые позволяют пользователю строить, получать доступ и управлять хранилищами данных DB2 UDB. С помощью центра хранилищ данных пользователи могут автоматизировать процесс извлечения, преобразования и загрузки, который должен быть выполнен для заполнения хранилища данных, а также планировать, поддерживать и отслеживать каждую фазу этого процесса. Центр хранилищ данных может использоваться также для:

•Настройки хранилища данных.

•Создания схемы типа «звезда».

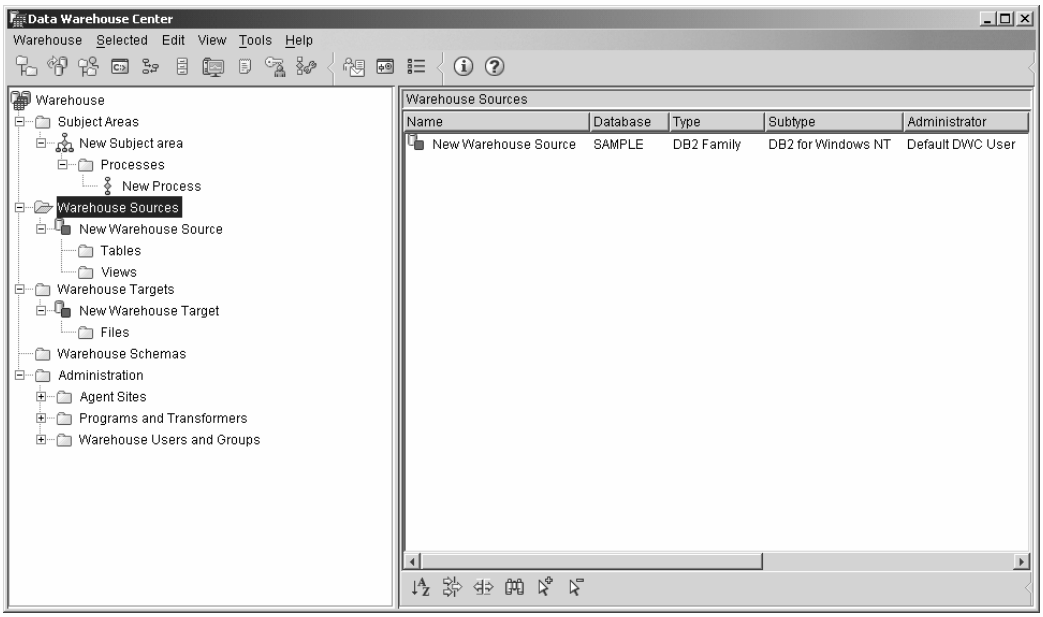
•Установки источников данных DB2 и не-DB2.

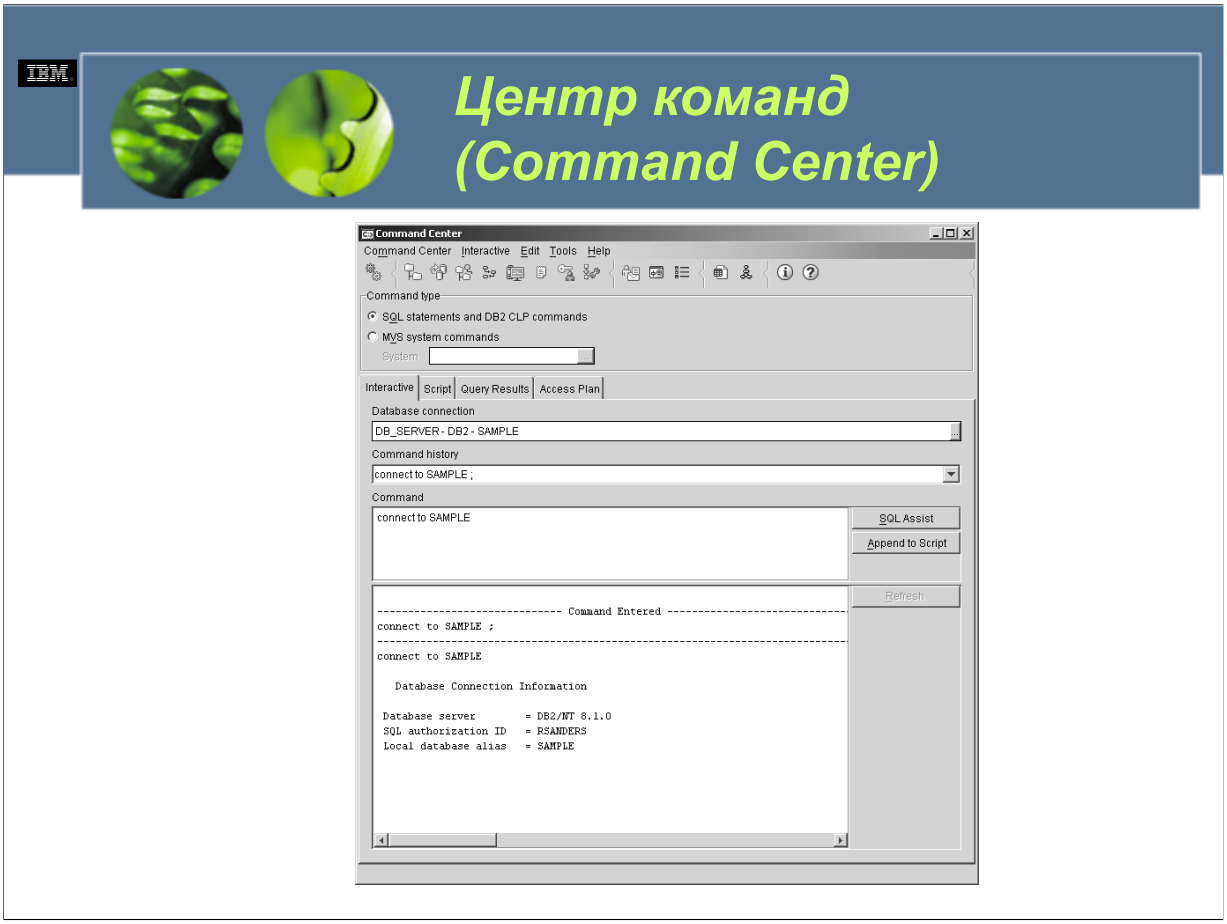
•Настройки доступа к хранилищу данных.

•Вычисления статистики.

•Управления метаданными и управляющей базой данных.

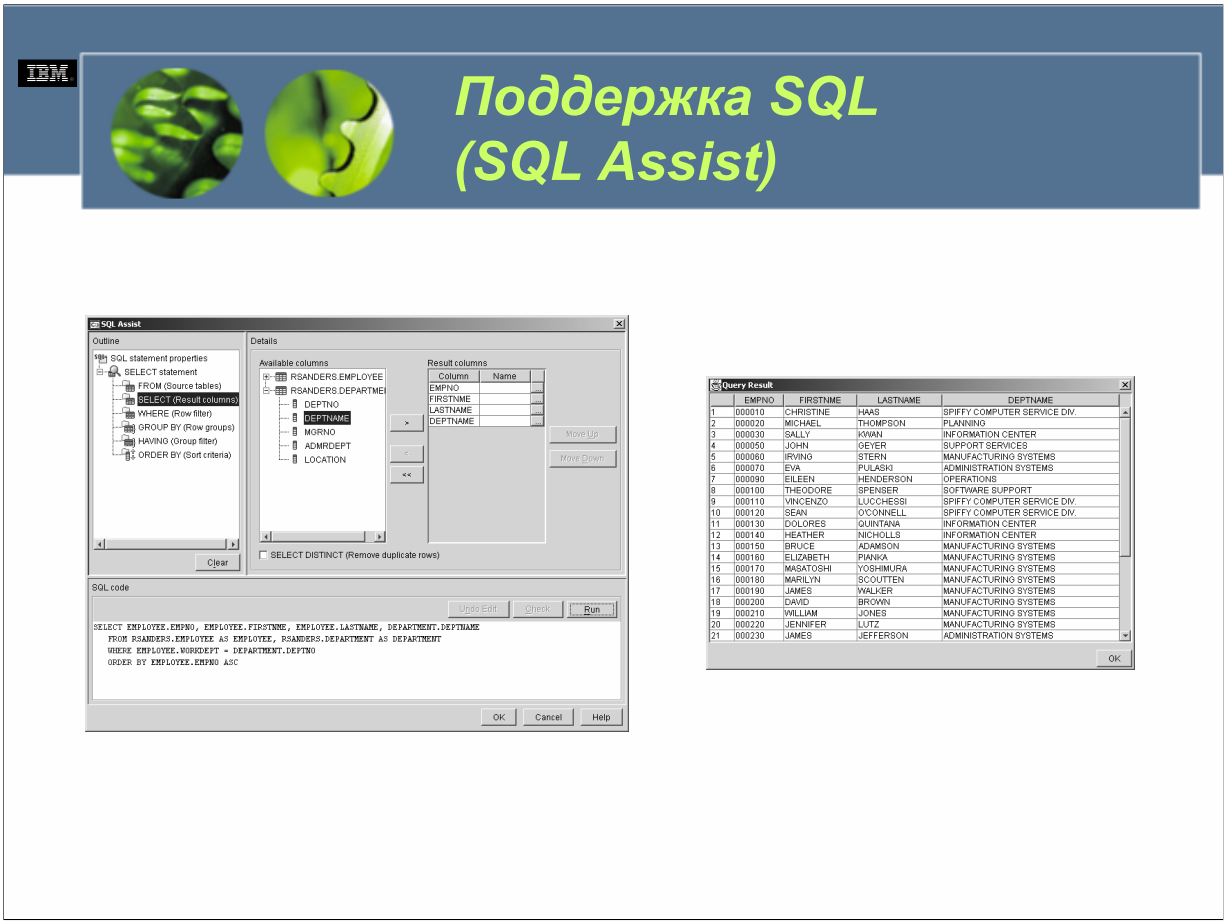
Центр хранилищ данных (Data Warehouse Center)





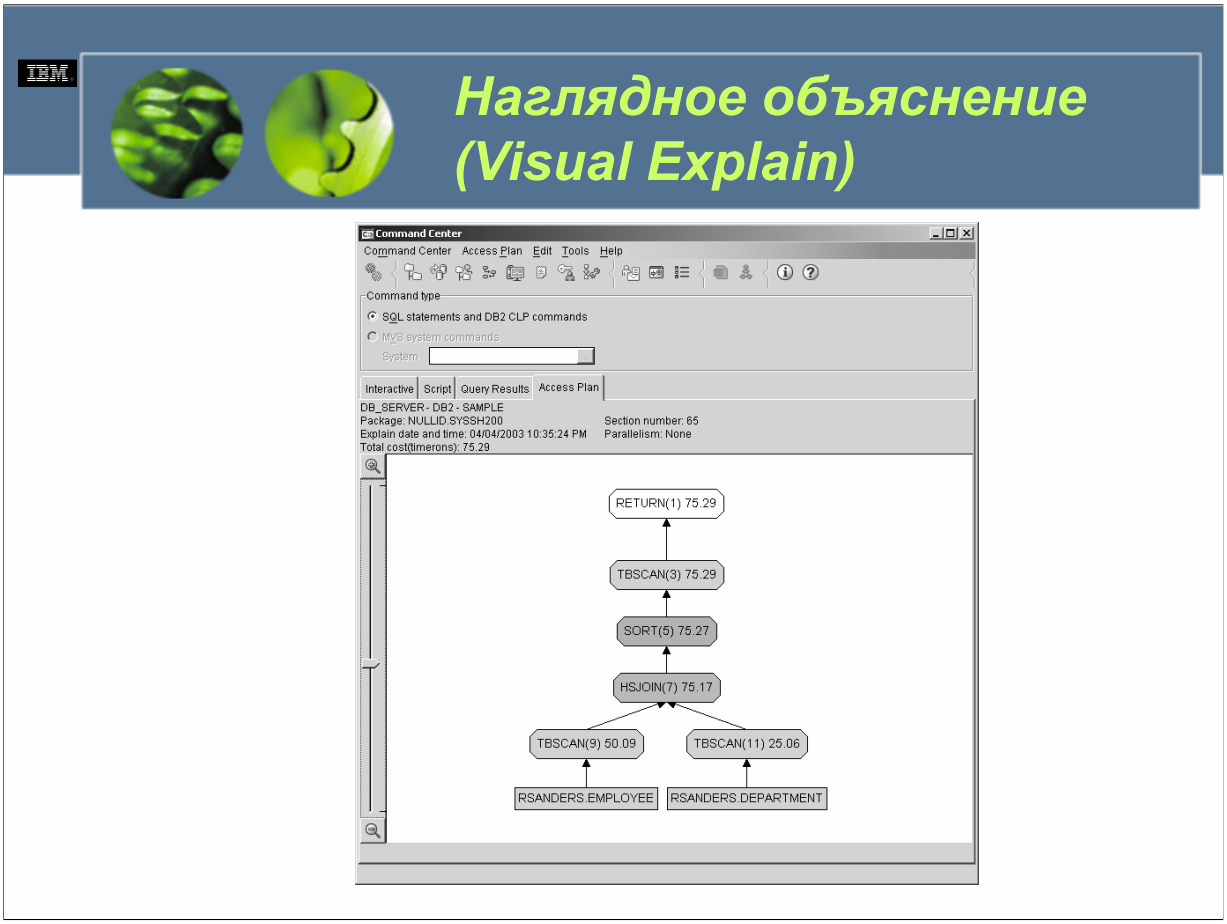
WCF03.0

Центр команд составлен из четырех различных отдельных страниц: страницы *Интерактивно* (*Interactive*), страницы *Сценарии* (*Script*), страницы *Результаты запроса* (*Query Results*) и страницы *План доступа* (*Access Plan*). На странице интерактивной работы пользователи могут вводить и выполнять операторы SQL или команды DB2. На странице сценариев пользователи могут выполнять команды последовательно, создавать и сохранять сценарии, запускать существующие сценарии или планировать задания. На странице результатов запросов пользователи могут видеть результаты любого выполненного запроса. На странице плана доступа пользователи могут видеть план доступа для любого объяснимого оператора, который был указан на странице интерактивной работы или на странице сценариев. (Если на странице сценариев указано более одного оператора SQL, план доступа будет создан лишь для первого оператора).



WCF03.0

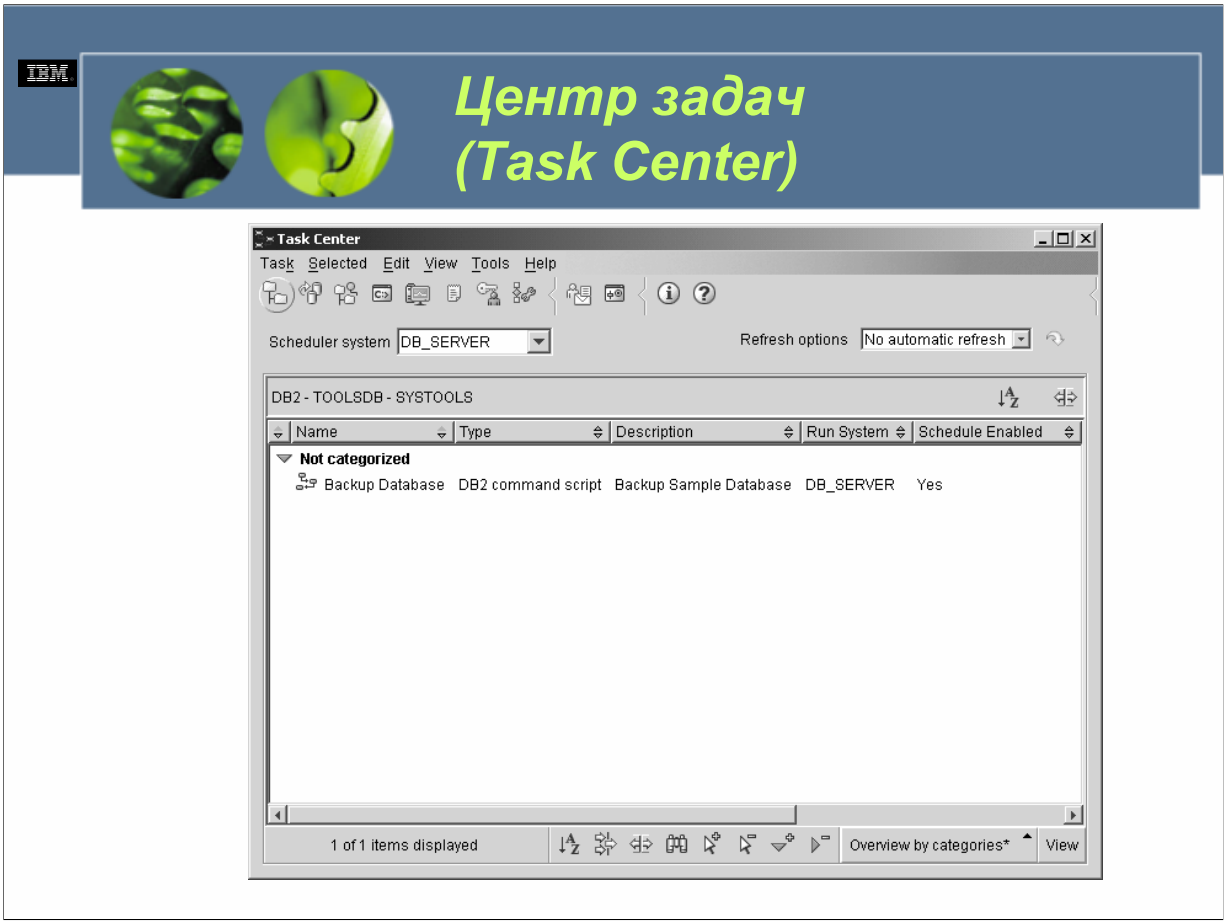
Поддержка SQL является интерактивным графическим приложением, которое позволяет пользователям визуально конструировать сложные операторы SQL SELECT, INSERT, UPDATE и DELETE и проверять результаты их выполнения. Поддержка SQL вызывается непосредственно из центра команд, а после конструирования оператора SQL в поддержке SQL он может быть записан обратно в центр команд, где его можно выполнить немедленно или сохранить в файле сценария, где его можно выполнить позже, используя центр задач. На рис. 2-13 показано, как инструмент поддержка SQL выглядит в Windows 2000 server после его использования для построения сложного запроса. На рис. 2-14 показано, как были бы отображены результаты этого запроса, если запрос был бы выполнен из поддержки SQL (путем выбора кнопки «Run», расположенной в правом нижнем углу экрана).



WCF03.0

Наглядное объяснение(Visual Explain) является графическим приложением, позволяющим пользователям просматривать подробности плана доступа (включая статистику в системных каталогах), выбранного оптимизатором DB2 для данного оператора SQL, без фактического исполнения оператора. С помощью наглядного объяснения каждая использованная таблица, производная таблица и индекс вместе с произведенными для каждого из них операциями представляется в виде узлов на диаграмме, а фактический поток данных представляется в виде связей между узлами. Эта информация позволяет пользователям быстро просматривать статистику, использованную во время оптимизации определенного запроса, определять, улучшил бы индекс доступ к таблице или нет, получить информацию о стоимости, необходимой для осуществления определенной операции, и понять, как были объединены таблицы. Вооруженные этой информацией, администраторы могут производить изменения в проекте базы данных, а разработчики приложений могут точно отрегулировать операторы SQL, чтобы повысить общую производительность.

Аналогично поддержке SQL , наглядное объяснение может вызываться непосредственно из центра команд. Однако в отличие от подддержки SQL , наглядное объяснение может вызываться также из меню управляющего центра. При вызове из центра команд наглядное объяснение может генерировать информацию плана доступа для сложных запросов, которые были созданы с помощью поддержки SQL.

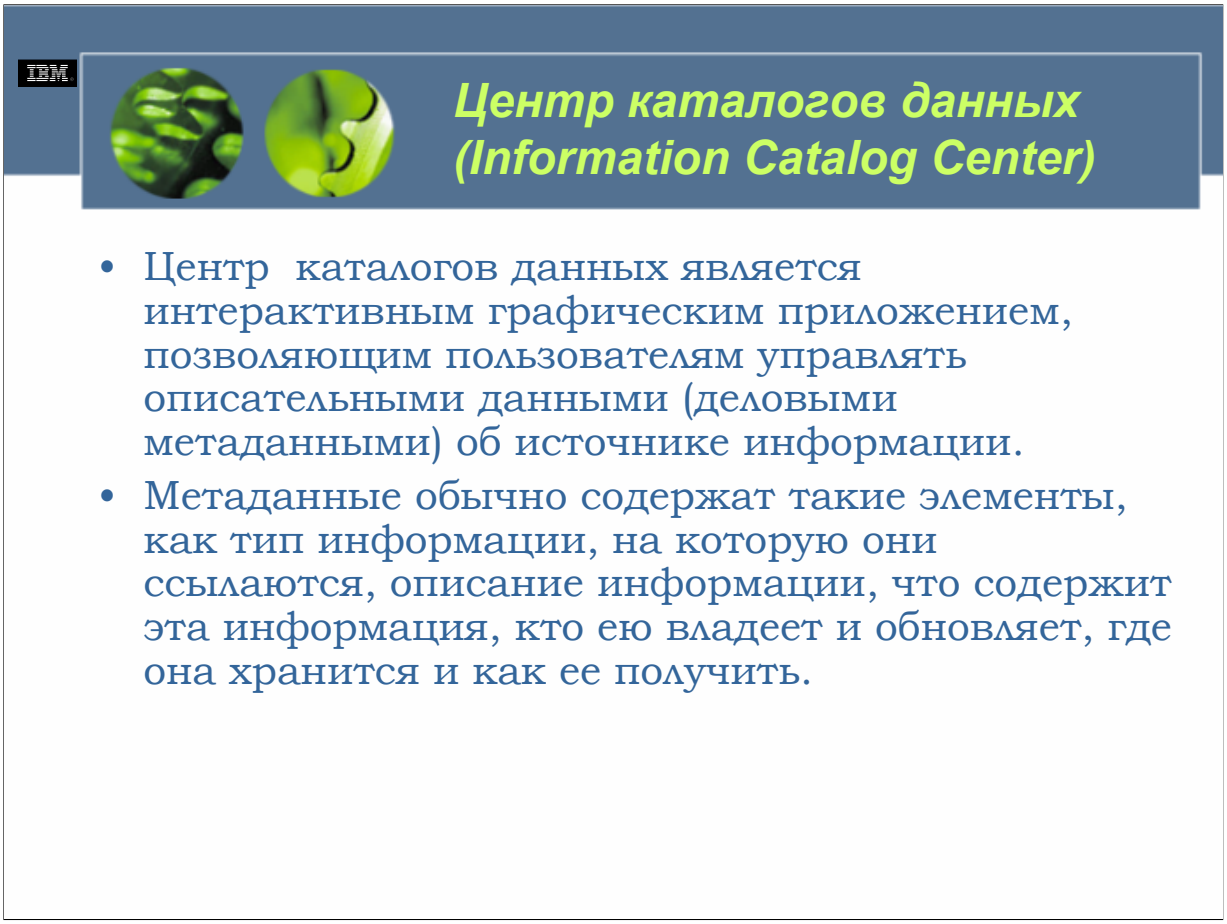


WCF03.0

Центр задач (Task Center) является интерактивным графическим приложением, позволяющим пользователям планировать задачи, запускать задачи и посылать уведомления о завершенных задачах другим пользователям. Задача является сценарием вместе со всеми связанными с ним условиями успеха, расписаниями и уведомлениями. Пользователи могут создавать задачи в центре задач, генерировать задачу путем сохранения результатов из диалогового окна или мастера DB2, создавать сценарий в другом инструменте и сохранить его для центра задач или импортировать существующий сценарий. Такие сценарии могут содержать команды DB2, операторы SQL или команды операционной системы.

Центр задач использует наборы кодов успешного завершения (коды возврата или диапазон кодов возврата, получение которых означает, что задача была выполнена успешно) для оценки успешности или неудачи выполнения им любой задачи. Коды возврата, которые выходят за рамки указанного диапазона, рассматриваются, как неудача. Более того, центр задач оценивает код возврата SQLCA каждого оператора SQL, выполненного в сценарии DB2, и если любой из операторов завершается неудачно, неудачей завершается весь сценарий.

Помимо оценки успеха или неудачи определенной задачи, центр задач может выполнить одно или более действий в случае успешного выполнения определенной задачи, и выполнить другие действия, если та же самая задача завершится неудачно. Центр задач можно также сконфигурировать для осуществления одного или более действий каждый раз, когда завершается запланированная задача, независимо от того, завершилась она успешно или нет.

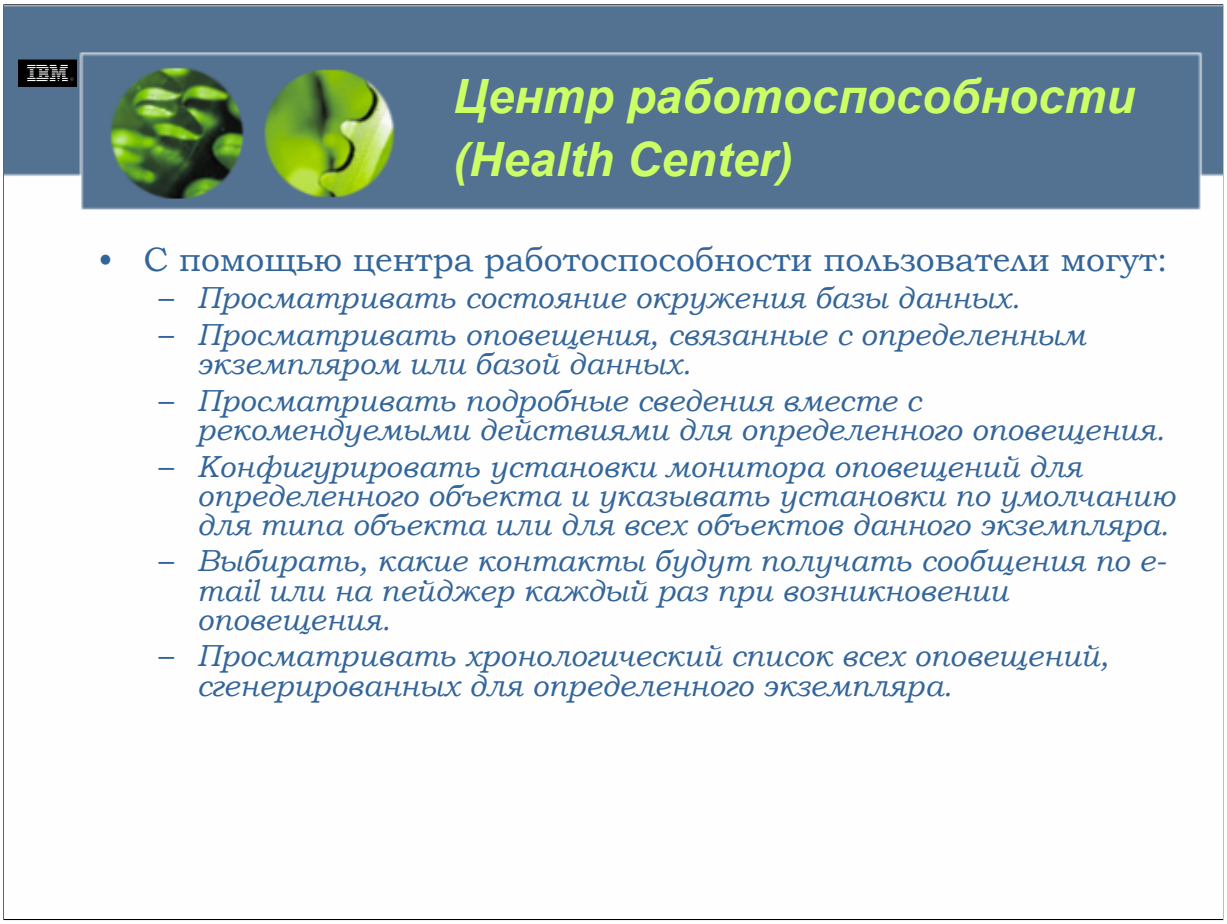


WCF03.0

Центр каталогов данных (Information Catalog Center) является интерактивным графическим приложением, позволяющим пользователям управлять описательными данными (деловыми метаданными) об источнике информации. Эти метаданные обычно содержат такие элементы, как тип информации, на которую они ссылаются, описание информации, что содержит эта информация, кто ею владеет и обновляет, где она хранится и как ее получить. В сущности центр каталогов данных делает для организации то же самое, что каталог электронных карт делает для библиотеки, и даже еще больше.

С помощью центра каталогов данных пользователи могут искать определенные объекты, хранящиеся в информационном каталоге, просматривать любые взаимоотношения, в которых объект принимает участие, просматривать происхождение объекта и создавать для объекта комментарии. Пользователи с соответствующими полномочиями могут также создавать новые объекты для определенного информационного каталога.

Центр каталогов помогает администраторам организовывать объекты метаданных, требуя, чтобы каждый объект основывался на типе объекта, и позволяя администраторам определять типы отношений и дополнительные типы объектов. Более того, центр каталогов обеспечивает безопасность на уровне объектов, поэтому для каждого объекта устанавливаются привилегии, допуская больший контроль над деловой информацией.



WCF03.0

Центр работоспособности (Health Center) является интерактивным графическим приложением, позволяющим пользователям отслеживать состояние окружения баз данных. Каждый раз, когда DB2 UDB активна, фоновый процесс непрерывно отслеживает набор индикаторов работоспособности для проблемных областей. Если текущее значение индикатора работоспособности выходит за рамки приемлемого рабочего диапазона, определяемого его порогами предупреждения и аварийного сигнала, монитор работоспособности генерирует оповещение. (DB2 UDB поставляется с набором предопределенных пороговых значений, которые можно настроить в соответствии с вашими потребностями).

С помощью центра работоспособности пользователи могут:

•Просматривать состояние окружения базы данных. Рядом с каждым объектом, перечисленным в навигационном дереве центра работоспособности, отображается значок, указывающий на уровень оповещения для определенного объекта (или для любых объектов, содержащихся в данном объекте). Если значком является зеленый ромбик, соответствующий объект в настоящее время не имеет никаких оповещений.

•Просматривать оповещения, связанные с определенным экземпляром или базой данных. Когда выбран объект в навигационном дереве центра работоспособности. оповещения для этого объекта отображаются на панели, расположенной непосредственно справа от навигационного дерева.

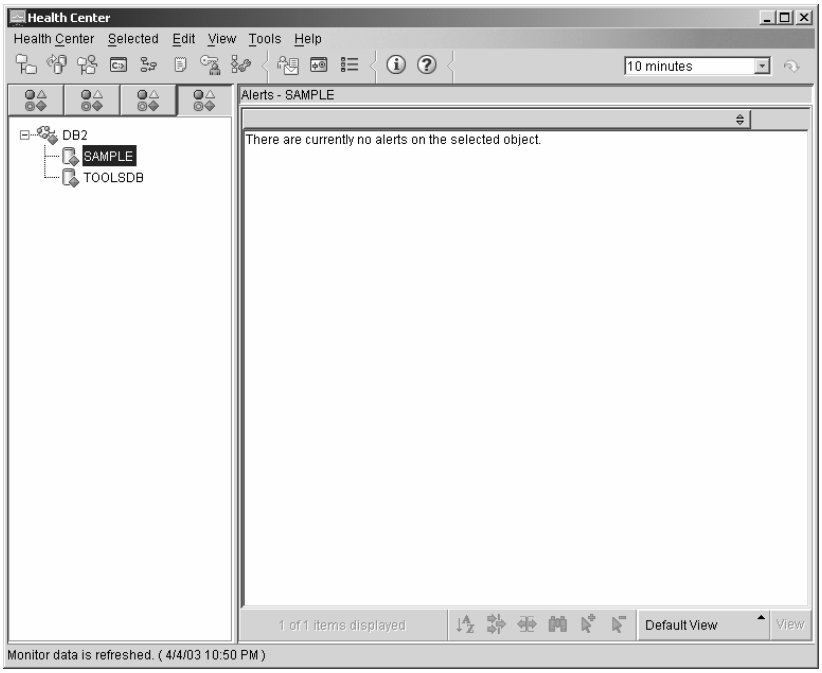
•Просматривать подробные сведения вместе с рекомендуемыми действиями для определенного оповещения. когда выбран элемент определенного оповещения, отображается диалоговое окно, содержащее блокнот. На первой странице этого блокнота содержится подробная информация об этом оповещении. На второй странице рекомендуются действия для исправления ситуации, которая вызвала появление этого оповещения.

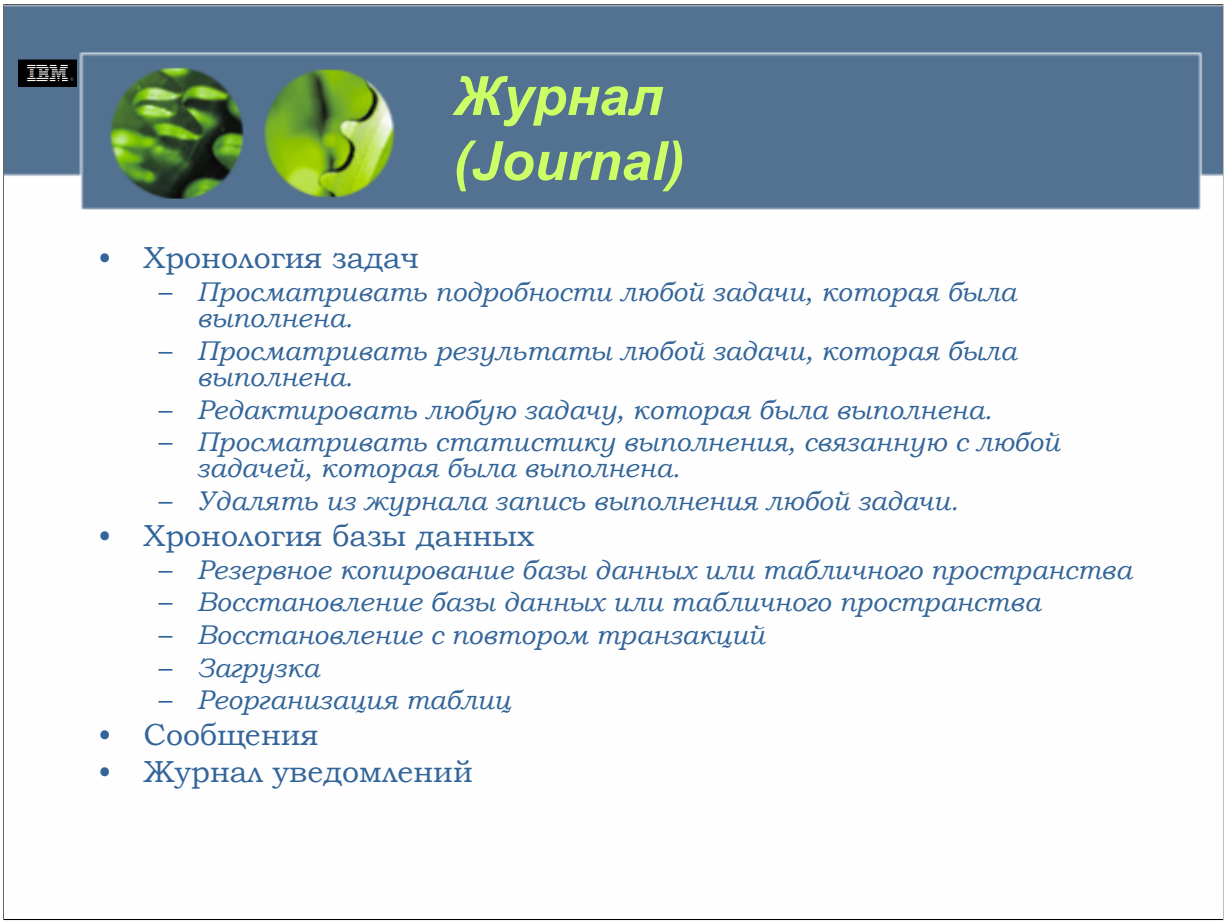
•Конфигурировать установки монитора оповещений для определенного объекта и указывать установки по умолчанию для типа объекта или для всех объектов данного экземпляра.

•Выбирать, какие контакты будут получать сообщения по e-mail или на пейджер каждый раз при возникновении оповещения.

•Просматривать хронологический список всех оповещений, сгенерированных для определенного экземпляра.

Центр работоспособности (Health Center)





WCF03.0

Журнал (Journal) является интерактивным графическим приложением, отслеживающим хронологическую информацию о задачах, действиях и операциях баз данных, действиях центра управления, сообщениях и оповещениях. Чтобы представить эти сведения организованным образом, журнал использует несколько различных представлений. Это:

•Хронология задач

•Хронология базы данных

•Сообщения

•Журнал уведомлений

Представление *хронология задач (Task History)* отображает результаты заданий, которые уже были выполнены. Это представление содержит по одному элементу для каждой отдельной задачи (независимо от того, сколько раз задача выполнялась), и позволяет пользователям:

•Просматривать подробности любой задачи, которая была выполнена.

•Просматривать результаты любой задачи, которая была выполнена.

•Редактировать любую задачу, которая была выполнена.

•Просматривать статистику выполнения, связанную с любой задачей, которая была выполнена.

•Удалять из журнала запись выполнения любой задачи.

Представление *хронология базы данных (Database History)* отображает сведения, хранящиеся в файле хронологии восстановлений базы данных. Файл хронологии восстановления автоматически обновляется каждый раз, когда выполняется любая из следующих операций:

•Резервное копирование базы данных или табличного пространства

•Восстановление базы данных или табличного пространства

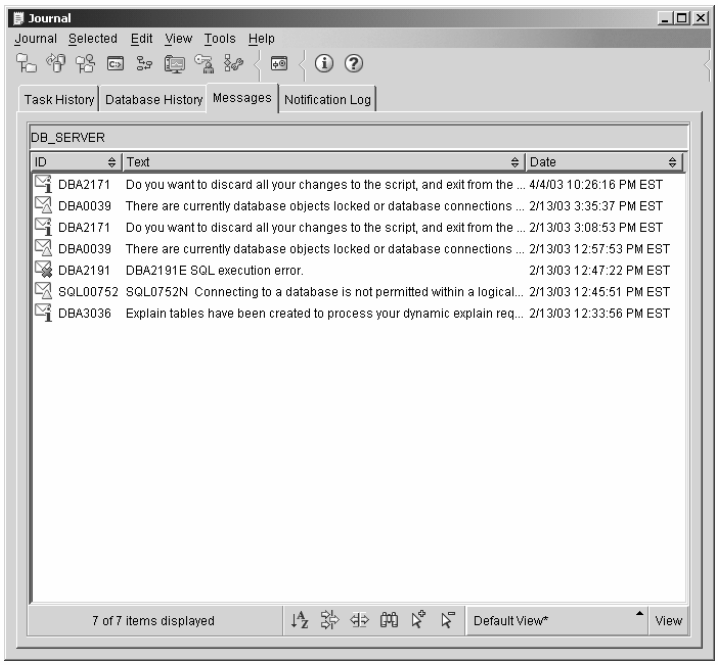
•Восстановление с повтором транзакций

•Загрузка

•Реорганизация таблиц

Представление *сообщения (Messages)* отображает текущую хронологию сообщений, которые были отправлены центром управления или любым другим графическим инструментом, а производная таблица *журнал уведомлений (Notification Log)* отображает информацию из журнала административных уведомлений.

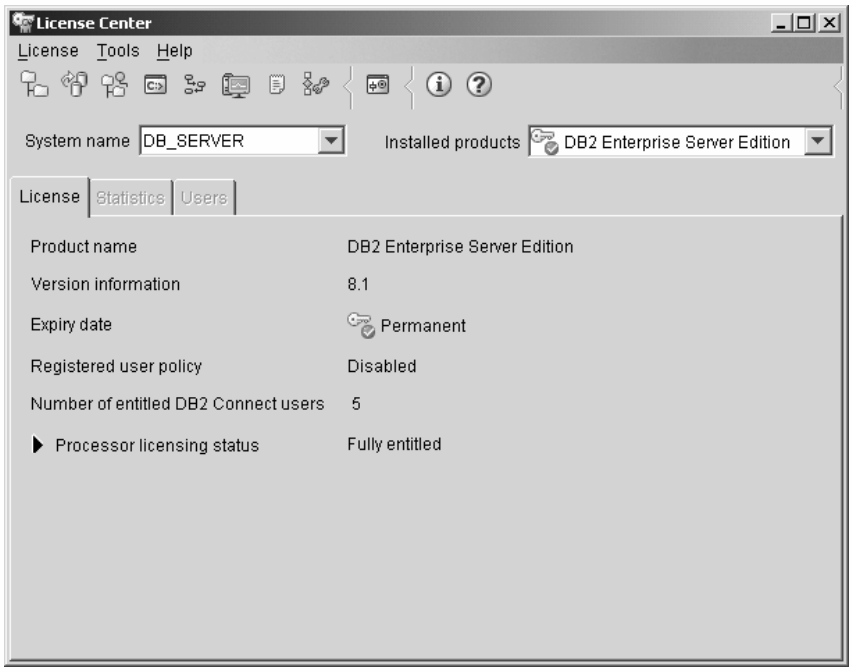
Журнал (Journal)

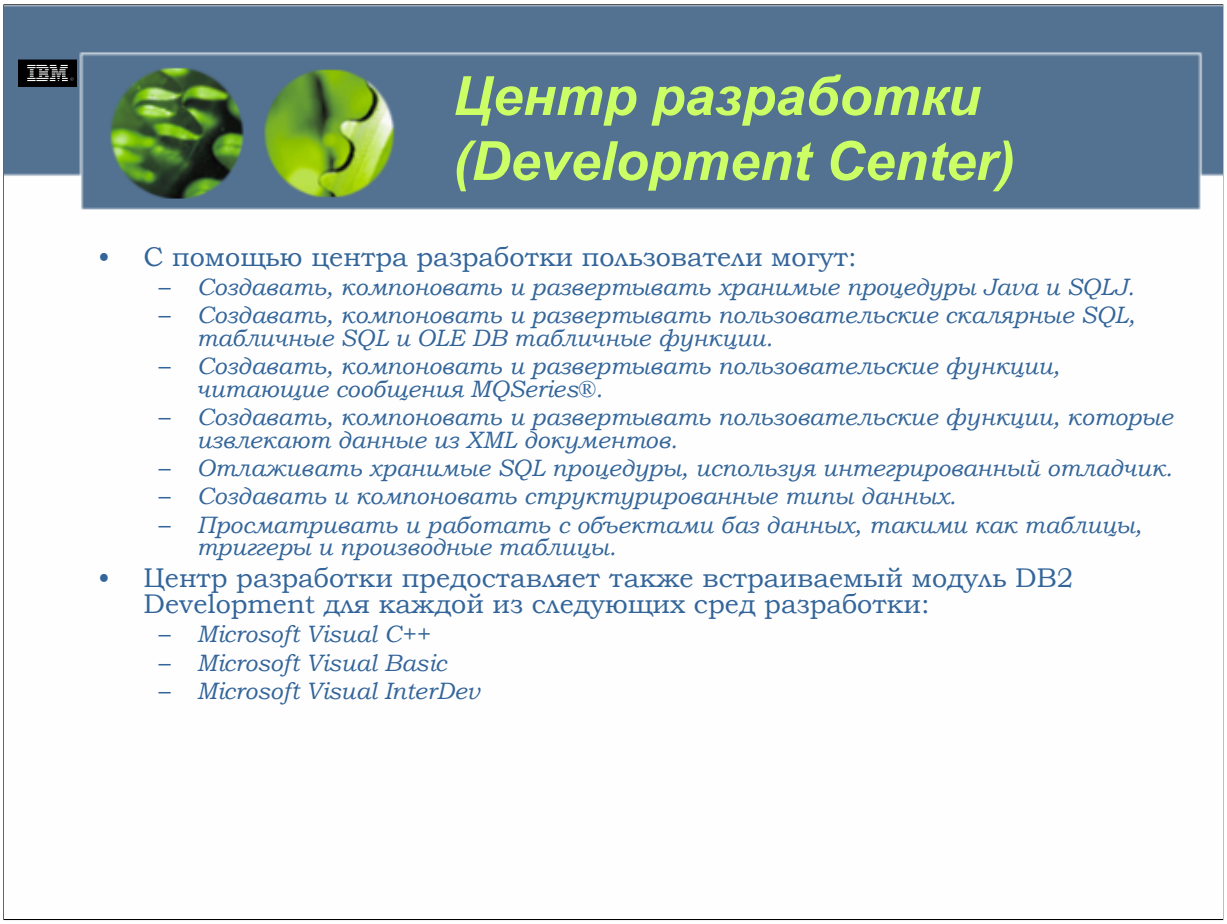


WCF03.0

Центр лицензий (License Center) является интерактивным графическим приложением, позволяющим пользователям просматривать сведения о лицензиях, относящихся к каждому установленному на определенной системе продукту DB2 UDB. Такая информация включает сведения о состоянии процессора, сведения о политике одновременной работы пользователей, сведения о лицензии и статистике пользователей или подробности. Этот инструмент может также использоваться для добавления или удаления лицензий или зарегистрированных пользователей, изменения политик типов лицензий, изменения числа одновременно работающих пользователей, изменения числа лицензированных процессоров, изменения числа лицензий процессоров интернет и конфигурирования определенной системы для соответствующего мониторинга лицензий.

Центр лицензий (License Center)





WCF03.0

Центр разработки (Development Center) является интерактивным графическим приложением, предоставляющим пользователям единую среду разработки, поддерживающую все семейство DB2 UDB. С помощью центра разработки пользователи могут:

•Создавать, компоновать и развертывать хранимые процедуры Java и SQLJ.

•Создавать, компоновать и развертывать пользовательские скалярные SQL, табличные SQL и OLE DB табличные функции.

•Создавать, компоновать и развертывать пользовательские функции, читающие сообщения MQSeries®.

•Создавать, компоновать и развертывать пользовательские функции, которые извлекают данные из XML документов.

•Отлаживать хранимые SQL процедуры, используя интегрированный отладчик.

•Создавать и компоновать структурированные типы данных.

•Просматривать и работать с объектами баз данных, такими как таблицы, триггеры и производные таблицы.

Центр разработки предоставляет также встраиваемый модуль DB2 Development для каждой из следующих сред разработки:

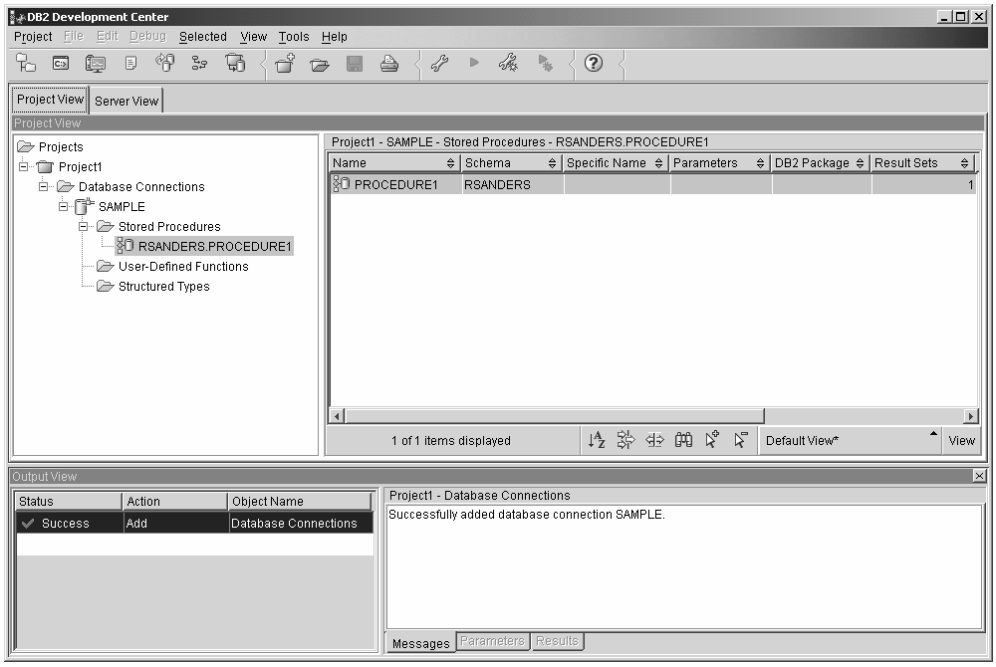
•Microsoft Visual C++

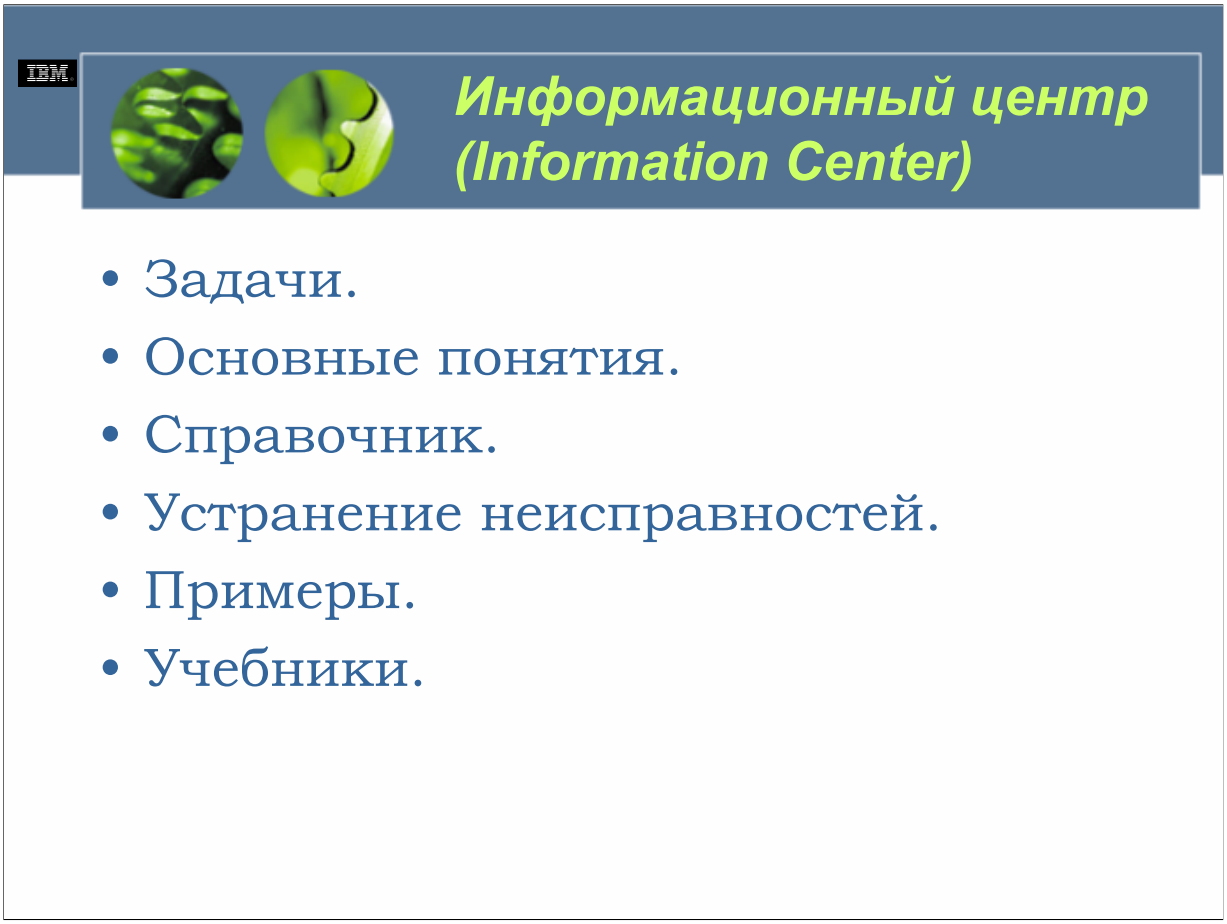
•Microsoft Visual Basic

•Microsoft Visual InterDev

С помощью этих встраиваемых модулей пользователи легко могут получить доступ к центру разработки непосредственно из среды разработки Microsoft. Это упрощает разработку и встраивание пользовательских функций в ходе разработки приложений DB2 UDB.

Центр разработки (Development Center)





WCF03.0

Информационный центр (Information Center) является интерактивным графическим приложением, дающим пользователям возможность получать доступ к электронной копии документации продукта DB2 Universal Database. Информационный центр упрощает быстрое обнаружение нужной информации, поскольку он подразделяет доступную документацию по продукту на различные категории. Список главных тем для каждой категории представлен в отдельных представлениях. Доступные представления включают:

Задачи. Это представление может использоваться для локализации инструкций о том, как осуществлять определенную задачу. Показанные в данном представлении темы включают обычные задачи, которые пользователи могут выполнить с помощью центра управления, вместе с другими главными административными задачами и задачами по разработке приложений.

Основные понятия. Это представление может быть использовано для просмотра того, какие онлайн-руководства по DB2 UDB доступны, и для поиска информации в определенной книге.

Справочник. Это представление может использоваться для обнаружения справочной информации. Это представление предназначено для предоставления быстрого доступа к информации, содержащейся в справочных руководствах по DB2 UDB (*Command Reference, SQL Reference, Call Level Interface Guide and Reference* и *Administrative API Reference*).

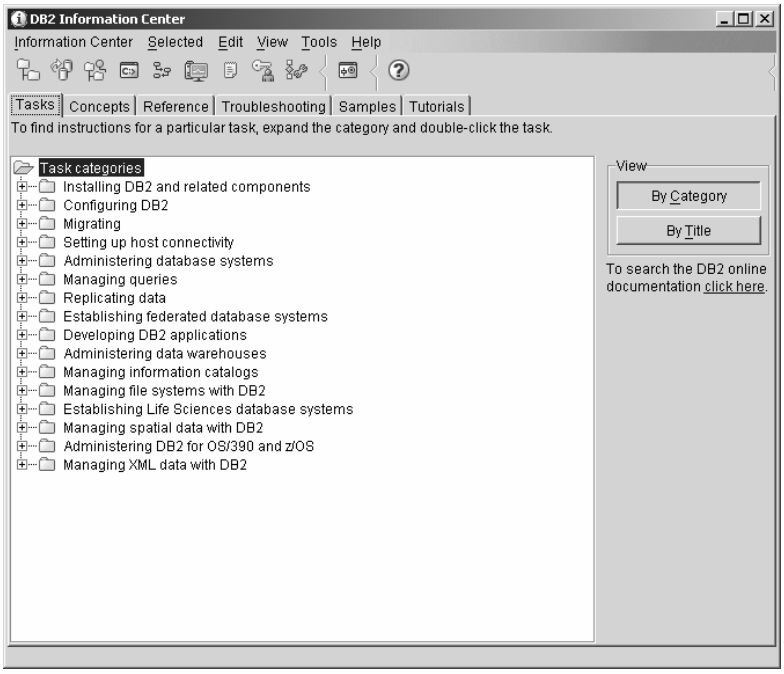
Устранение неисправностей. Это представление может использоваться для быстрого нахождения подробной информации о любой встреченной в ходе использования DB2 UDB проблеме. Часто эта информация содержит рекомендации, следуя которым можно разрешить проблему.

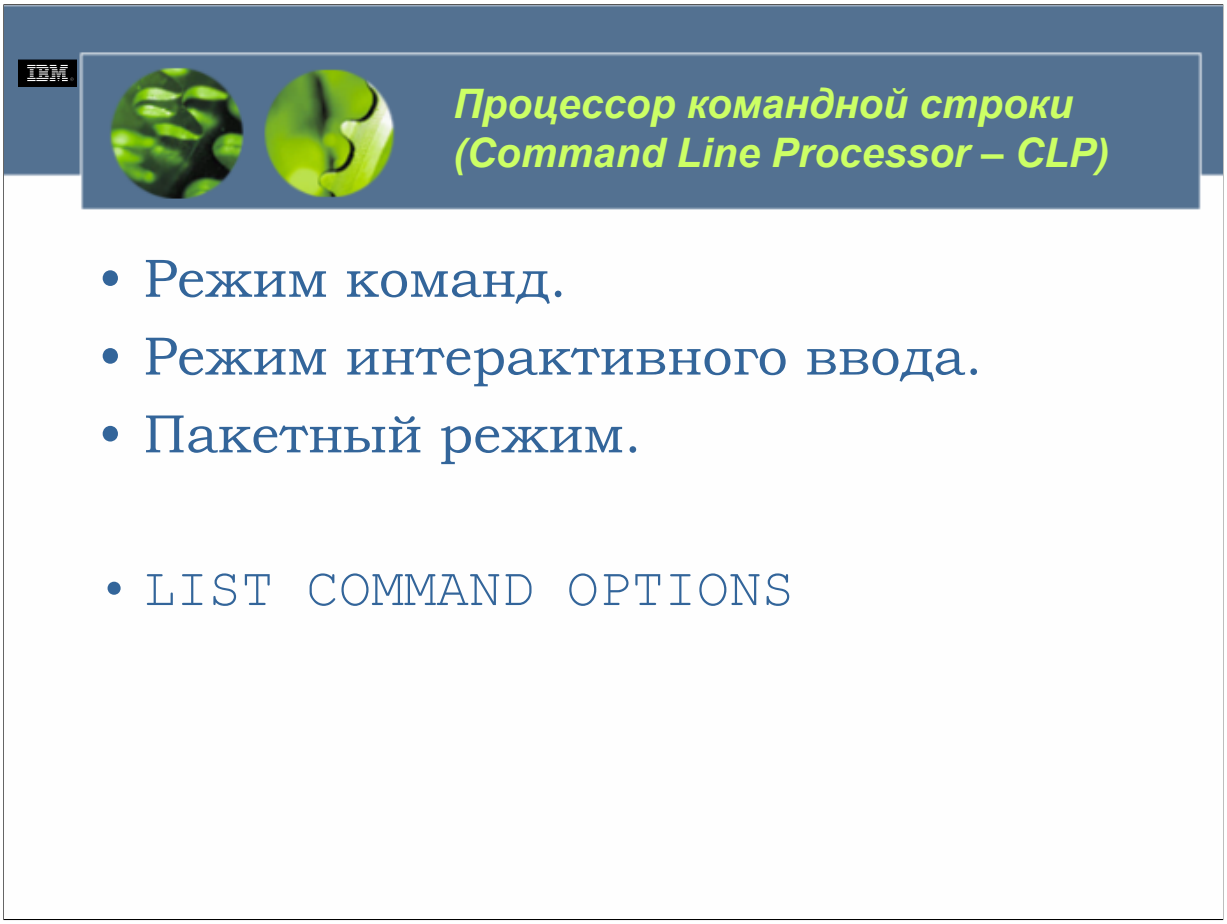
Примеры. Данное представление может использоваться для просмотра описаний и исходного кода примеров программ, которые предоставляются с каждой редакцией Developer's Edition DB2 UDB или DB2 UDB Application Development Client. Когда представленные примеры программ просматриваются с использованием информационного центра, они для упрощения понимания форматируются цветом и гипертекстовыми ссылками.

Учебники. Это представление может использоваться для обнаружения учебников, которые можно использовать для быстрого обучения выполнению определенной задачи или использования определенного инструмента, являющегося частью семейства DB2 UDB.

После выбора наиболее подходящего доступного представления пользователь проходит через список представленных тем до тех пор, пока не будет найдена нужная тема. Затем, после выбора подходящей темы, соответствующее окно документации DB2 UDB может быть отображено в окне веб-браузера.

Информационный центр (Information Center)





WCF03.0

Процессор командной строки (Command Line Processor – CLP) является текстовым приложением, позволяющим пользователям набирать команды DB2 UDB, системные команды и операторы SQL, а также просматривать результаты выполненных операторов/команд. Процессор командной строки можно запустить в трех различных режимах:

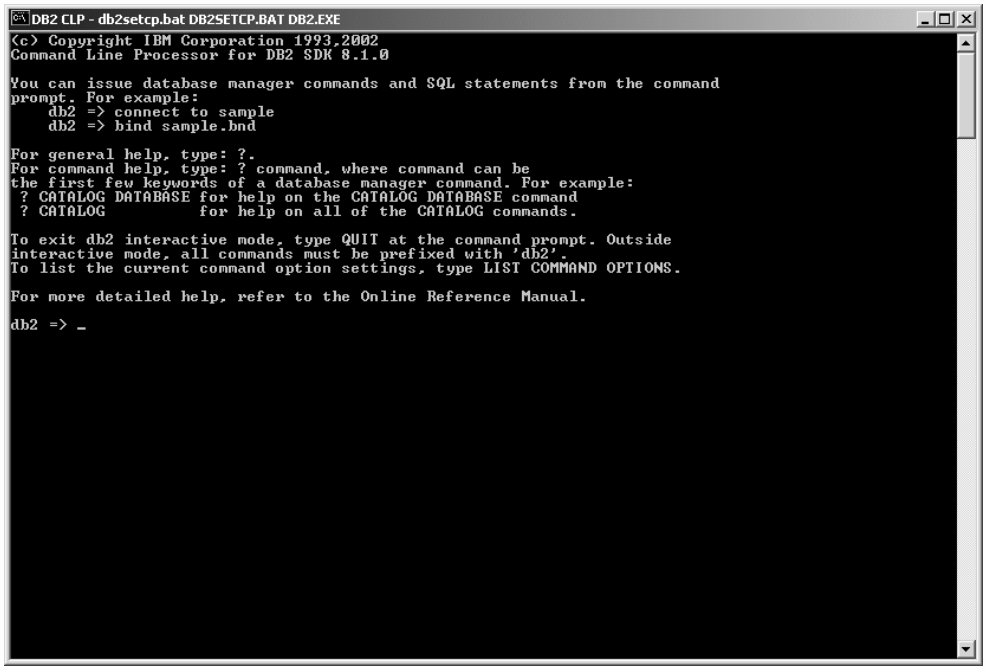
Режим команд. Когда процессор командной строки запущен в режиме команд, пользователь просто вводит в системном приглашении команду DB2 UDB, команду системы или оператор SQL с предшествующими символами “db2”. (Например, команда подключения к SAMPLE вводится как db2 connect to SAMPLE). Если команда содержит символы, имеющие для используемой операционной системы особое значение, они должны быть заключены в символы кавычек для обеспечения их должного исполнения (например, db2 “select count(*) from employee”). Если команда для исполнения слишком большая и выходит за пределы одной строки, в конце строки, которая должна быть продолжена, можно поместить пробел, за которым следует символ продолжения строки (\), а оставшаяся часть команды может следовать на новой строке.

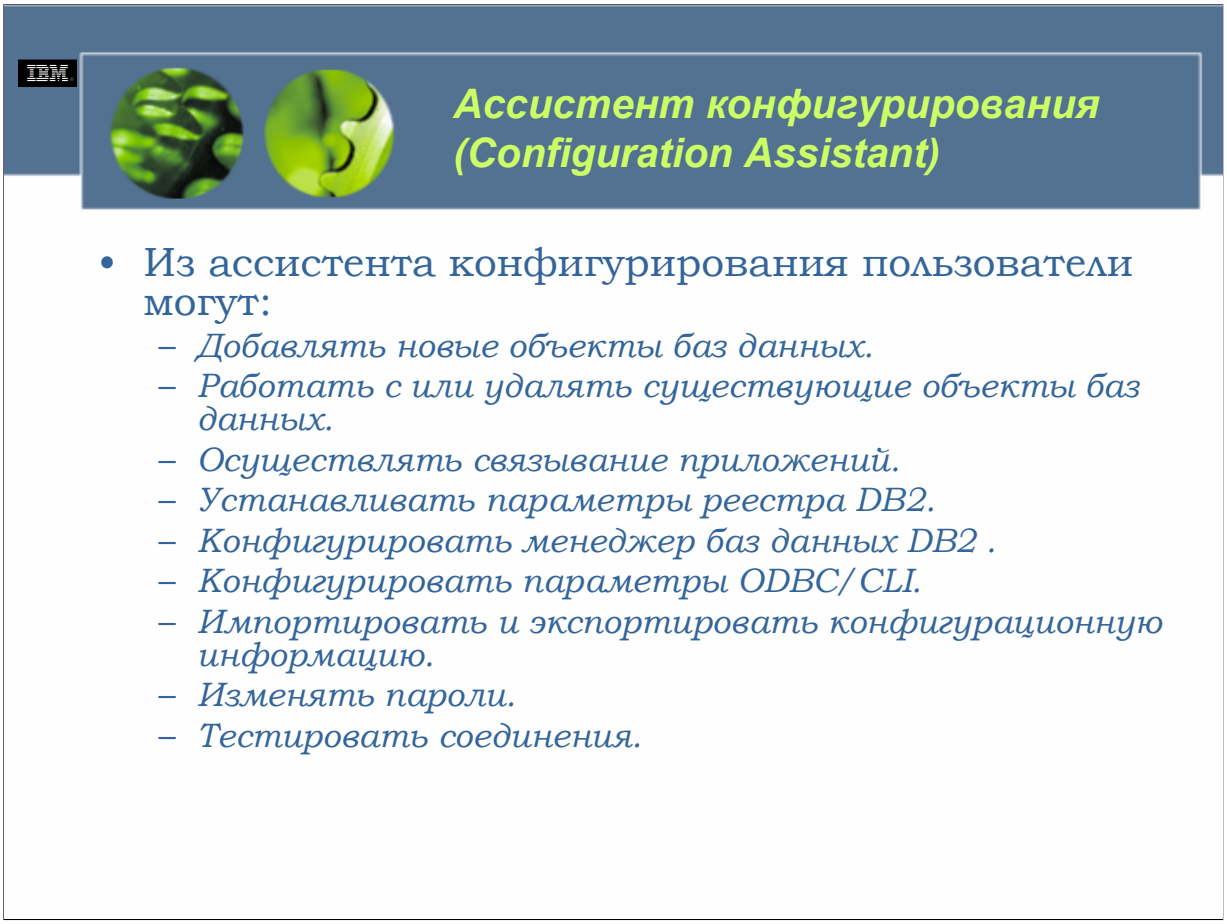
Режим интерактивного ввода. Когда процессор командной строки запущен в режиме интерактивного ввода, префикс “db2” предоставляется автоматически (на что указывает вводное приглашение **db2 =>**). Для запуска процессора командной строки в режиме интерактивного ввода вы просто вводите в ответ на системное приглашение команду “db2”. Для выхода из интерактивного режима вы вводите в ответ на приглашение процессора командной строки команду “quit”. За исключением этого правила, применяющиеся к использованию режима команд процессора командной строки, применяются также к использованию режима интерактивного ввода.

Пакетный режим. Когда процессор командной строки запущен в пакетном режиме, предполагается, что все исполняемые команды и/или операторы SQL были сохранены в текстовом ASCII файле. (Команды/операторы, хранящиеся в этом файле, не должны предваряться символами “db2”). Чтобы запустить процессор командной строки в пакетном режиме, вы просто вводите в ответ на системное приглашение команду db2 –f *xxxxxxxx* (где *xxxxxxxx* является именем файла, содержащего набор команд для исполнения).

Имеются различные опции командной строки, которые могут быть указаны при вызове процессора командной строки; список всех доступных опций можно получить, выполнив команду LIST COMMAND OPTIONS либо из системной командной строки, либо из командной строки процессора командной строки (когда он запущен в режиме интерактивного ввода).

Процессор командной строки (Command Line Processor – CLP)





WCF03.0

Ассистент конфигурирования (Configuration Assistant; рис. 2-23) является интерактивным графическим приложением, позволяющим пользователям конфигурировать клиенты таким образом, что они могут получать доступ к базам данных, хранящимся на удаленных серверах DB2; для получения доступа к экземпляру или базе данных на другом сервере или системе, эта система должна быть каталогизирована в каталоге узлов рабочей станции клиента, а информация о базе данных должна быть каталогизирована в каталоге баз данных рабочей станции клиента. Ассистент конфигурирования предоставляет способ поддержки списка баз данных, с которыми могут соединяться пользователи/приложения. С его помощью пользователи могут быстро каталогизировать узлы и базы данных без необходимости знания неотъемлемых сложностей, необходимых для осуществления этих задач. Ассистент конфигурирования может также действовать в качестве облегченной альтернативы центра управления в ситуациях, когда не был установлен полный набор доступных графических инструментов.

Из ассистента конфигурирования пользователи могут:

•Добавлять новые объекты баз данных.

•Работать с или удалять существующие объекты баз данных.

•Осуществлять связывание приложений.

•Устанавливать параметры реестра DB2.

•Конфигурировать менеджер баз данных DB2 .

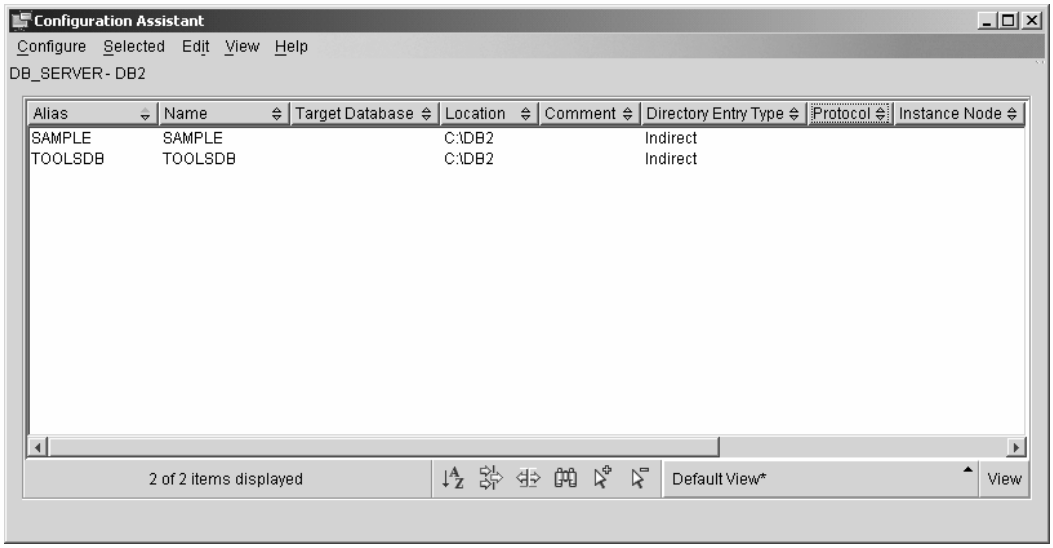
•Конфигурировать параметры ODBC/CLI.

•Импортировать и экспортировать конфигурационную информацию.

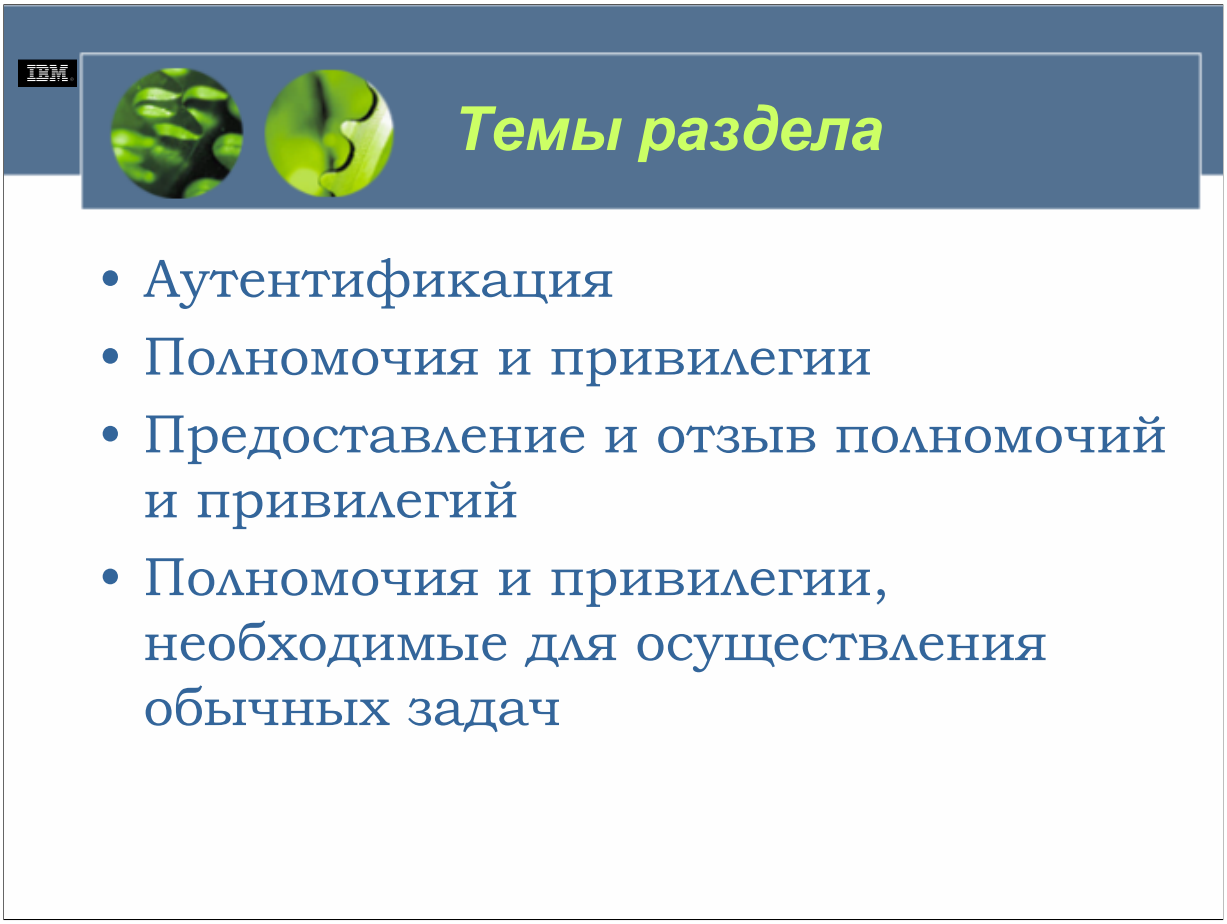
•Изменять пароли.

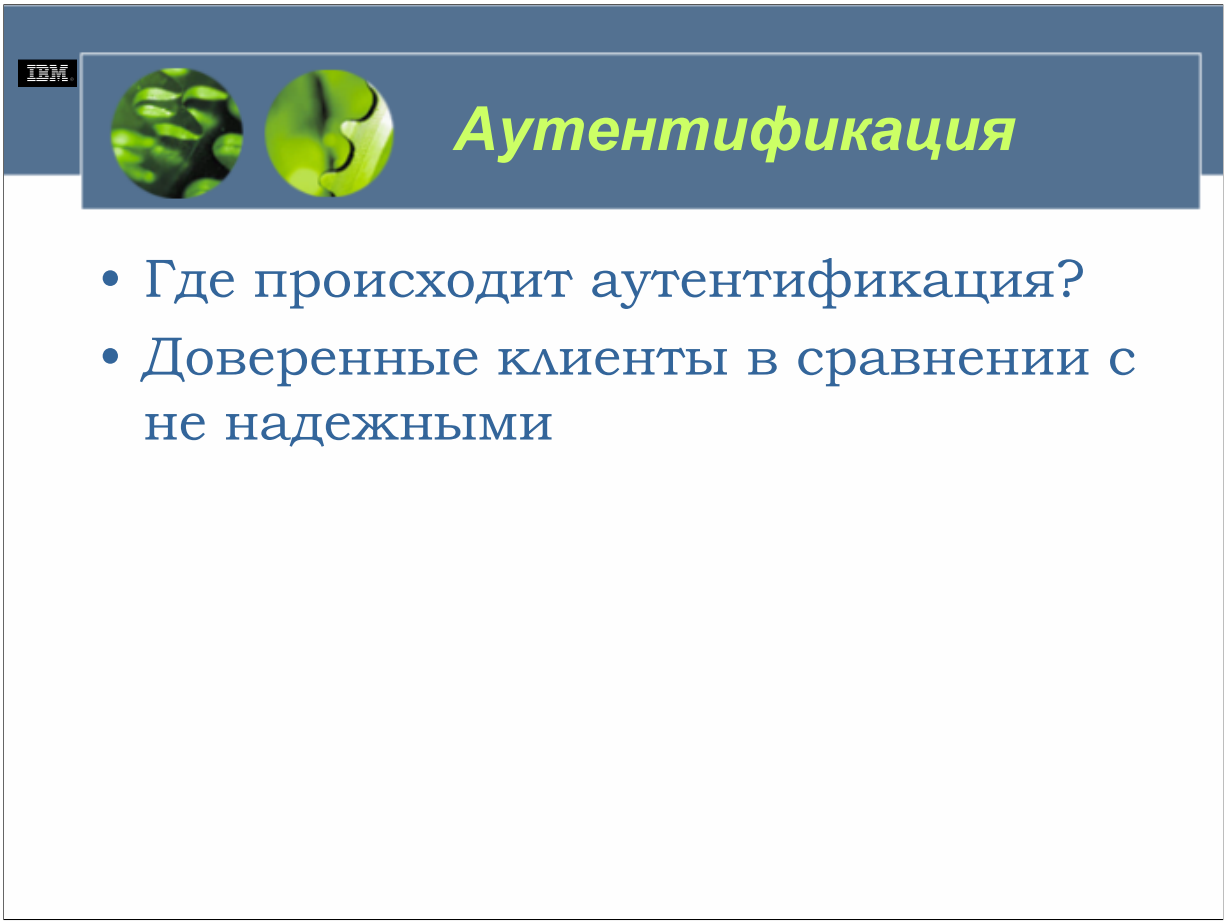
•Тестировать соединения.

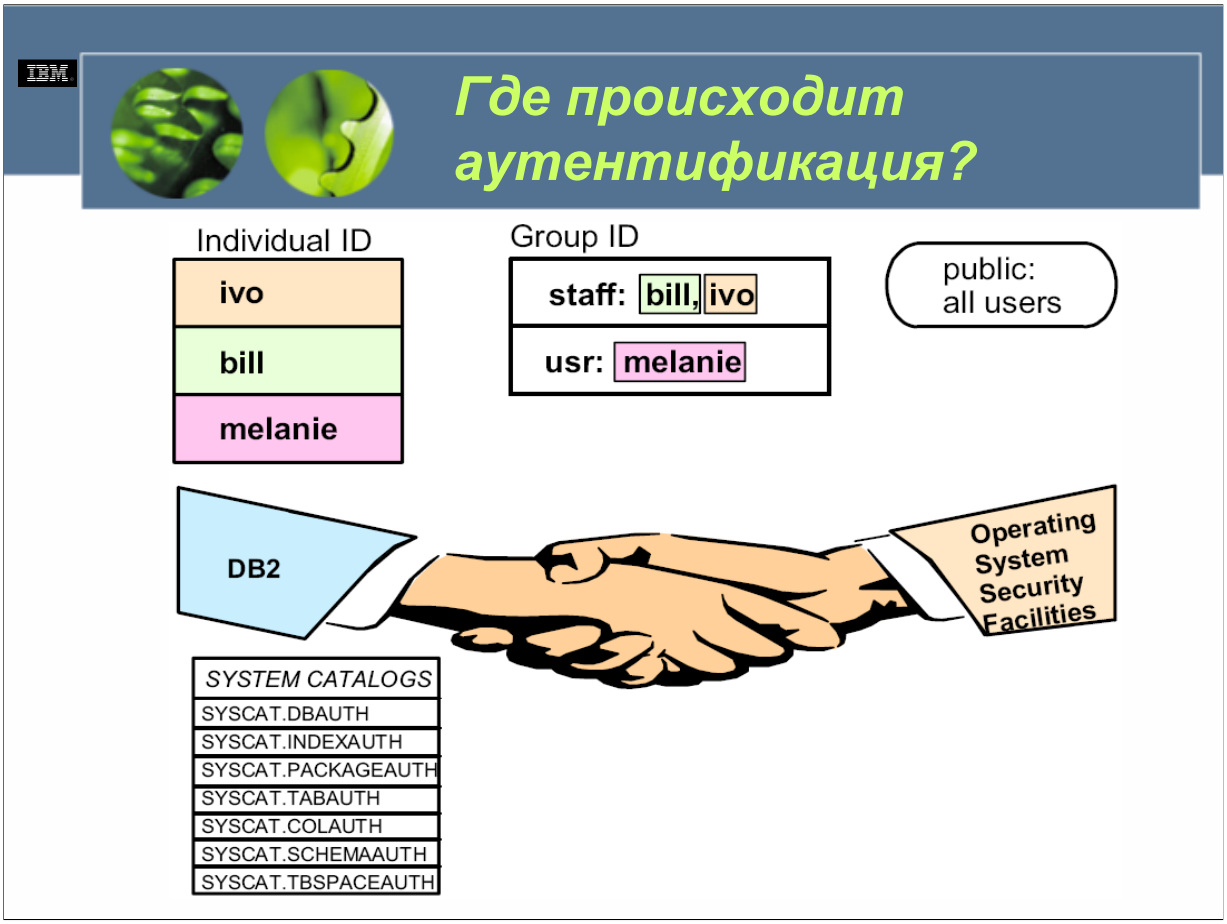
Ассистент конфигурирования (Configuration Assistant)



WCF03.0







WCF03.0

Первым порталом безопасности, который должно пройти большинство пользователей на своем пути к получению доступа к экземпляру DB2 UDB или базе данных, является процесс, известный как *аутентификация*. Назначением аутентификации является подтверждение того, что пользователь на самом деле является тем, за кого себя выдает. Обычно аутентификация осуществляется путем внешних средств безопасности, не являющихся частью DB2 UDB. Эти средства безопасности могут быть частью операционной системы (как в случае с AIX, Solaris, Linux, HP-UX, Windows 2000/NT и многих других), они могут быть отдельным добавочным продуктом (например, службы безопасности среды распределенных вычислений (DCE)), или они могут вовсе не существовать (что по умолчанию является верным для Windows 95, Windows 98 и Windows Millennium Edition). Если средство безопасности все же существует, перед тем, как пользователь может быть аутентифицирован, ему нужно представить два элемента: уникальный ID пользователя и соответствующий пароль. ID пользователя идентифицирует пользователя для средства безопасности, тогда как пароль, который является информацией, известной лишь только пользователю и средству безопасности, используется для подтверждения того, что пользователь на самом деле является тем, за кого себя выдает.

WCF03.0

В версии 8.1 DB2 UDB доступны следующие типы аутентификации:

SERVER. Аутентификация происходит на рабочей станции сервера с использованием средств безопасности, предоставленных операционной системой сервера. (ID пользователя и пароль, предоставленные пользователем, желающим подключиться к экземпляру или соединиться с базой данных, сравниваются с комбинацией ID пользователя и пароля, хранящейся на сервере, для определения того, разрешен ли пользователю доступ к экземпляру/базе данных). По умолчанию при первом создании экземпляра используется этот тип аутентификации.

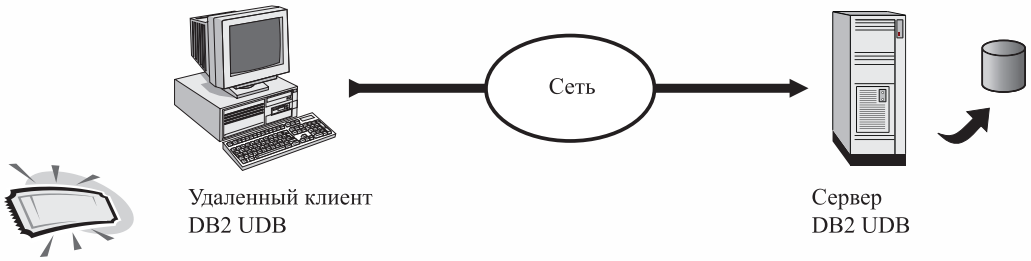
SERVER_ENCRYPT. Аутентификация происходит на рабочей станции сервера с использованием средства безопасности, предоставленного операционной системой сервера. Однако пароль, предоставленный пользователем, желающим подключиться к экземпляру или соединиться с базой данных, сохраненный на сервере, может быть зашифрован на рабочей станции клиента перед отправкой его на рабочую станцию сервера для проверки.

CLIENT. Аутентификация происходит на рабочей станции клиента или разделе базы данных, где вызывается клиентское приложение, с использованием средств безопасности, предоставленных операционной системой клиента, предполагая, что они доступны. Если средства безопасности недоступны, аутентификация обрабатывается несколько другим способом. (ID пользователя и пароль, предоставленные пользователем, желающим подключиться к экземпляру или соединиться с базой данных, сравниваются с комбинацией ID пользователя и пароля, хранящимися в клиенте/узле, для определения того, разрешен ли пользователю доступ к экземпляру или базе данных).

KERBEROS. Аутентификация происходит на рабочей станции сервера с использованием средства безопасности, поддерживающего протокол безопасности Kerberos. Протокол безопасности Kerberos является протоколом, осуществляющим аутентификацию в качестве службы третьей стороны с использованием обычной криптографии для создания общего секретного ключа. Ключ становится мандатом, используемым для подтверждения идентичности пользователей каждый раз, когда запрашиваются локальные или сетевые службы; это устраняет необходимость передавать через сеть ID пользователя и пароль в виде текста ASCII. (Если как клиент, так и сервер поддерживают протокол безопасности Kerberos, ID пользователя и пароль, предоставленные пользователем, желающим подключиться к экземпляру или соединиться с базой данных, шифруются на рабочей станции клиента и посылаются на сервер для подтверждения). Следует заметить, что тип аутентификации KERBEROS поддерживается лишь на клиентах и серверах, использующих операционную систему Windows 2000, Windows XP или Window .NET. Кроме того, рабочие станции как клиента, так и сервера должны либо принадлежать к одному и тому же домену Windows, либо принадлежать доверенным доменам.

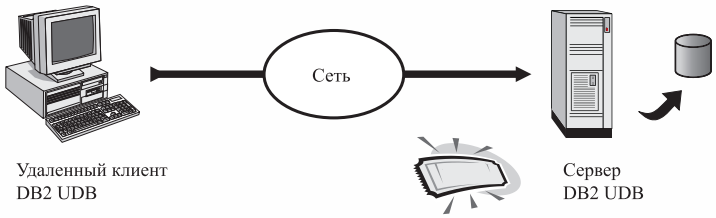
KRB_SERVER_ENCRYPT. Аутентификация происходит на рабочей станции сервера с использованием метода аутентификации либо KERBEROS, либо SERVER_ENCRYPT. Если тип аутентификации клиента установлен в KERBEROS, аутентификация осуществляется на сервере с использованием системы безопасности Kerberos. С другой стороны, если в качестве типа аутентификации клиента установлено что-то еще помимо KERBEROS или если служба аутентификации Kerberos недоступна, сервер действует, как если бы был указан тип аутентификации SERVER_ENCRYPT, и применяются правила этого способа аутентификации.

Аутентификация на клиенте

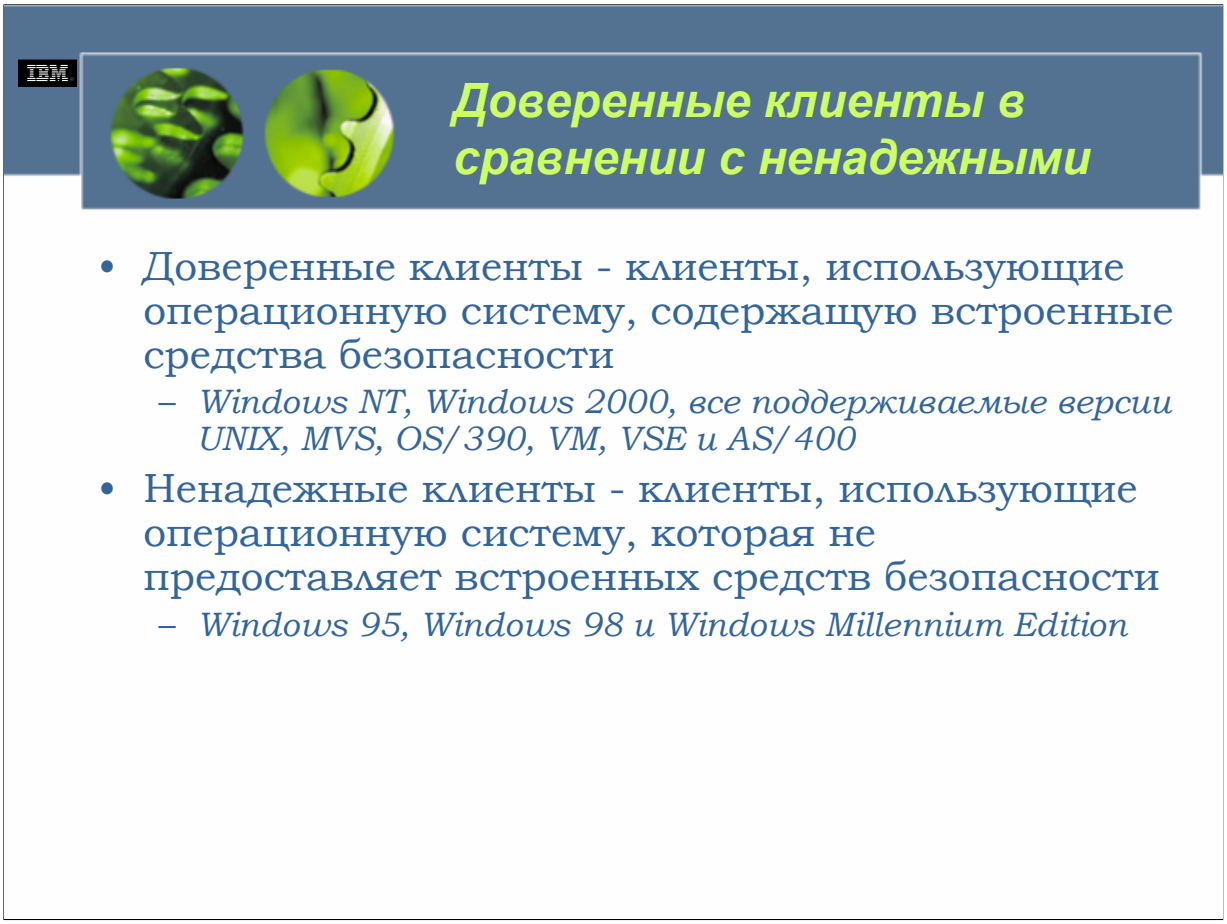


Тип аутентификации клиента	Тип аутентификации сервера	Используемый тип аутентификации
CLIENT	CLIENT	CLIENT

Аутентификация на сервере

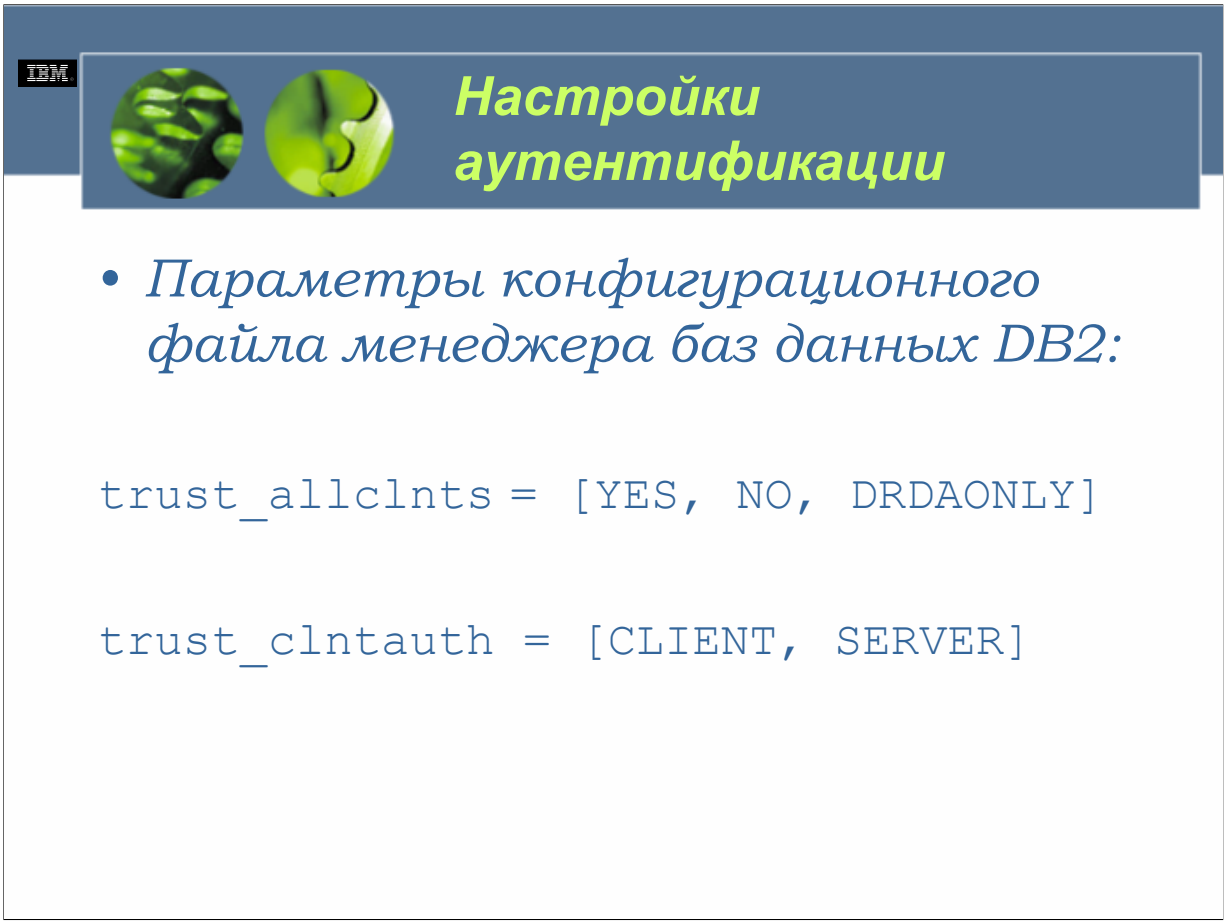


Тип аутентификации клиента	Тип аутентификации сервера	Используемый тип аутентификации
SERVER	SER VER	SERVER
SERVER	SERVER_ENCRYPT	SERVER
SERVER_ENCRYPT	SERVER_ENCRYPT	SERVER_ENCRYPT
KERBEROS	KERBEROS	KERBEROS
KERBEROS	KRB_SERVER_ENCRYPT	KERBEROS
KRB_SERVER_ENCRYPT	KRB_SERVER_ENCRYPT	KRB_SERVER_ENCRYPT
SERVER	KRB_SERVER_ENCRYPT	SERVER_ENCRYPT
SERVER_ENCRYPT	KRB_SERVER_ENCRYPT	SERVER_ENCRYPT



WCF03.0

Если как сервер, так и клиент сконфигурированы для использования типа аутентификации CLIENT, аутентификация происходит на рабочей станции клиента (если база данных не является многораздельной) или в разделе базы данных, из которого вызывается клиентское приложение (если база данных является многораздельной), с использованием средств безопасности, предоставляемых операционной системой клиента. Но что случится, если рабочая станция использует операционную систему, не содержащую тесно интегрированные средства безопасности, и если не были предоставлены отдельные добавочные средства безопасности? Не компрометирует ли такая конфигурация безопасность? Ответом является нет. Однако в таких окружениях менеджер баз данных DB2 для экземпляра на сервере должен иметь возможность определять, какие клиенты будут отвечать за проверку пользователей и какие клиенты должны оставить обработку аутентификации пользователя серверу. Для принятия этого решения клиенты, использующие операционную систему, содержащую встроенные средства безопасности (например, Windows NT, Windows 2000, все поддерживаемые версии UNIX, MVS, OS/390, VM, VSE и AS/400), классифицируются в качестве *доверенных клиентов*, а клиенты, использующие операционную систему, которая не предоставляет встроенных средств безопасности (например, Windows 95, Windows 98 и Windows Millennium Edition), классифицируются в качестве *ненадежных клиентов*.



WCF03.0

Параметр *trust_allclnts* конфигурационного файла менеджера баз данных DB2 помогает менеджеру баз данных DB2 для экземпляра сервера решить, следует ли доверять клиентам или нет. Если этот конфигурационный параметр установлен в YES (что является значением по умолчанию), менеджер баз данных DB2 предполагает, что любой клиент, получающий доступ к экземпляру, является доверенным клиентом, и что аутентификация будет происходить на клиенте. Однако если этот конфигурационный параметр установлен в NO, менеджер баз данных DB2 предполагает, что для доступа к серверу будут использоваться один или более ненадежных клиента; следовательно, все пользователи должны аутентифицироваться на сервере. (Если этот конфигурационный параметр установлен в DRDAONLY, в качестве доверенных будут рассматриваться лишь клиенты MVS, OS/390, VM, VSE и OS/400). Важно заметить, что независимо от того, как установлен параметр *trust_allclnts*, каждый раз, когда к экземпляру или базе данных пытается получить доступ ненадежный клиент, аутентификация пользователя всегда происходит на сервере.

В некоторых случаях может быть желательным аутентифицировать пользователей на сервере, даже когда ненадежным клиентам доступ не нужен. В таких случаях конфигурационный параметр *trust_clntauth* конфигурационного файла DB2 Database Manager может использоваться для контроля того, где будут проверяться доверенные клиенты. При принятии для данного параметра значения по умолчанию (CLIENT), аутентификация для доверенных клиентов будет происходить на рабочей станции клиента. Однако если значение этого параметра изменено на SERVER, аутентификация для всех доверенных клиентов будет происходить на сервере.

WCF03.0

После того, как пользователь был аутентифицирован, и было установлено соединение с экземпляром или базой данных, менеджер баз данных DB2 оценивает *полномочия* и *привилегии*, которые были назначены пользователю (они могут быть назначены непосредственно пользователю, или они могут быть получены косвенно из групповых привилегий, которые были назначены группе, членом которой является пользователь), чтобы определить, какие операции пользователю разрешено выполнять. Полномочия выражают набор привилегий и/или прав для осуществления высокоуровневых административных операций и операций с утилитами и сопровождения в отношении экземпляра или базы данных. Привилегии, с другой стороны, выражают права осуществлять определенные действия в отношении определенных ресурсов базы данных (таких как таблицы и производные таблицы). Полномочия и привилегии действуют совместно для управления доступом к менеджеру баз данных DB2 , например, к одной или более базе данных, запущенных под контролем этого экземпляра, и к определенным объектам базы данных. Пользователи могут работать лишь с теми объектами, для которых им была предоставлена соответствующая авторизация – т.е. необходимое полномочие или привилегия.

Function	SYSADM	SYSCTRL	SYSMAINT	DBADM	LOAD
MIGRATE DATABASE	YES				
UPDATE DBM CFG	YES				
GRANT/REVOKE DBADM	YES				
UPDATE db/node/dcs directories	YES	YES			
FORCE USERS OFF SYSTEM	YES	YES			
CREATE/DROP DATABASE	YES	YES			
CREATE/DROP/ALTER TABLE SPACE	YES	YES			
RESTORE TO NEW DATABASE	YES	YES			
UPDATE DB CFG	YES	YES	YES		
BACKUP DATABASE or TABLE SPACE	YES	YES	YES		
RESTORE TO EXISTING DATABASE	YES	YES	YES		
PERFORM ROLLFORWARD RECOVERY	YES	YES	YES		
START/STOP DATABASE INSTANCE	YES	YES	YES		
RESTORE TABLE SPACE	YES	YES	YES		
RUN TRACE	YES	YES	YES		
TAKE DBM or DB SNAPSHOTS	YES	YES	YES		
QUERY TABLE SPACE STATE	YES	YES	YES	YES	YES
UPDATE LOG HISTORY FILES	YES	YES	YES	YES	
QUIESCE TABLE SPACE	YES	YES	YES	YES	YES
REORG TABLE	YES	YES	YES	YES	
RUN RUNSTATS UTILITY	YES	YES	YES	YES	YES
READ LOG FILES	YES			YES	
CREATE/ACTIVATE/DROP EVENT MONITORS	YES			YES	

WCF03.0

DB2 UDB использует пять различных уровней полномочий для контроля того, как пользователи осуществляют административные операции и операции по сопровождению экземпляра или базы данных. Этими пятью уровнями являются:

Полномочие системного администратора (SYSADM)

Полномочие управления системой (SYSCTRL)

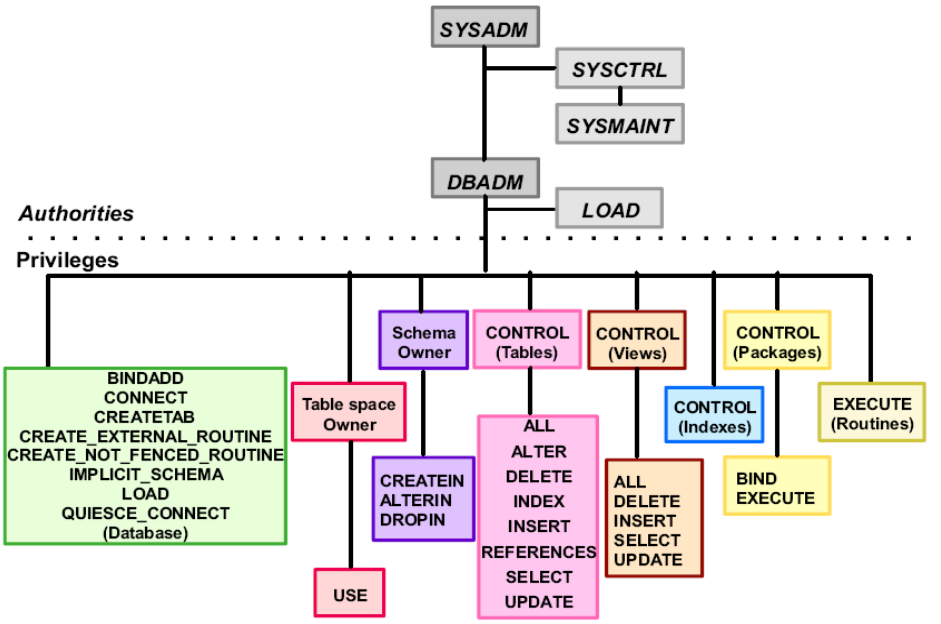
Полномочие обслуживания системы (SYSMAINT)

Полномочие администратора базы данных (DBADM)

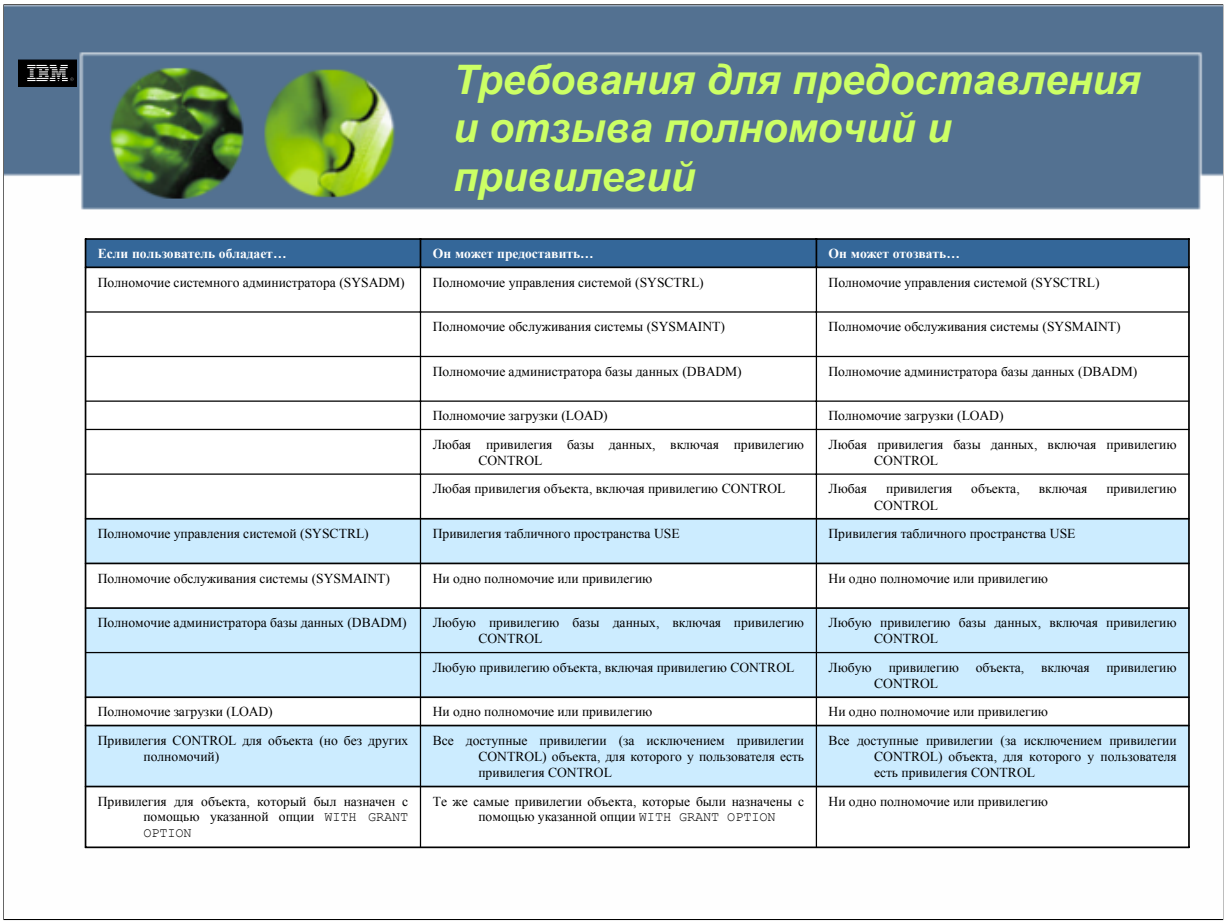
Полномочие загрузки (LOAD)

Первые три из этих уровней применяются к экземпляру менеджера баз данных DB2 (и ко всем базам данных, которые управляются этим экземпляром), тогда как оставшиеся два применяются лишь к определенным базам данных в рамках экземпляра. Более того, три полномочия уровня экземпляра могут назначаться лишь группам; имена групп, которым назначаются эти полномочия, хранятся в конфигурационном файле менеджера баз данных DB2 , который связан с данным экземпляром. Напротив, два полномочия уровня базы данных могут назначаться отдельным пользователям и/или группе пользователей; группы и пользователи, которым назначены полномочия уровня базы данных, записываются в таблицах системного каталога той базы данных, к которой полномочия относятся.

Привилегии



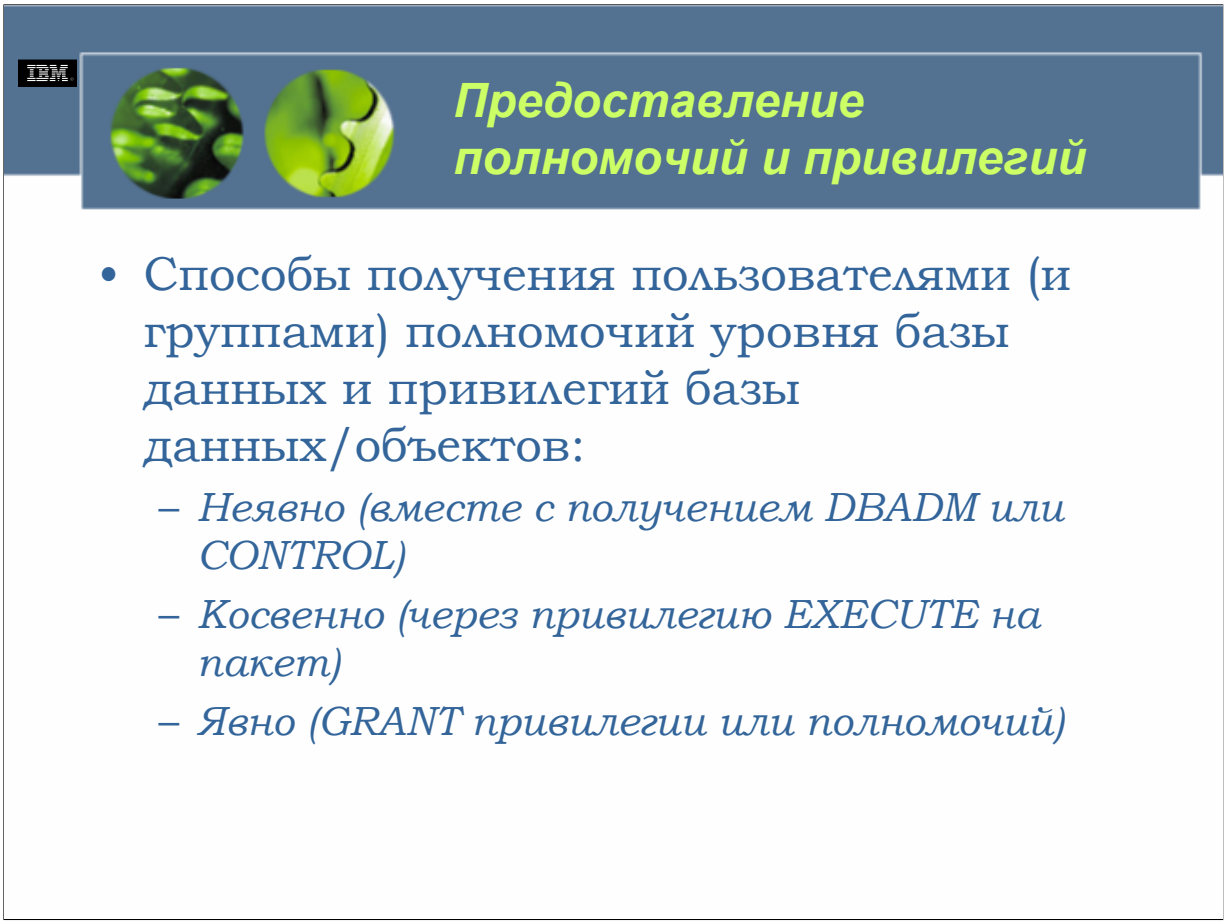
WCF03.0



Если пользователь обладает…	Он может предоставить…	Он может отозвать…
Полномочие системного администратора (SYSADM)	Полномочие управления системой (SYSCTRL)	Полномочие управления системой (SYSCTRL)
	Полномочие обслуживания системы (SYSMAINT)	Полномочие обслуживания системы (SYSMAINT)
	Полномочие администратора базы данных (DBADM)	Полномочие администратора базы данных (DBADM)
	Полномочие загрузки (LOAD)	Полномочие загрузки (LOAD)
	Любая привилегия базы данных, включая привилегию CONTROL	Любая привилегия базы данных, включая привилегию CONTROL
	Любая привилегия объекта, включая привилегию CONTROL	Любая привилегия объекта, включая привилегию CONTROL
Полномочие управления системой (SYSCTRL)	Привилегия табличного пространства USE	Привилегия табличного пространства USE
Полномочие обслуживания системы (SYSMAINT)	Ни одно полномочие или привилегию	Ни одно полномочие или привилегию
Полномочие администратора базы данных (DBADM)	Любую привилегию базы данных, включая привилегию CONTROL	Любую привилегию базы данных, включая привилегию CONTROL
	Любую привилегию объекта, включая привилегию CONTROL	Любую привилегию объекта, включая привилегию CONTROL
Полномочие загрузки (LOAD)	Ни одно полномочие или привилегию	Ни одно полномочие или привилегию
Привилегия CONTROL для объекта (но без других полномочий)	Все доступные привилегии (за исключением привилегии CONTROL) объекта, для которого у пользователя есть привилегия CONTROL	Все доступные привилегии (за исключением привилегии CONTROL) объекта, для которого у пользователя есть привилегия CONTROL
Привилегия для объекта, который был назначен с помощью указанной опции `WITH GRANT OPTION`	Те же самые привилегии объекта, которые были назначены с помощью указанной опции `WITH GRANT OPTION`	Ни одно полномочие или привилегию

WCF03.0

Уровни полномочий и привилегии контролируют не только то, что пользователи могут и что не могут делать, они также контролируют, какие полномочия и привилегии пользователь может предоставлять и отзывать у других пользователей и групп.



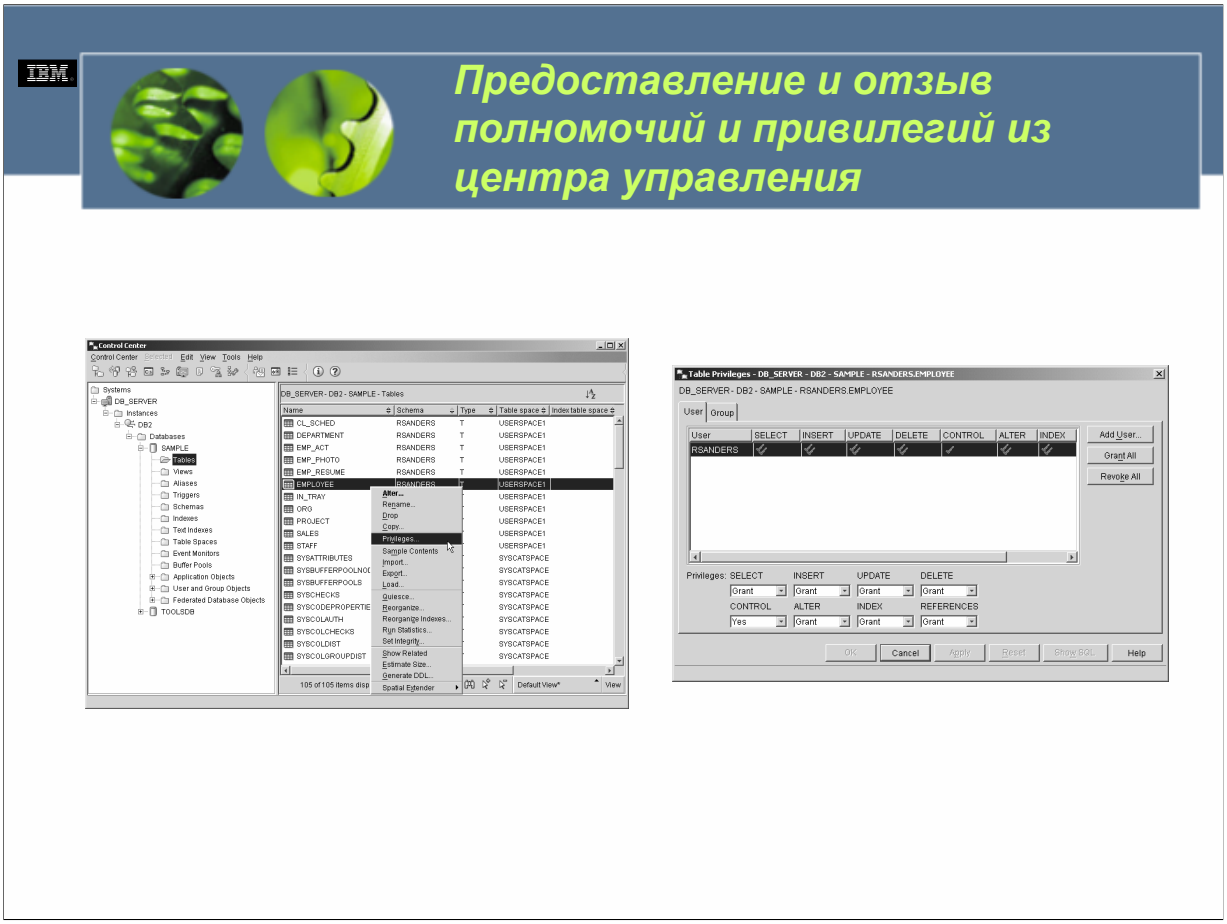
WCF03.0

Есть три различных способа получения пользователями (и группами) полномочий уровня базы данных и привилегий базы данных/объектов. Это:

Неявный. Когда пользователь создает базу данных, он неявно получает полномочие DBADM для этой базы данных вместе с несколькими привилегиями базы данных. Аналогично, когда пользователь создает объект базы данных, он неявно получает все доступные для данного объекта привилегии вместе с возможностью предоставления любой комбинации этих привилегий (за исключением привилегии CONTROL) другим пользователям и группам. Привилегии могут также неявно предоставляться каждый раз, когда пользователю явным образом предоставляется привилегия более высокого уровня (например, если пользователю явно предоставляется привилегия CONTROL для табличного пространства, он неявно получит также привилегию USE для этого табличного пространства). Помните, что такие неявно предоставляемые привилегии не отзываются автоматически, когда отзывается привилегия более высокого уровня, которая вызвала их неявное предоставление.

Косвенный. Косвенно назначаемые привилегии обычно связаны с пакетами; когда пользователь исполняет пакет, для выполнения которого требуются привилегии, которых у пользователя нет (например, пакет, удаляющий строку данных из таблицы, требует привилегию DELETE для этой таблицы), пользователю косвенно предоставляются эти привилегии со специальной целью исполнения этого пакета. Косвенно предоставленные привилегии являются временными и не существуют вне той области действия, в которой они были предоставлены.

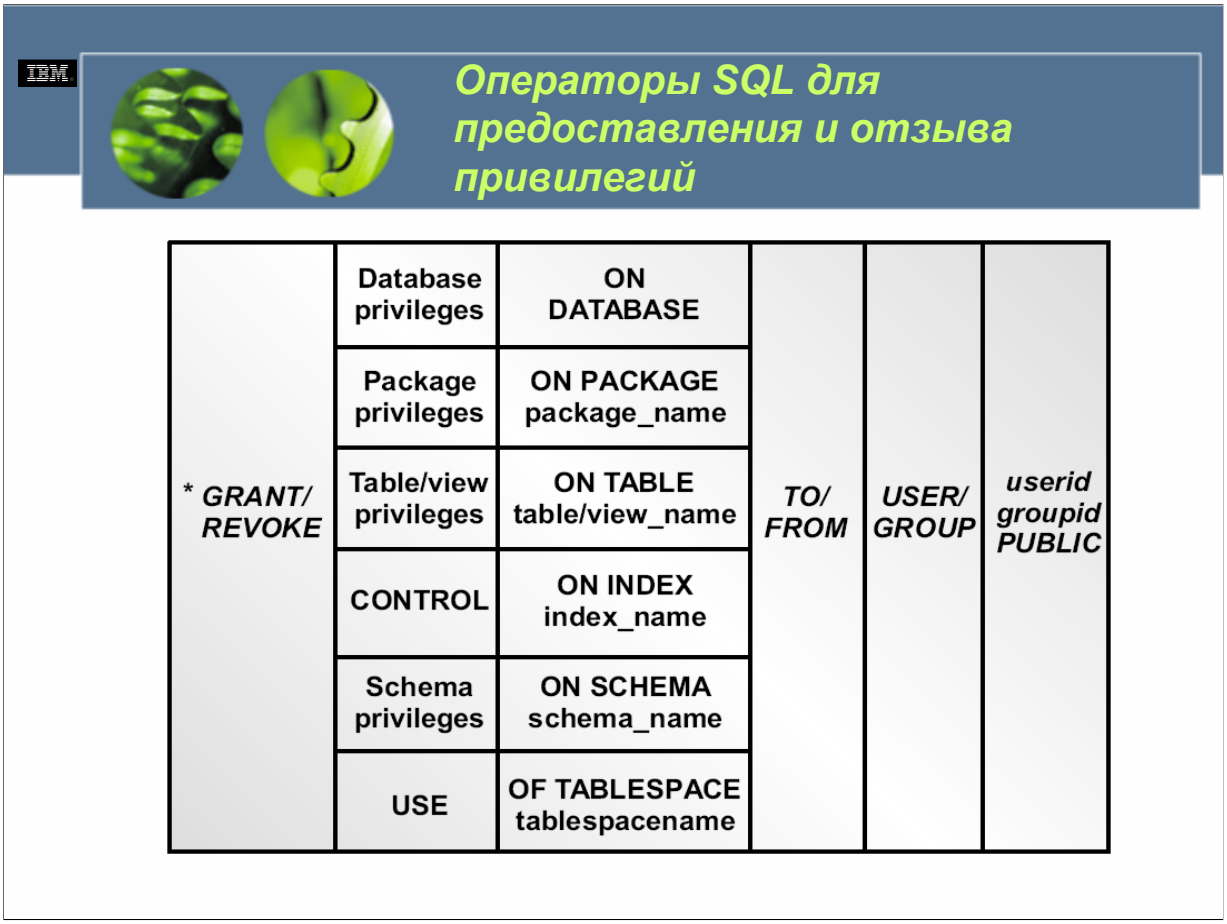
Явный. Полномочия уровня баз данных, привилегии баз данных и привилегии объектов могут быть предоставлены или отозваны у отдельного пользователя или группы пользователей явным образом любым пользователем, у которого есть на это полномочия. Чтобы явным образом предоставлять полномочия для большинства объектов базы данных, у пользователя должно быть полномочие SYSADM, полномочие DBADM или привилегия CONTROL для этого объекта. В качестве альтернативы пользователь может явным образом предоставлять любые привилегии, которые были ему назначены с указанным WITH GRANT OPTION. Для предоставления привилегии CONTROL для любого объекта у пользователя должно быть полномочие SYSADM или DBADM; для предоставления полномочия DBADM пользователь должен обладать полномочием SYSADM.



WCF03.0

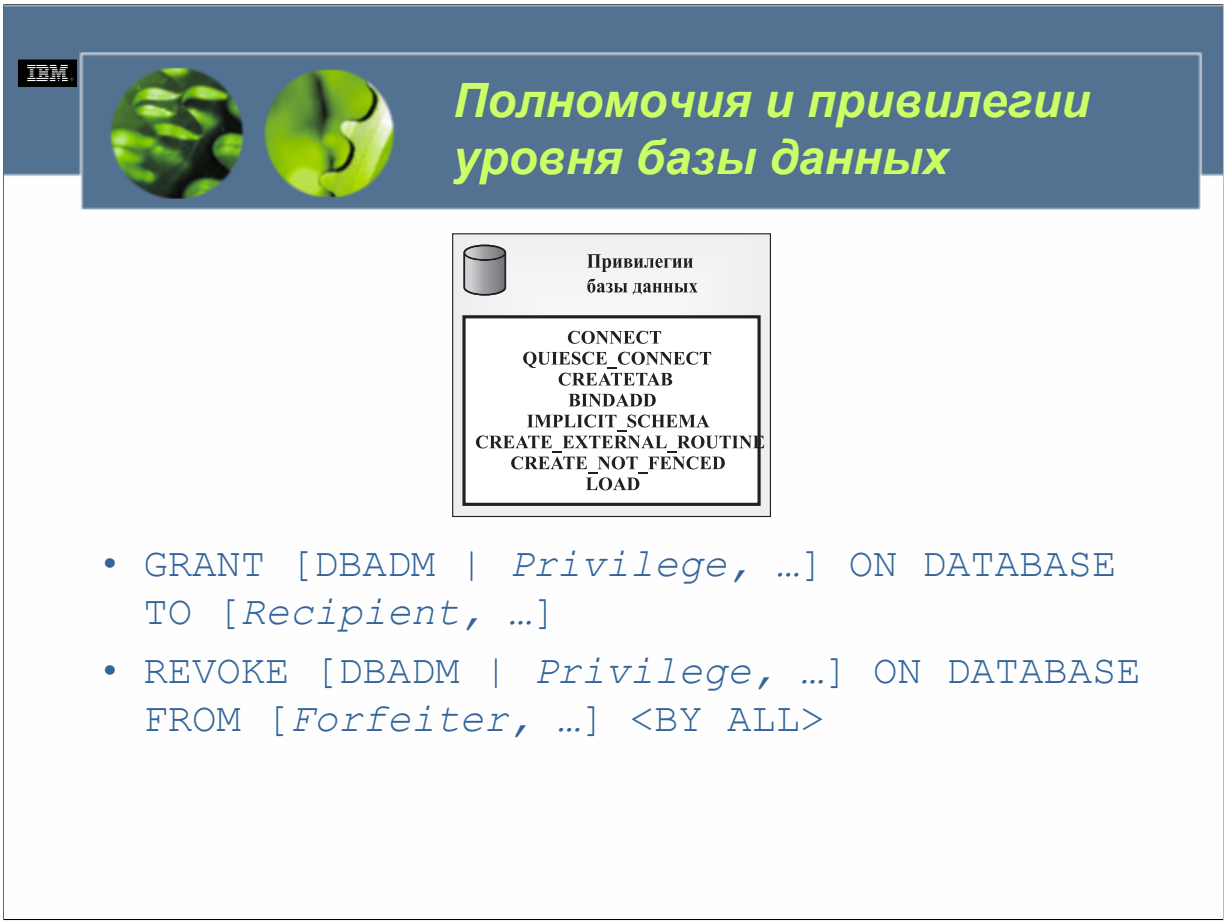
Одним из способов явного предоставления и отзыва полномочий уровня базы данных, а также нескольких доступных привилегий, является использование различных диалоговых окон управления полномочиями и привилегиями, предоставляемыми центром управления. Эти диалоговые окна вызываются выделением имени соответствующих базы данных или объекта, отображенных в панелях центра управления, и выбора пункта либо *Полномочия (Authorities)*, либо *Привилегии (Privileges)* из меню соответствующей базы данных или объекта. На рис. 3-15 показаны элементы меню, которые должны быть выбраны в центре управления для вызова диалога Привилегии таблиц (Table Privileges) для определенной таблицы. На рис. 3-16 показано, как диалог Привилегии таблиц мог бы выглядеть сразу после первоначального создания таблицы. (Одиночная галочка под привилегией означает, что отображенным индивиду или группе была предоставлена эта привилегия; двойная галочка означает, что индивиду или группе была также предоставлена возможность предоставлять эту привилегию другим пользователям и группам).

Чтобы назначать привилегии отдельным пользователям из диалога Привилегии таблиц (или аналогичного диалога для других полномочий/привилегий), вы просто отмечаете определенного пользователя, выделив его элемент в списке признанных пользователей – если нужного пользователя в списке нет, их можно добавить, выбрав кнопку «Добавить пользователя» (“Add User”) – и назначаете соответствующие привилегии (или полномочия), используя выпадающий список «Привилегии» (или «Полномочия») или кнопки «Назначить все» (“Grant All”) или «Аннулировать все» (“Revoke all”). Чтобы назначить привилегии группе пользователей, вы выбираете вкладку «Группы» (“Group”) для отображения списка признанных групп и повторяете процесс (используя кнопку «Добавить группу» (“Add Group”) вместо кнопки «Добавить пользователя» для добавления нужной группы в список, если ее там нет).



WCF03.0

Не все привилегии можно явным образом предоставить пользователям/группам с помощью доступных диалогов управления привилегиями. Однако в ситуациях, когда не существует диалога для привилегий (и в ситуациях, когда вы решаете не использовать центр управления), полномочия уровня базы данных и привилегии базы данных/объекта можно явным образом предоставлять пользователям и/или группам, выполнив соответствующую форму оператора SQL GRANT. Синтаксис оператора SQL GRANT различается в зависимости от предоставляемых полномочий или привилегий – в следующих разделах показан синтаксис, использующийся для предоставления каждого полномочия уровня базы данных и доступной привилегии базы данных/объекта.



WCF03.0

CONNECT. Позволяет пользователю устанавливать соединение с базой данных.

QUIESCE_CONNECT. Позволяет пользователю устанавливать соединение с базой данных, когда она стабилизирована (когда доступ к ней ограничен).

CREATETAB. Позволяет пользователю создавать в базе данных новые таблицы.

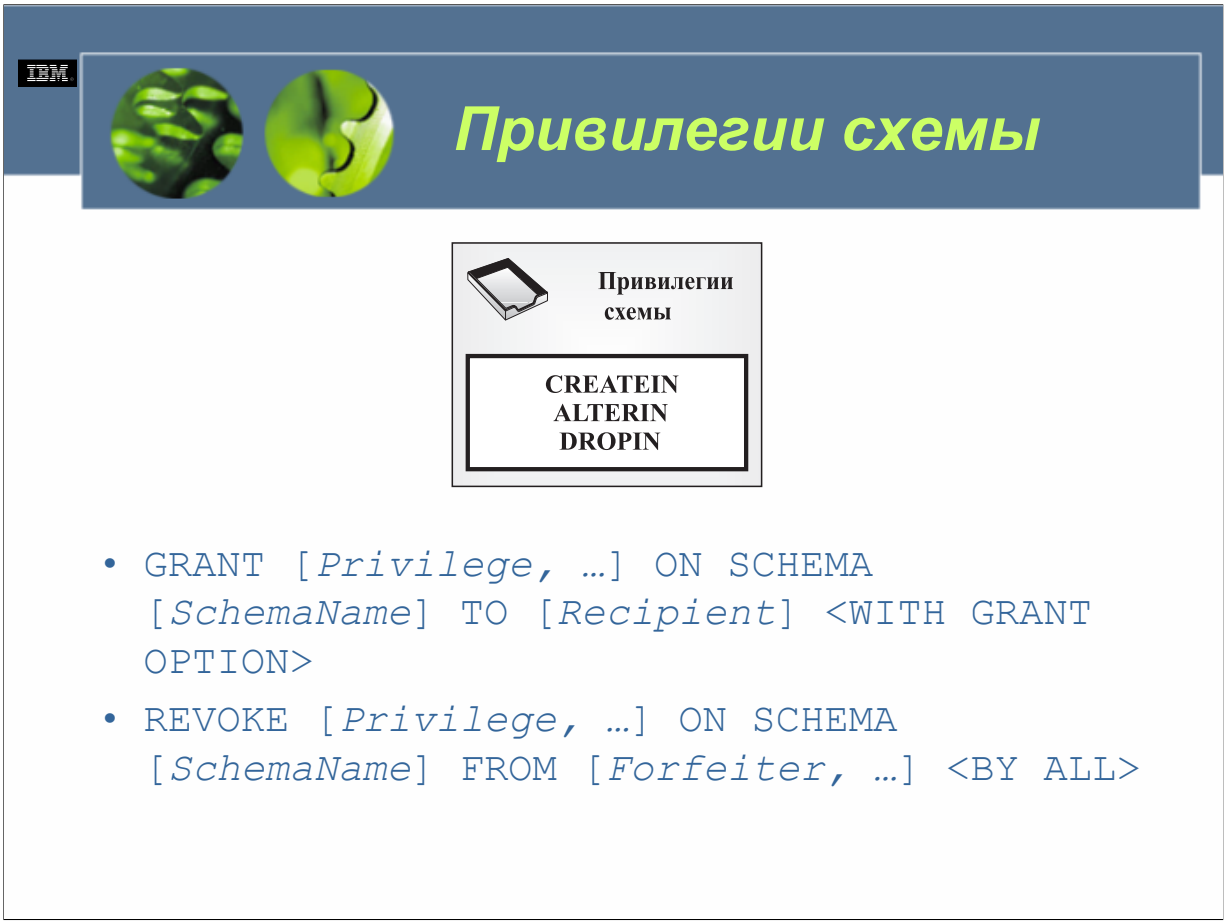
BINDADD. Позволяет пользователю создавать в базе данных пакеты (посредством прекомпилирования для базы данных файлов с исходным кодом приложения со встроенным SQL и/или привязки файлов связывания приложения к базе данных).

CREATE_EXTERNAL_ROUTINE. Позволяет пользователю создавать процедуру, которая может вызываться приложениями и другими пользователями базы данных, и сохранять ее в базе данных.

CREATE_NOT_FENCED. Позволяет пользователю создавать неизолированные пользовательские функции (UDF) и сохранять их в базе данных. (Неизолированные UDF являются UDF, которые считаются достаточно «безопасными», чтобы запускать в процессе рабочего окружения или адресном пространстве менеджера баз данных DB2 . Если функция не зарегистрирована в качестве неизолированной, менеджер баз данных DB2 изолирует свои внутренние ресурсы таким образом, что они не могут использоваться такой функцией).

IMPLICIT_SCHEMA. Позволяет пользователю неявно создавать в базе данных новую схему путем создания объекта и присваивания этому объекту имени схемы, которая отличается от имен всех других схем, которые уже существуют в базе данных.

LOAD. Позволяет пользователю массово загружать данные в одну или более таблиц в базе данных.



WCF03.0

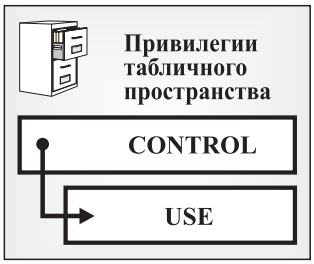
CREATEIN. Позволяет пользователю создавать объекты в данной схеме.

ALTERIN. Позволяет пользователю изменять комментарий, связанный с любым объектом в схеме, или изменять любой объект, находящийся в данной схеме.

DROPIN. Позволяет пользователю удалять (уничтожать) любой объект в данной схеме.

Объекты, которыми можно манипулировать внутри схемы, включают таблицы, производные таблицы, индексы, пакеты, пользовательские типы данных, пользовательские функции, триггеры, хранимые процедуры и алиасы. Владелец схемы (обычно индивид, создавший схему) автоматически получает эти привилегии вместе с правом предоставлять любую комбинацию этих привилегий другим пользователям или группам.

Привилегии табличного пространства



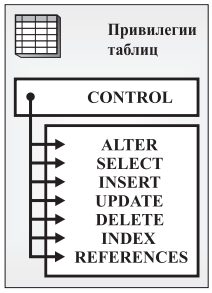
- CRANT USE OF TABLESPACE [*TablespaceName*] TO [*Recipient, …*] <WITH GRANT OPTION>
- REVOKE USE OF TABLESPACE [*TablespaceName*] FROM [*Forfeiter, …*] <BY ALL>

WCF03.0

CONTROL. Предоставляет пользователю каждую доступную привилегию табличного пространства, позволяет пользователю удалять (уничтожать) табличное пространство из базы данных и дает пользователю возможность предоставлять или отбирать у других пользователей или групп привилегию табличного пространства USE. (Предоставлять и отбирать привилегии CONTROL для объекта разрешается лишь пользователям, обладающим полномочиями SYSADM или DBADM).

USE. Позволяет пользователю создавать таблицы в табличном пространстве. (Эта привилегия используется для контроля того, в каких табличных пространствах разрешено определенному пользователю создавать таблицы).

Привилегии таблицы



- GRANT [ALL <PRIVILEGES> | *Privilege* <(*ColumnName, …*)>, …] ON TABLE [*TableName*] TO [*Recipient, …*] <WITH GRANT OPTION>
- REVOKE [ALL <PRIVILEGES> | *Privilege, …*] ON TABLE [*TableName*] FROM [*Forfeiter, …*] <BY ALL>

WCF03.0

CONTROL. Предоставляет пользователю каждую доступную привилегию, позволяет пользователю удалять (уничтожать) таблицу в базе данных и дает пользователю возможность предоставлять или отбирать у других пользователей и групп любые доступные привилегии таблиц (за исключением привилегии CONTROL).

ALTER. Позволяет пользователю выполнять для таблицы оператор SQL ALTER TABLE. Другими словами, позволяет пользователю добавлять в таблицу столбцы, добавлять или изменять комментарии, связанные с каждой таблицей и/или с ее столбцами, создавать для таблицы первичный ключ, создавать для таблицы уникальное ограничение, создавать или уничтожать для таблицы проверочное ограничение и создавать для таблицы триггеры (при условии, что у пользователя есть соответствующие привилегии для каждого объекта, на который ссылается триггер).

SELECT. Позволяет пользователю выполнять для таблицы оператор SQL SELECT. Другими словами, позволяет пользователю получать данные из таблицы, создавать производную таблицу, которая ссылается на таблицу и запускать для таблицы утилиту EXPORT.

INSERT. Позволяет пользователю выполнять для таблицы оператор SQL INSERT. Другими словами, позволяет пользователю добавлять данные в таблицу и запускать для таблицы утилиту IMPORT.

UPDATE. Позволяет пользователю выполнять для таблицы оператор SQL UPDATE. Другими словами, позволяет пользователю изменять данные в таблице. (Эта привилегия может быть предоставлена для всей таблицы или ограничиваться одной или несколькими столбцами внутри таблицы).

DELETE. Позволяет пользователю выполнять для таблицы оператор SQL DELETE. Другими словами, позволяет пользователю удалять из таблицы строки данных.

INDEX. Позволяет пользователю создавать индексы для таблицы.

REFERENCES. Позволяет пользователю создавать и уничтожать ограничения внешних ключей, которые ссылаются на таблицу в родительском отношении. (Эта привилегия может быть предоставлена для всей таблицы или ограничиваться одним или несколькими столбцами в таблице, в последнем случае в качестве родительского ключа в реляционном ограничении могут участвовать лишь эти столбцы).

Привилегии индекса

- GRANT CONTROL ON INDEX [*IndexName*] TO [*Recipient*]
- REVOKE CONTROL ON INDEX [*IndexName*] FROM [*Forfeiter, …*] <BY ALL>

Привилегии производной таблицы

- GRANT [ALL <PRIVILEGES> | *Privilege* <(*ColumnName*, …)>, …] ON [*ViewName*] TO [*Recipient*] <WITH GRANT OPTION>
- REVOKE [ALL <PRIVILEGES> | *Privilege*, …] ON [*ViewName*] FROM [*Forfeiter*, …] <BY ALL>

WCF03.0

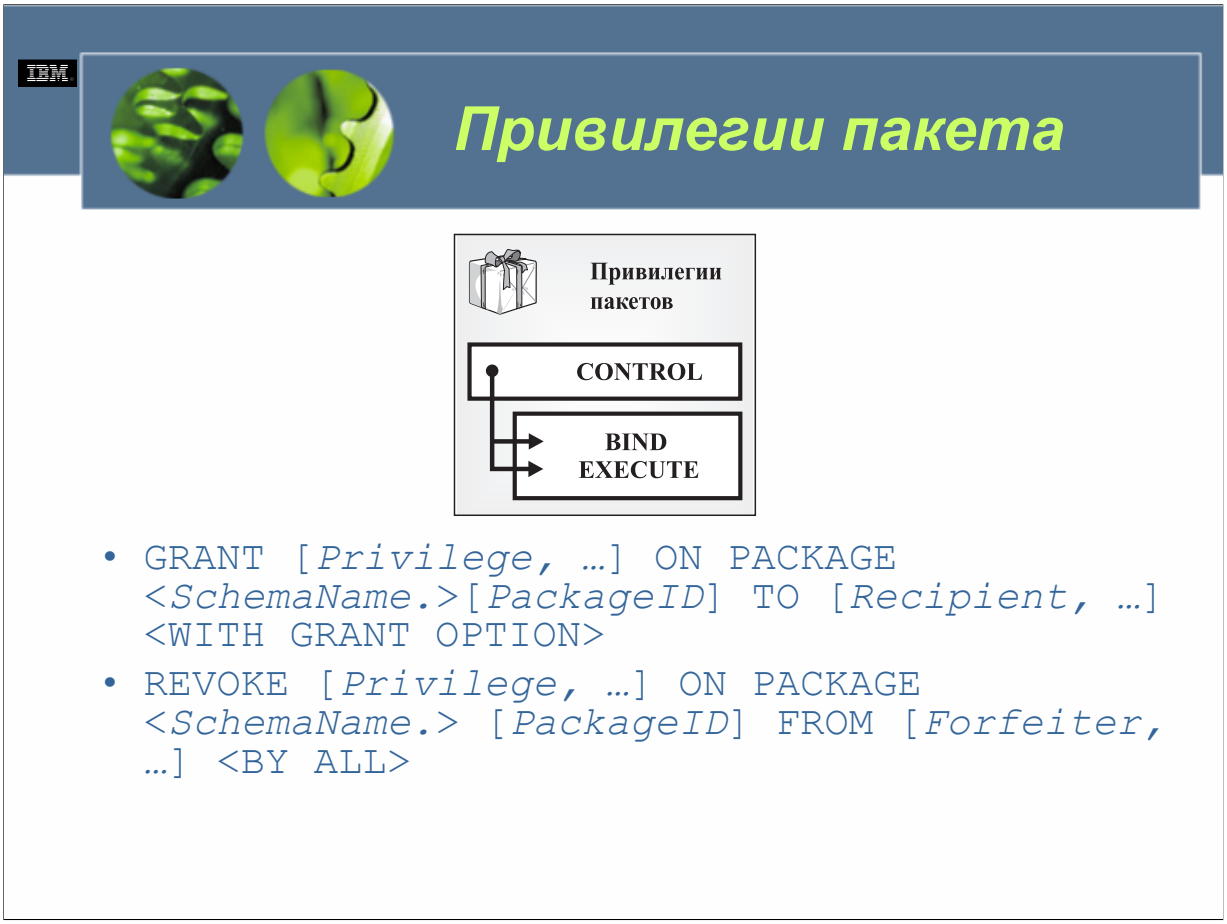
CONTROL. Предоставляет пользователю каждую доступную привилегию производных таблиц, позволяет пользователю удалять (уничтожать) производную таблицу в базе данных и дает пользователю возможность предоставлять или отнимать у других пользователей и групп любые доступные привилегии производных таблиц (за исключением привилегии CONTROL).

SELECT. Позволяет пользователю получать данные из производной таблицы, создавать другую производную таблицу, которая ссылается на первую, и запускать для производной таблицы утилиту EXPORT.

INSERT. Позволяет пользователю добавлять данные в производную таблицу.

UPDATE. Позволяет пользователю изменять данные в производной таблице. (Эта привилегия может быть предоставлена для всей производной таблицы или быть ограничена одной или несколькими столбцами в производной таблице).

DELETE. Позволяет пользователю удалять строки данных из производной таблицы.



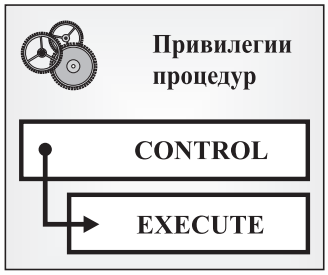
WCF03.0

CONTROL. Предоставляет пользователю каждую доступную привилегию пакета, позволяет пользователю удалять (уничтожать) пакет в базе данных и дает пользователю возможность предоставлять и отнимать у других пользователей и групп любые доступные привилегии пакета (за исключением привилегии CONTROL).

BIND. Позволяет пользователю повторно связывать или добавлять новые версии пакетов к пакету, который уже был связан с базой данных. (Кроме привилегии пакета BIND, пользователь должен обладать привилегиями для выполнения операторов SQL, составляющими пакет, прежде чем пакет может быть успешно связан повторно).

EXECUTE. Позволяет пользователю исполнить пакет. (Пользователь, у которого есть привилегия EXECUTE для определенного пакета, может исполнить этот пакет, даже если у него нет привилегий, необходимых для выполнения операторов SQL, содержащихся в пакете. Это потому, что любая привилегия, необходимая для выполнения операторов SQL в пакете, предоставляется пользователю пакета неявным образом. Важно отметить, что для неявного предоставления привилегий создатель пакета должен обладать привилегиями в качестве индивидуального пользователя или члена группы PUBLIC – а не члена другой именованной группы).

Привилегии процедур

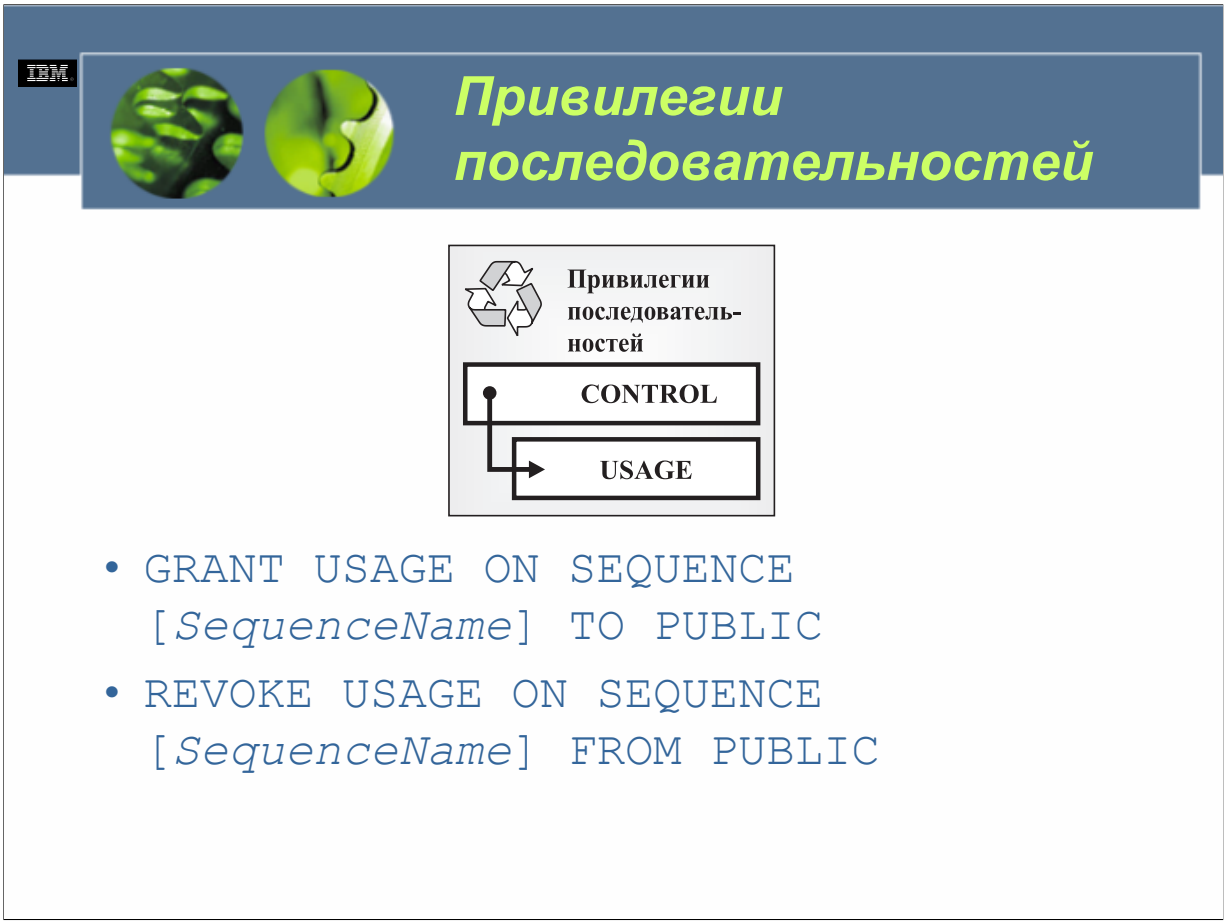


- GRANT EXECUTE ON [*RoutineName* | FUNCTION <*SchemaName.*> * | METHOD * FOR [*TypeName*] | METHOD * FOR <*SchemaName.*> * | PROCEDURE <*SchemaName.*> *] TO [*Recipient, …*] <WITH GRANT OPTION>
- REVOKE EXECUTE ON [*RoutineName* | FUNCTION <*SchemaName.*> * | METHOD * FOR [*TypeName*] | METHOD * FOR <*SchemaName.*> * | PROCEDURE <*SchemaName.*> *] FROM [*Forfeiter, …*] <BY ALL> RESTRICT

WCF03.0

CONTROL. Предоставляет пользователю каждую доступную привилегию процедуры, позволяет пользователю удалять (уничтожать) процедуру из базы данных и дает пользователю возможность предоставлять или отзывать у других пользователей и групп любые доступные привилегии процедуры (за исключением привилегии CONTROL).

EXECUTE. Позволяет пользователю вызывать процедуру, создавать функцию, которая происходит из этой процедуры (при условии, что процедура является функцией), и ссылаться на эту процедуру в операторе DDL или при создании ограничения.



WCF03.0

CONTROL. Предоставляет пользователю каждую доступную привилегию последовательности, позволяет пользователю удалять (уничтожать) последовательность из базы данных и дает пользователю возможность предоставлять или отзывать у других пользователей и групп любые доступные привилегии последовательности (за исключением привилегии CONTROL).

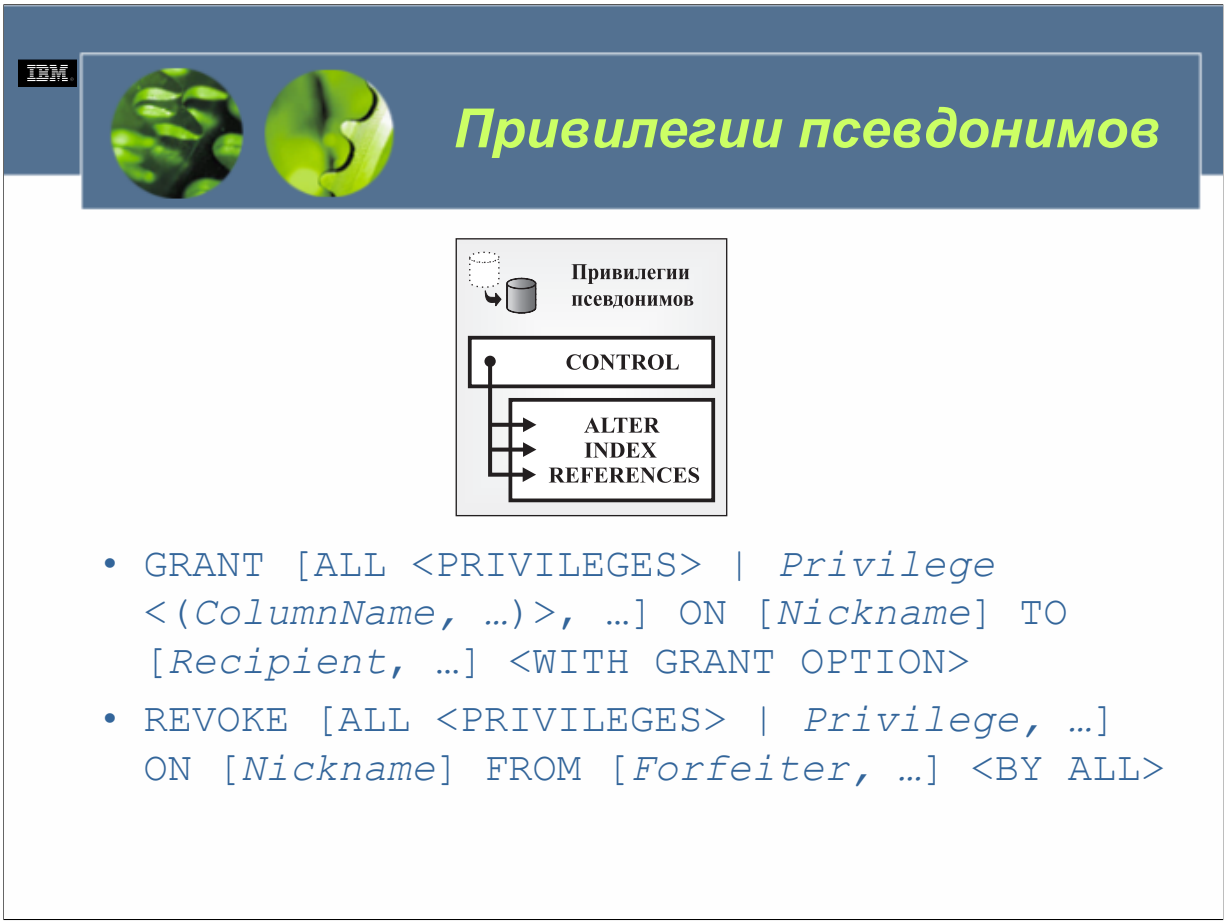
USAGE. Позволяет пользователю использовать выражения PREVVAL и NEXTVAL, связанные с последовательностью. (Выражение PREVVAL возвращает последнее сгенерированное значение для определенной последовательности; выражение NEXTVAL возвращает следующее значение для определенной последовательности).

Привилегии сервера

- GRANT PASSTHRU ON SERVER [*ServerName*] TO [*Recipient, …*]
- REVOKE PASSTHRU ON SERVER [*ServerName*] FROM [*Forfeiter, …*] <BY ALL>

WCF03.0

PASSTHRU позволяет пользователю выдавать операторы языка определения данных (DDL) и языка обработки данных (DML) SQL (в качестве проходных операций) непосредственно источнику данных через сервер объединения.



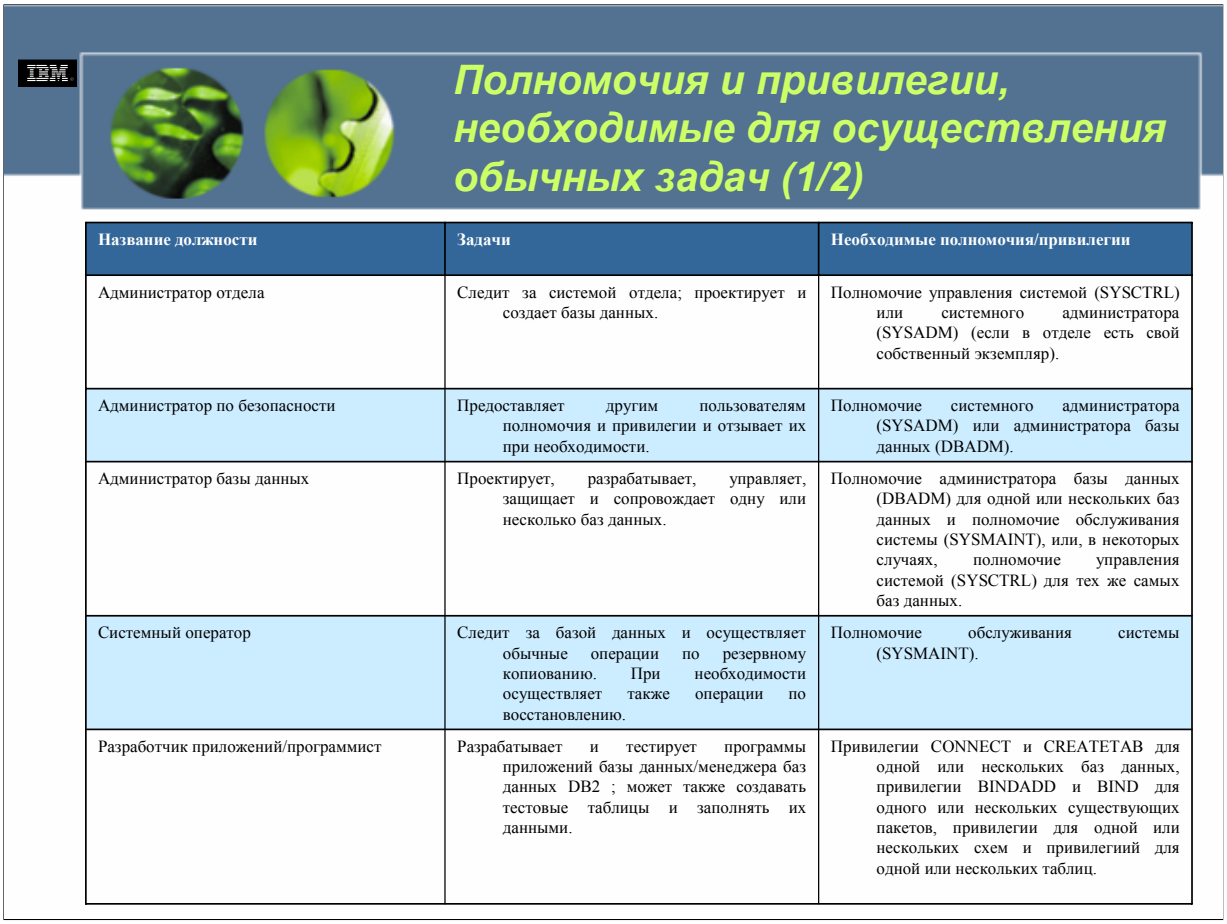
WCF03.0

CONTROL. Предоставляет пользователю каждую доступную привилегию псевдонима, позволяет пользователю удалять (уничтожать) псевдоним в базе данных и дает пользователю возможность предоставлять или отзывать у других пользователей и групп любые доступные привилегии псевдонимов (за исключением привилегии CONTROL).

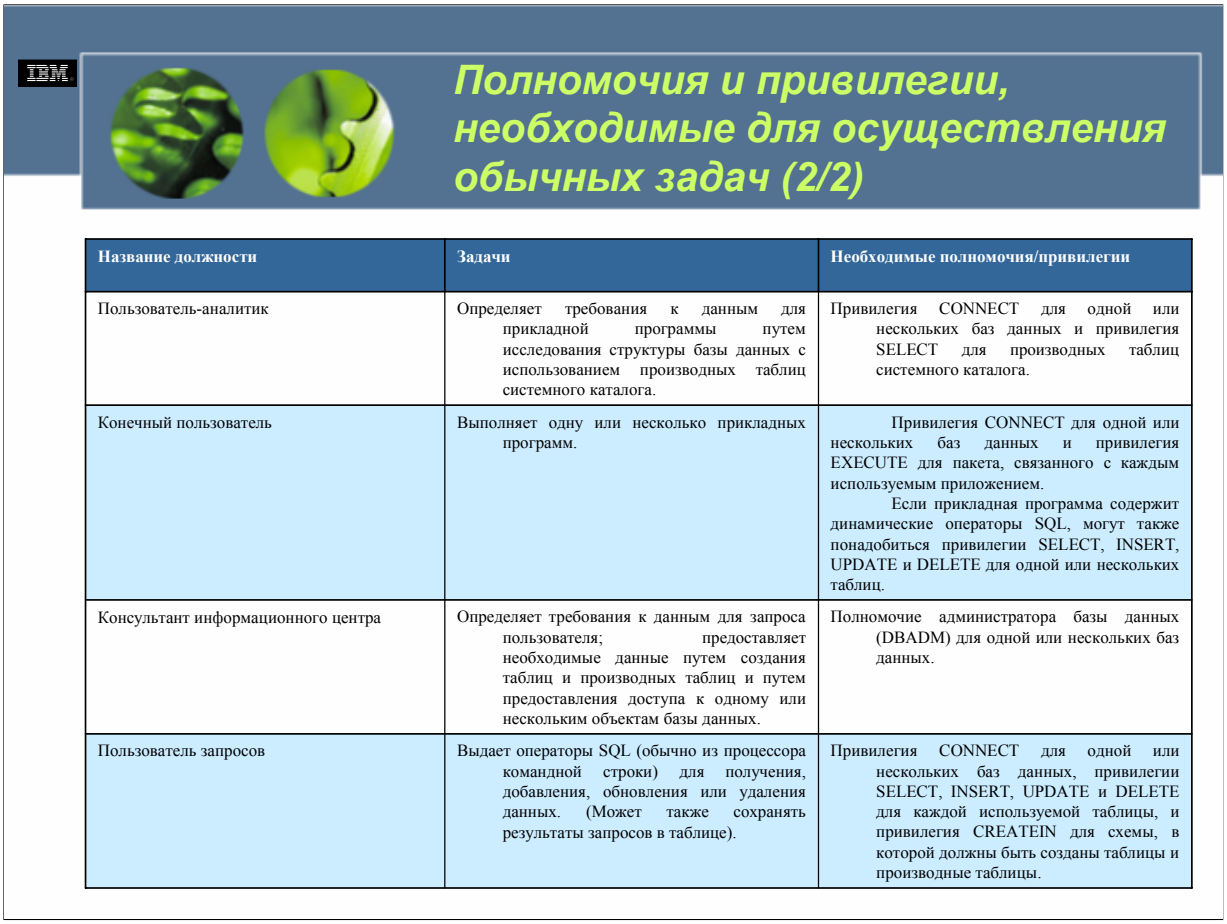
ALTER. Позволяет пользователю изменять имена столбцов в псевдониме, добавлять или изменять типы данных DB2, на который отображается тип данных определенного столбца псевдонима, и указывать опции столбцов для определенного столбца псевдонима.

INDEX. Позволяет пользователю создавать спецификацию индекса для псевдонима.

REFERENCES. Позволяет пользователю создавать и уничтожать ограничения внешнего ключа, который ссылается на псевдоним в родительском отношении. (Эта привилегия может быть предоставлена для всего псевдонима или ограничиваться одним или несколькими столбцами в псевдониме).



До сих пор мы установили доступные полномочия и привилегии и рассмотрели, как эти полномочия и привилегии предоставляются и отзываются. Но для эффективного использования полномочий и привилегий вы должны иметь возможность определять, какие полномочия и привилегии подходят для отдельного пользователя, а какие нет. Часто индивиду назначается полный набор полномочий и привилегий на основе названия его должности и/или служебных обязанностей. Затем, когда все индивиды начинают работать с базой данных, набор полномочий и привилегий изменяется нужным образом. Некоторые из наиболее типичных названий должностей вместе с задачами, которые часто их сопровождают, и полномочиями/привилегиями, необходимыми для выполнения этих задач, можно видеть в таблице.

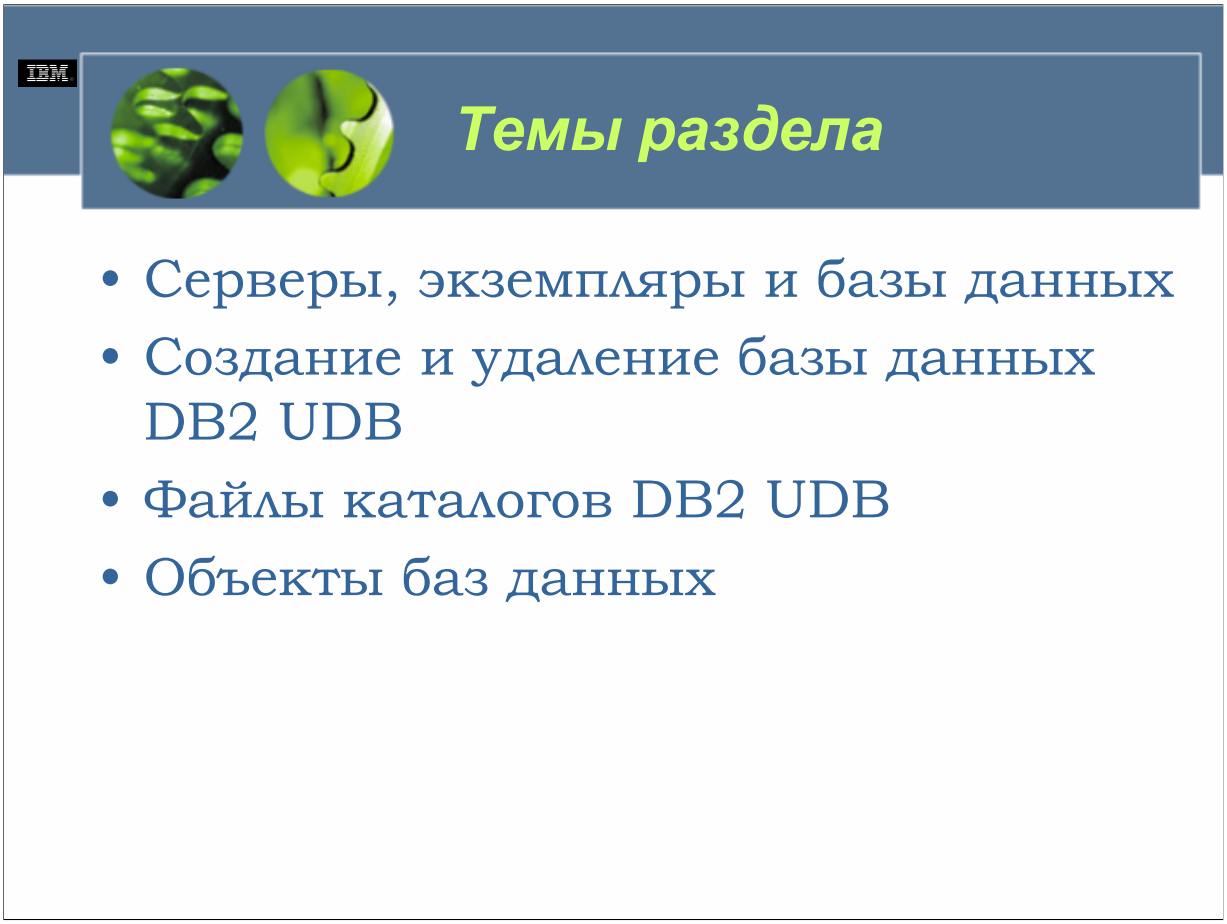


Название должности	Задачи	Необходимые полномочия/привилегии
Пользователь-аналитик	Определяет требования к данным для прикладной программы путем исследования структуры базы данных с использованием производных таблиц системного каталога.	Привилегия CONNECT для одной или нескольких баз данных и привилегия SELECT для производных таблиц системного каталога.
Конечный пользователь	Выполняет одну или несколько прикладных программ.	Привилегия CONNECT для одной или нескольких баз данных и привилегия EXECUTE для пакета, связанного с каждым используемым приложением. Если прикладная программа содержит динамические операторы SQL, могут также понадобиться привилегии SELECT, INSERT, UPDATE и DELETE для одной или нескольких таблиц.
Консультант информационного центра	Определяет требования к данным для запроса пользователя; предоставляет необходимые данные путем создания таблиц и производных таблиц и путем предоставления доступа к одному или нескольким объектам базы данных.	Полномочие администратора базы данных (DBADM) для одной или нескольких баз данных.
Пользователь запросов	Выдает операторы SQL (обычно из процессора командной строки) для получения, добавления, обновления или удаления данных. (Может также сохранять результаты запросов в таблице).	Привилегия CONNECT для одной или нескольких баз данных, привилегии SELECT, INSERT, UPDATE и DELETE для каждой используемой таблицы, и привилегия CREATEIN для схемы, в которой должны быть созданы таблицы и производные таблицы.

WCF03.0

До сих пор мы установили доступные полномочия и привилегии и рассмотрели, как эти полномочия и привилегии предоставляются и отзываются. Но для эффективного использования полномочий и привилегий вы должны иметь возможность определять, какие полномочия и привилегии подходят для отдельного пользователя, а какие нет. Часто индивиду назначается полный набор полномочий и привилегий на основе названия его должности и/или служебных обязанностей. Затем, когда все индивиды начинают работать с базой данных, набор полномочий и привилегий изменяется нужным образом. Некоторые из наиболее типичных названий должностей вместе с задачами, которые часто их сопровождают, и полномочиями/привилегиями, необходимыми для выполнения этих задач, можно видеть в таблице.

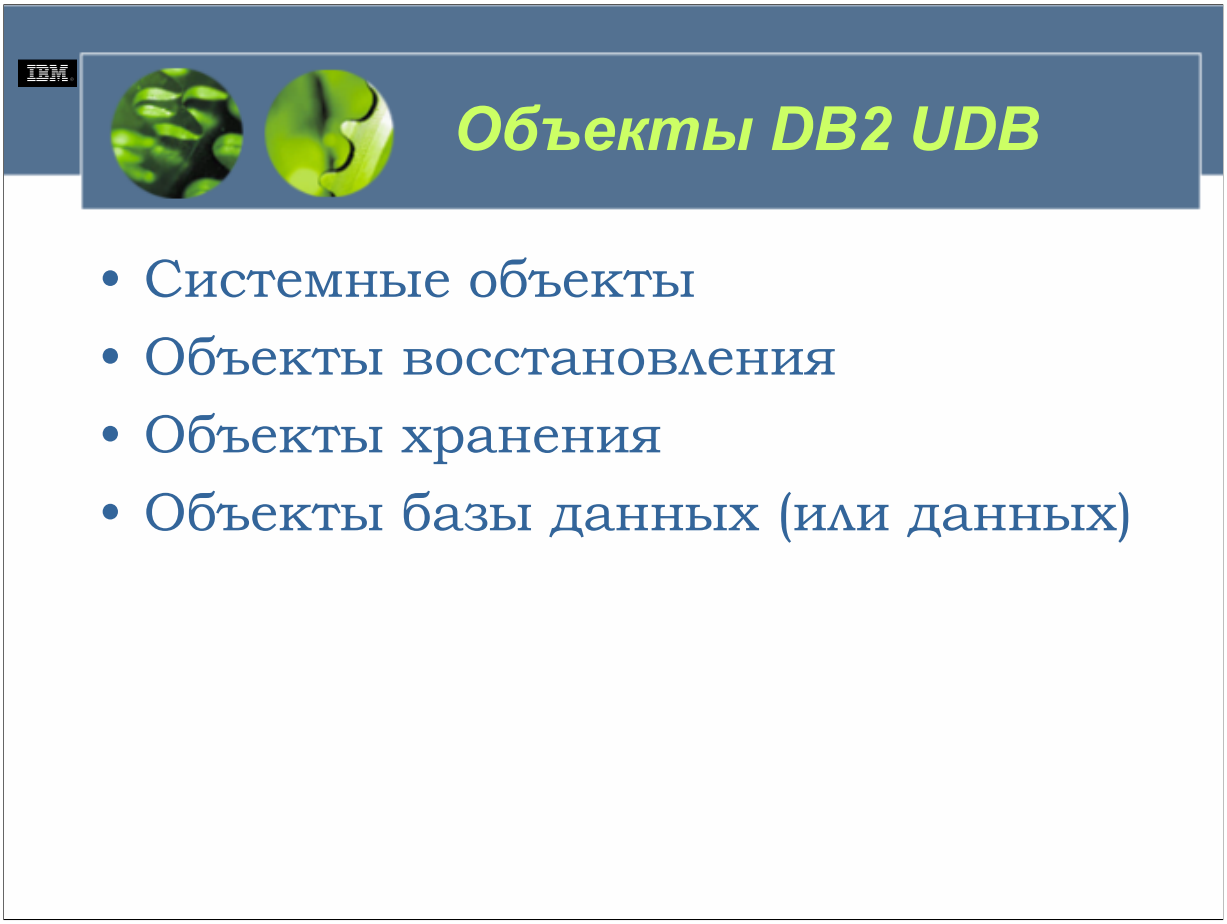
WCF03.0

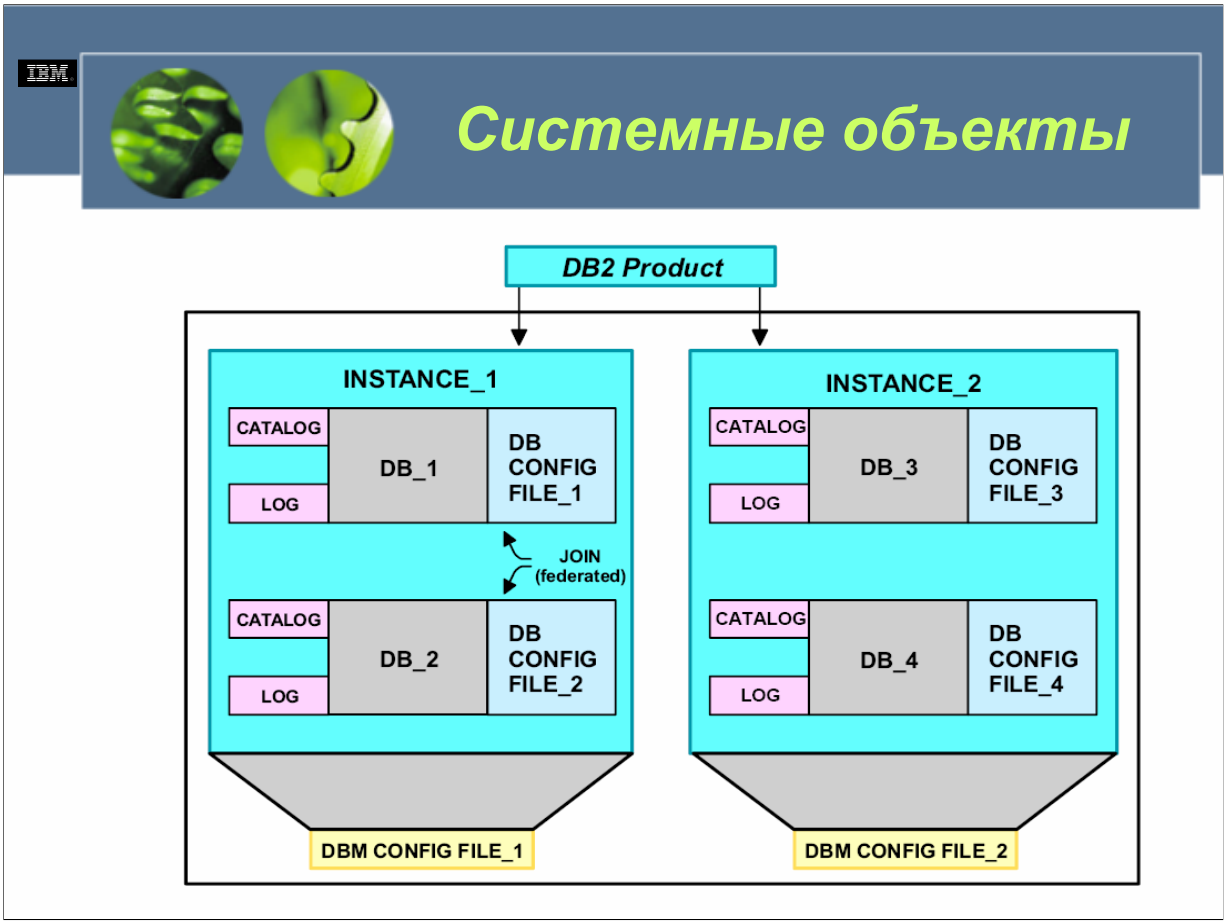


WCF03.0

Универсальная база данных DB2 видит окружение в виде иерархии объектов. Рабочие станции (или серверы), на которых была установлена DB2 UDB, занимает высший уровень иерархии. В ходе процесса инсталляции программные файлы для фонового процесса, известного как менеджер баз данных DB2 (*DB2 Database Manager)*, физически копируются в определенное место на сервере, и создается *экземпляр* менеджера баз данных DB2 . Экземпляры занимают второй уровень в иерархии и отвечают за управление системными ресурсами и базами данных, которые подпадают под их контроль. Хотя первоначально создается лишь один экземпляр, могут существовать несколько экземпляров. Каждый экземпляр ведет себя, как отдельная инсталляция DB2 UDB, даже если все экземпляры в пределах системы разделяют одни и те же программные файлы менеджера баз данных DB2 (которые были скопированы на рабочую станцию в ходе процесса инсталляции). И хотя несколько экземпляров совместно используют один и тот же двоичный код, каждый работает независимо от других и имеет свое собственное окружение (которое можно изменять, изменяя содержимое связанных с ними конфигурационных файлов).

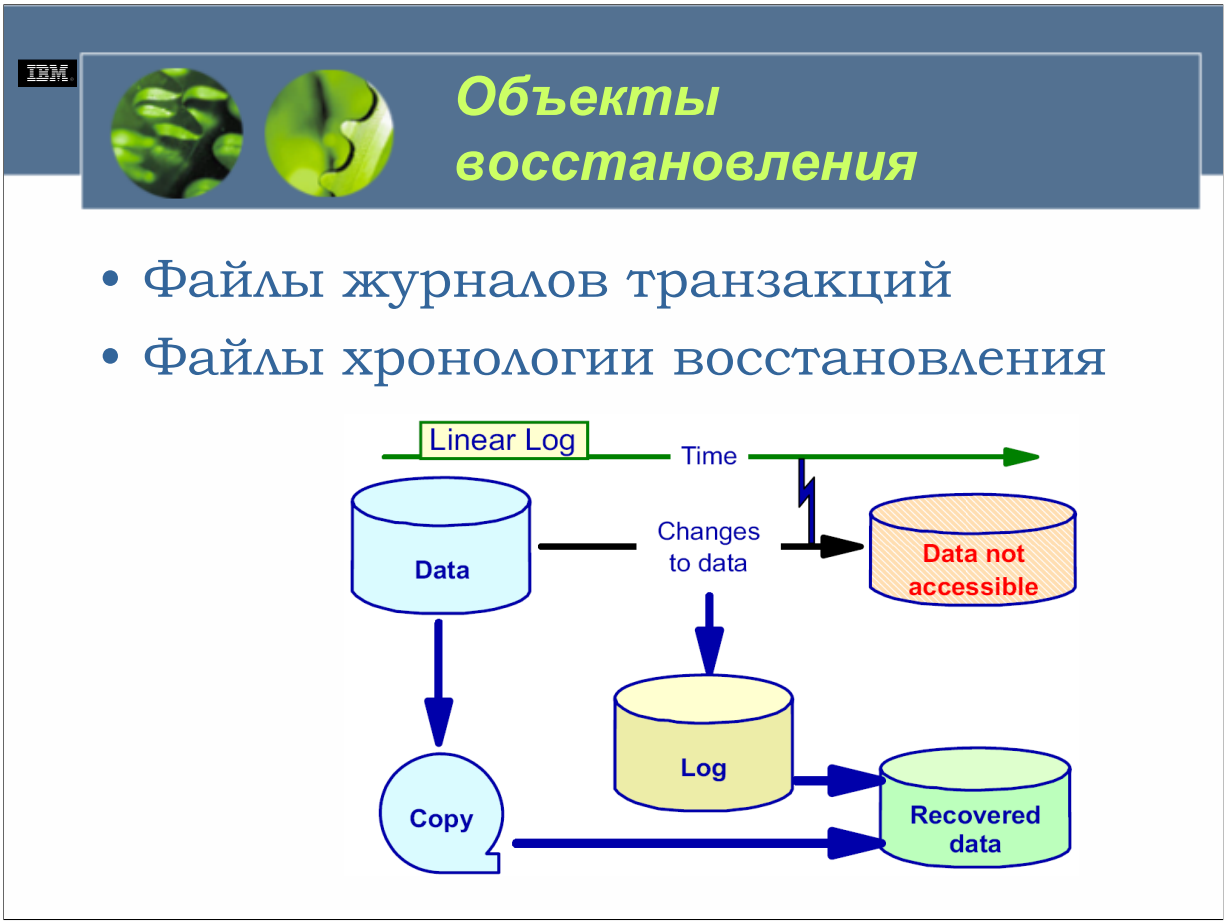
Каждый экземпляр контролирует доступ к одной или нескольким базам данных. Базы данных составляют третий уровень в иерархии и отвечают за управление хранением, изменение и получение данных. Аналогично экземплярам, базы данных работают независимо друг от друга. У каждой базы данных есть свое собственное окружение (также контролируемое конфигурационным файлом), а также свой собственный набор предоставляемых полномочий и привилегий для управления тем, как пользователи взаимодействуют с данными и объектами базы данных, которые она контролирует.





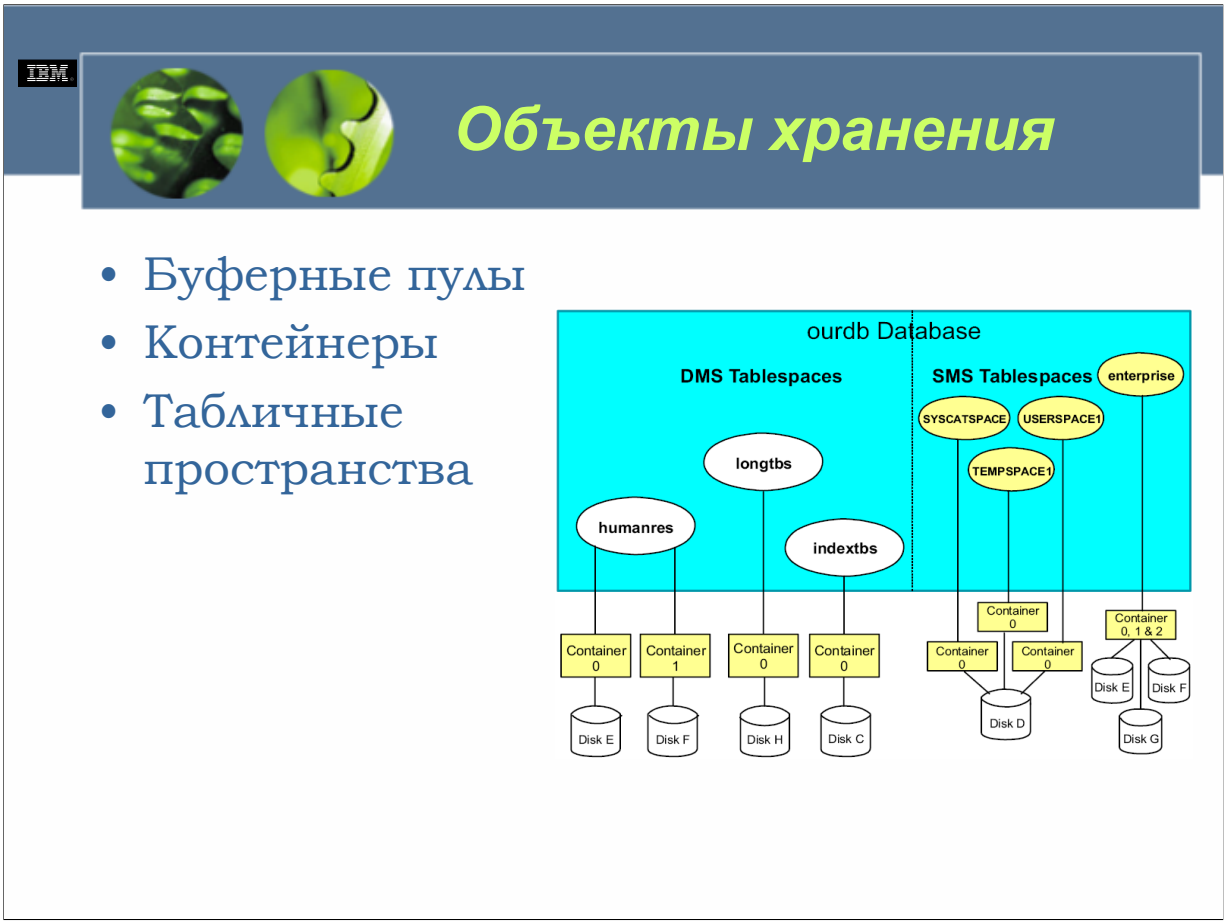
WCF03.0

Системные объекты состоят из переменных реестра, конфигурационных файлов экземпляров и конфигурационных файлов индивидуальных баз данных. Переменные реестра устанавливаются на уровне системы и влияют на каждый экземпляр, находящийся на определенном сервере. Конфигурационные файлы экземпляров (известные также как конфигурационные файлы менеджера баз данных DB2) создаются и назначаются отдельным экземплярам в ходе процесса создания экземпляра. Значения в конфигурационном файле экземпляра контролируют то, как выделяются ресурсы для данного отдельного экземпляра, а их изменения влияют на каждую базу данных, которая попадает под контроль данного экземпляра. (Значения для многих из параметров в конфигурационном файле экземпляра могут изменяться для повышения общей производительности или параллелизма). Конфигурационные файлы баз данных создаются и назначаются отдельным базам данных в ходе процесса создания базы данных. Значения в конфигурационном файле базы данных контролируют то, как выделяются ресурсы для данной отдельной базы данных, а изменения в них могут повысить производительность или вместимость, в зависимости от типа деятельности, с которой сталкивается база данных.



WCF03.0

Объекты восстановления состоят из файлов журналов транзакций и файлов хронологии восстановления. По умолчанию при создании базы данных автоматически создаются один файл хронологии восстановления и три файла журнала транзакций. Файлы хронологии восстановления используются совместно с резервными копиями базы данных и файлами журнала транзакций для координации операций восстановления базы данных. Файл хронологии восстановления содержит информацию о каждой выполненной операции резервного копирования, тогда как файлы журнала транзакций содержат записи недавно выполненных операций базы данных. В том случае, если база данных должна быть восстановлена из-за ошибки приложения, пользователя или системы, события, сохраненные в файлах журнала транзакций, проигрываются заново для возвращения базы данных в согласованное (согласованность базы данных подробно описывается в главе 7) и стабильное состояние или для возвращения базы данных в то состояние, в котором она находилась в момент времени, когда возникла ошибка, если включено восстановление с повтором транзакций. Вы не можете непосредственно изменять файлы журнала транзакций или файлы хронологии восстановления; однако вы обнаружите, что их содержание важно, когда нужно починить базу данных, которая была разрушена или повреждена.

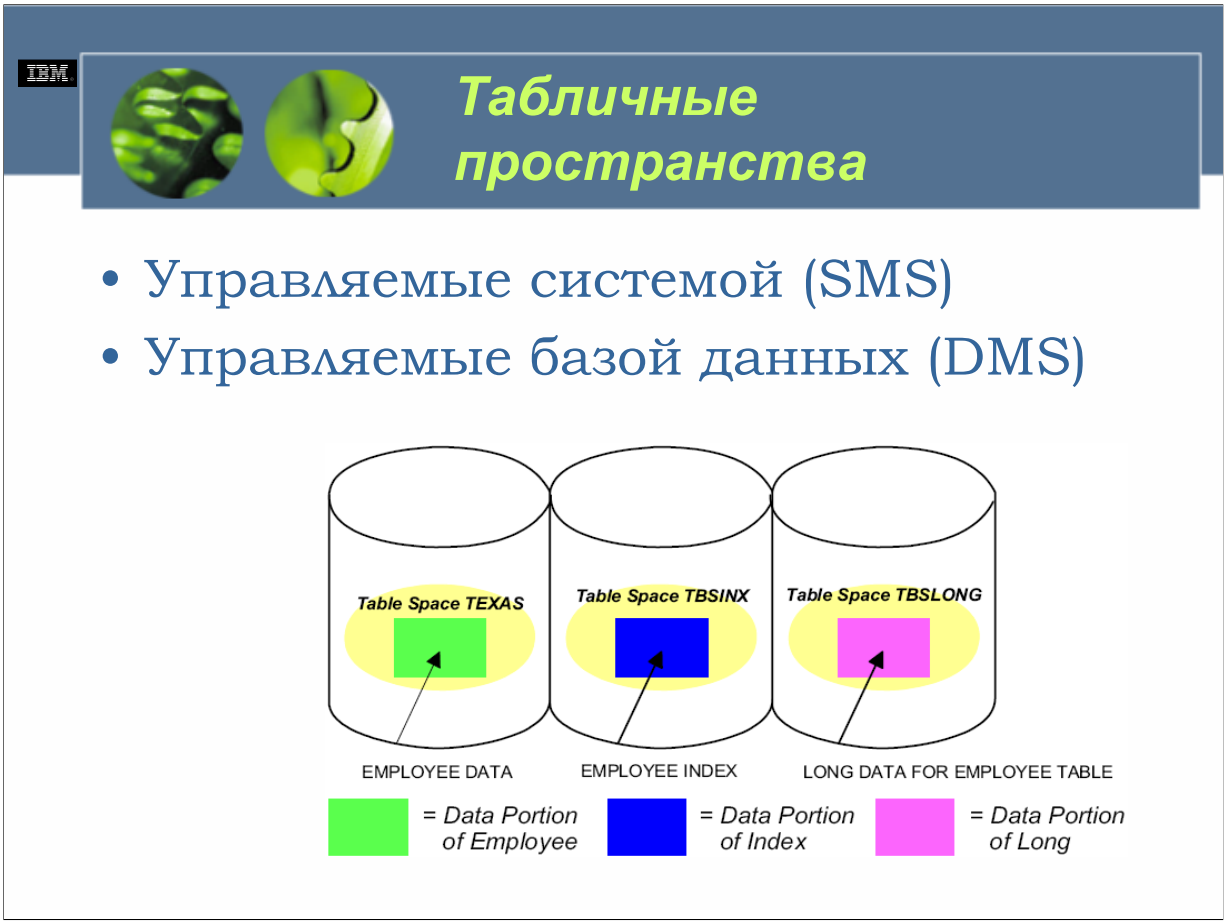


WCF03.0

Буферные пулы. Буферный пул является разделом памяти, который был зарезервирован с единственной целью кэширования страниц данных по мере их чтения из физического хранилища. Каждый раз, когда для разрешения запроса нужны данные, в физическом хранилище находится страница, на которой сохранены данные (данные хранятся в разделах, называемых *страницами*), и перемещается в буферный пул, где она затем считывается и/или изменяется. Если страница изменена, она копируется обратно в физическое хранилище; однако все прочтенные страницы остаются в памяти до тех пор, пока не понадобится занимаемое ими пространство или пока не будут завершены все соединения с базой данных. Более того, каждый раз при возвращении страницы менеджер баз данных DB2 использует набор эвристических алгоритмов, чтобы попытаться определить, какие страницы понадобятся в следующий раз – эти страницы также возвращаются (это называется *предварительной выборкой (prefetch)*). Сохранение всех загруженных страниц и предварительная выборка выполняются для повышения общей производительности; доступ к данным осуществляется гораздо быстрее, когда они хранятся в памяти, чем когда они хранятся на диске.

Контейнеры. Контейнер представляет собой разновидность физического хранилища, к которому менеджер баз данных DB2 зарезервировал доступ. Контейнер может быть каталогом, который может уже существовать и может не существовать, или физическим устройством, которое распознается операционной системой. (В операционных системах Linux и UNIX физическим устройством может быть любой логический том, который использует специальный символьный интерфейс; в операционных системах Windows физическим устройством является любой неформатированный раздел любого физического диска).

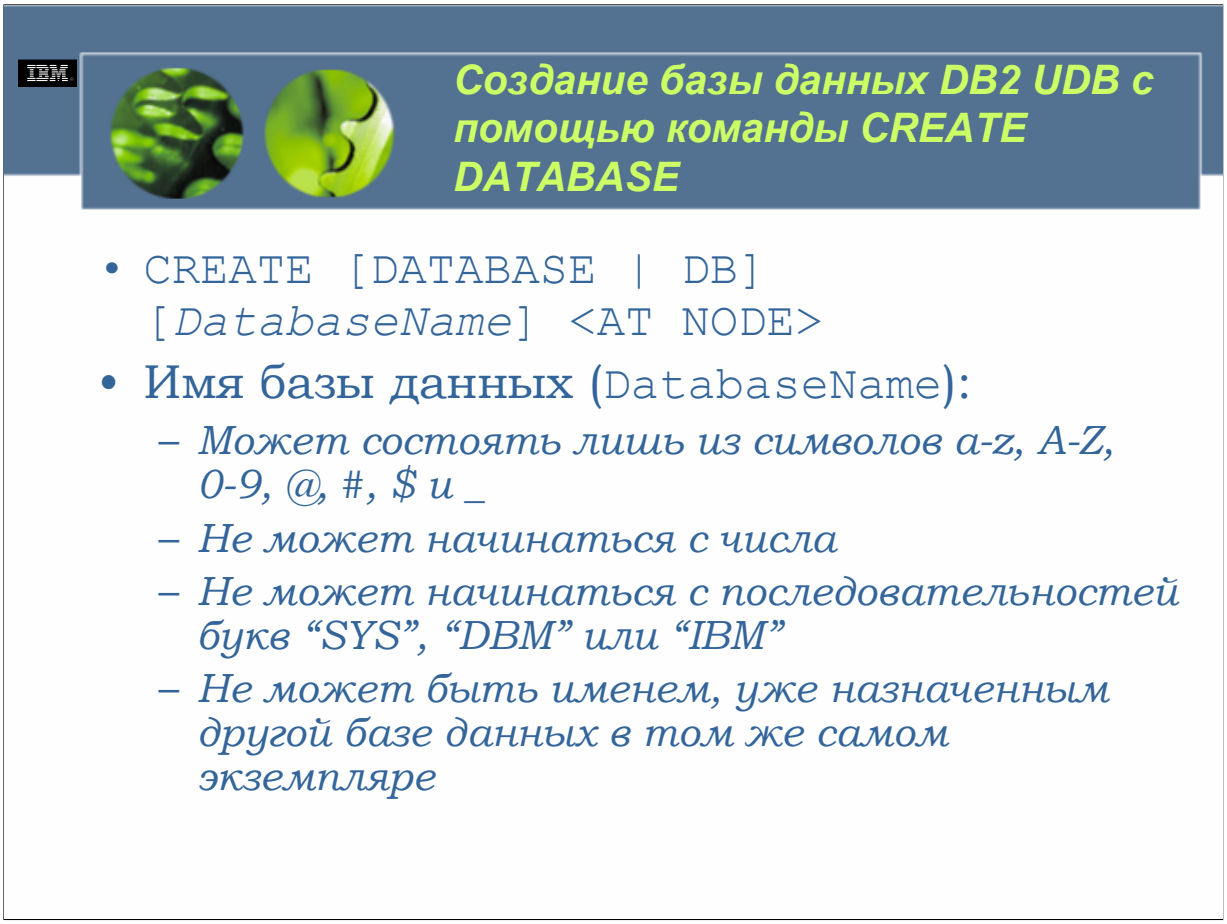
Табличные пространства. Табличные пространства используются для контролирования того, где физически хранятся данные, и для предоставления промежуточного уровня между таблицей и одним или несколькими контейнерами, в которых реально находятся данные таблицы. Единственное табличное пространство может охватывать множество контейнеров, но каждый контейнер может принадлежать лишь одном табличному пространству. Когда табличное пространство охватывает несколько контейнеров, данные записываются в каждый назначенный этому табличному пространству контейнер на манер карусели (в группах страниц, называемых *экстентами (extent)*); это помогает сбалансированно распределять данные между всеми контейнерами, которые принадлежат данному табличному пространству.



WCF03.0

Существует два типа табличных пространств: табличные пространства, управляемые системой (SMS), и табличные пространства, управляемые базой данных (DMS). В табличных пространствах SMS для хранилища могут использоваться лишь контейнеры каталогов, а менеджер файлов операционной системы отвечает за контролирование того, как используется пространство. В табличных пространствах DMS для хранилища могут использоваться только контейнеры файлов и/или устройств, а за контролирование того, как используется пространство, отвечает менеджер баз данных DB2 .

Если вы близко рассмотрите то, как DB2 UDB управляет данными, вы обнаружите, что в зависимости от того, как была определена таблица, каждая соответствующая запись может хранится в виде трех различных значений: в качестве обычного значения данных, как хранятся данные, создаваемые приложением, пользователем или триггером; в качестве значения индекса, как хранятся все данные, относящиеся к индексу; и в качестве одного или нескольких значений длинных данных, как хранятся значения длинных данных и больших объектов (LOB). Соответственно могут существовать три типа табличных пространств: обычное, длинных данных и временное. Как можно было представить, обычные данные и данные индекса могут находиться в обычных табличных пространствах, тогда как длинные данные и данные больших объектов могут находиться в табличных пространствах длинных данных. Временные табличные пространства, с другой стороны, используются для множества различных целей. Временные табличные пространства делятся на системные и пользовательские – системные временные табличные пространства используются для хранения внутренних временных данных, созданных при выполнении некоторых типов операций (например, сортировки данных, реорганизации таблиц, создания индексов и объединения таблиц), тогда как пользовательские временные табличные пространства используются для хранения объявленных глобальных временных таблиц, которые, в свою очередь, используются для хранения специфических данных приложений в течении короткого периода времени.



WCF03.0

Единственным значением, которое нужно предоставить при выполнении данного вида команды CREATE DATABASE, является имя для назначения создаваемой базе данных. Это имя:

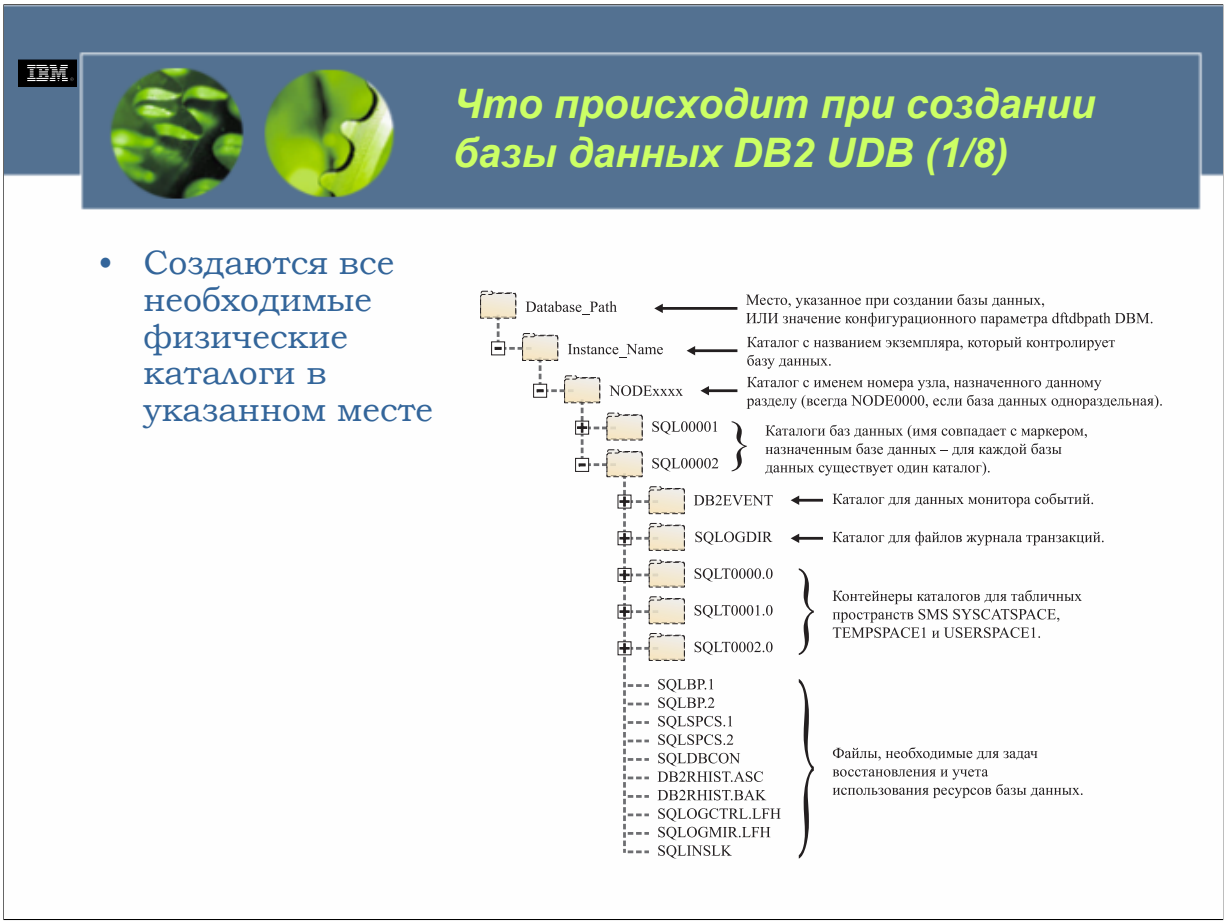
Может состоять лишь из символов a-z, A-Z, 0-9, @, #, $ и _.

Не может начинаться с числа.

Не может начинаться с последовательностей букв “SYS”, “DBM” или “IBM”.

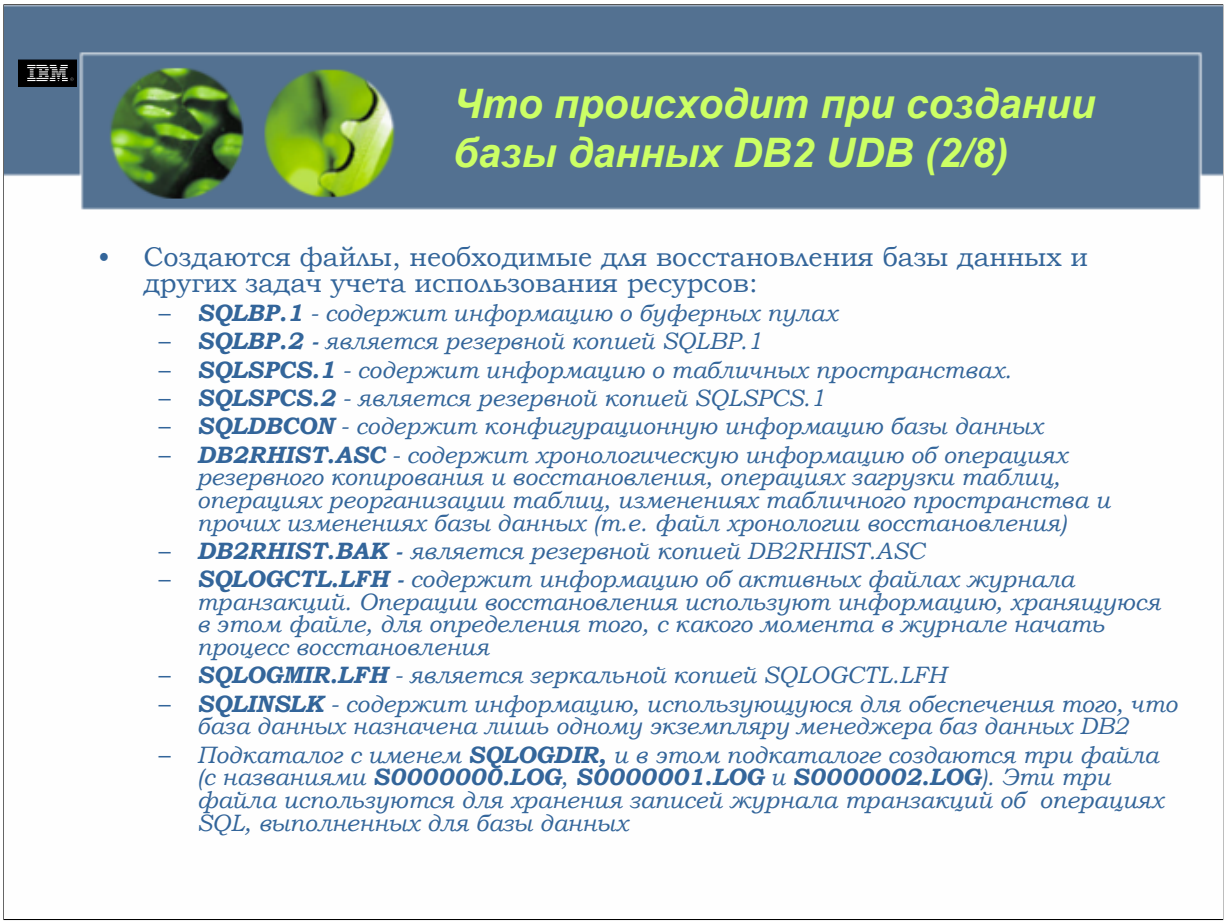
Не может быть именем, уже назначенным другой базе данных в том же самом экземпляре.

Конечно, доступна значительно более сложная форма команды CREATE DATABASE, предоставляющая вам значительно больший контроль над параметрами базы данных, и мы вскоре ее рассмотрим.



WCF03.0

Информация о каждой созданной базе данных сохраняется в специальном иерархическом дереве каталогов. Где в действительности создается это дерево каталогов, определяется информацией, предоставляемой командой CREATE DATABASE – если информация о местоположении не предоставлена, это дерево каталогов создается в месте, указанном в конфигурационном параметре *dftdbpath* менеджера баз данных DB2 , связанном с экземпляром, под которым создается база данных. Корневому каталогу этого иерархического дерева назначается имя экземпляра, с которым связана база данных. Этот каталог будет содержать подкаталог, которому назначено имя, соответствующее узлу раздела. Если база данных является многораздельной, этот каталог будет назван NODE*xxxx*, где *xxxx* является уникальным номером узла, который был назначен разделу; если база данных не является многораздельной, этот каталог будет назван NODE0000. Каталог с именем узла, в свою очередь, будет содержать по одному подкаталогу для каждой базы данных, которая была создана под этим узлом – имя, назначаемое каждому подкаталогу, соответствует маркеру базы данных, который был назначен базе данных (подкаталог для первой созданной базы данных будет назван SQL00001, подкаталог для второй базы данных буден назван SQL00002 и т.д.)



WCF03.0

После создания соответствующего каталога с именем базы данных, в этом каталоге создаются следующие файлы:

SQLBP.1. Этот файл содержит информацию о буферных пулах.

SQLBP.2. Этот файл является резервной копией SQLBP.1.

SQLSPCS.1. Этот файл содержит информацию о табличных пространствах.

SQLSPCS.2. Этот файл является резервной копией SQLSPCS.1.

SQLDBCON. Этот файл содержит конфигурационную информацию базы данных.

DB2RHIST.ASC. Этот файл содержит хронологическую информацию об операциях резервного копирования и восстановления, операциях загрузки таблиц, операциях реорганизации таблиц, изменениях табличного пространства и прочих изменениях базы данных (т.е. файл хронологии восстановления).

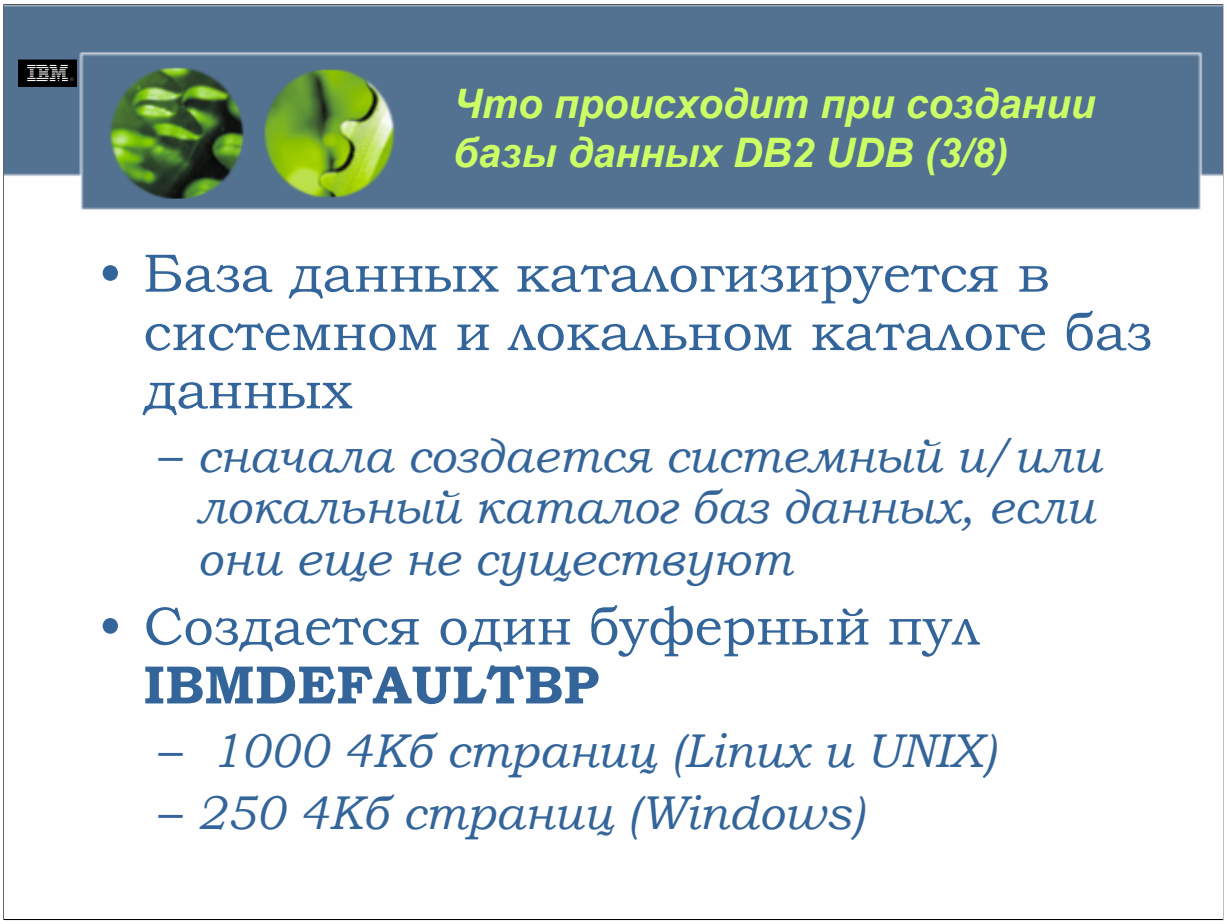
DB2RHIST.BAK. Этот файл является резервной копией DB2RHIST.ASC.

SQLOGCTL.LFH. Этот файл содержит информацию об активных файлах журнала транзакций. Операции восстановления используют информацию, хранящуюся в этом файле, для определения того, с какого момента в журнале начать процесс восстановления.

SQLOGMIR.LFH. Этот файл является зеркальной копией SQLOGCTL.LFH.

SQLINSLK. Этот файл содержит информацию, использующуюся для обеспечения того, что база данных назначена лишь одному экземпляру менеджера баз данных DB2 .

Создается также подкаталог с именем SQLOGDIR, и в этом подкаталоге создаются три файла (с названиями S0000000.LOG, S0000001.LOG и S0000002.LOG). Эти три файла используются для хранения записей журнала транзакций об операциях SQL, выполненных для базы данных.



WCF03.0

Для отслеживания того, где хранятся базы данных, и предоставления доступа к этим базам данных DB2 Universal Database использует набор специальных файлов. Поскольку информация, хранящаяся в этих файлах, используется аналогично информации, хранящейся в используемом каталоге товаров фирмы, они называются файлами каталогов. Каждый раз при создании базы данных в эти каталоги заносится имя базы данных и алиаса. (Мы более близко взглянем на доступные файлы каталога, когда будем рассматривать каталогизацию баз данных). В этих каталогах сохраняются также комментарии и значения кодового набора, если они указаны.

В ходе создания базы данных создается буферный пул с именем IBMDEFAULTBP. На платформах Linux и UNIX размер этого буферного пула составляет 1000 4Кб (килобайтных) страниц; на платформах Windows размер этого буферного пула равен 250 4Кб страниц. Реальная память, используемая этим буферным пулом (и в этом отношении, любыми другими буферными пулами, которые могут существовать), выделяется при установлении первого соединения с базой данных, а освобождается, когда все соединения с базой данных завершены.

WCF03.0

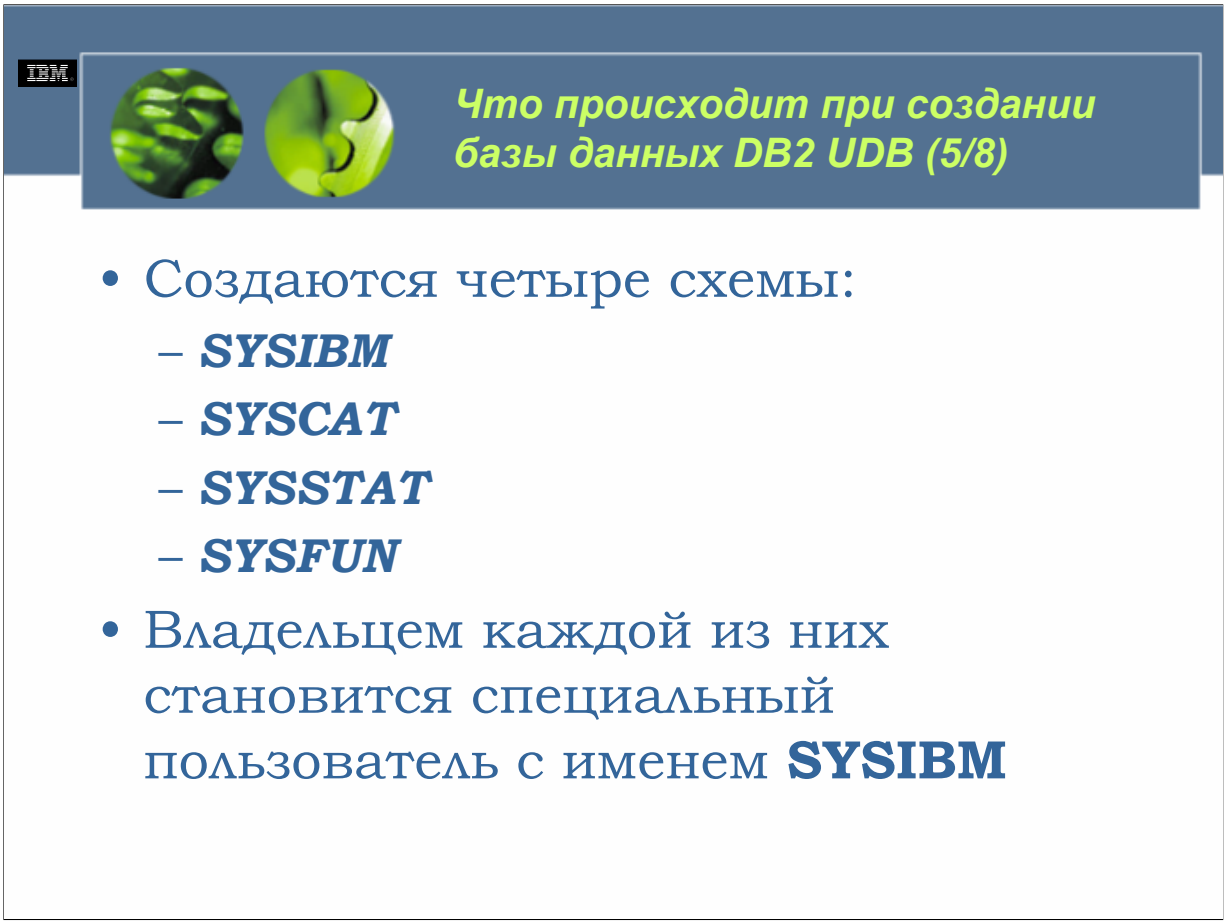
После создания буферного пула IBMDEFAULTBP создаются три табличных пространства, связанных с этим буферным пулом. Этими табличными пространствами являются:

Обычное табличное пространство с именем SYSCATSPACE1, которое используется для хранения таблиц системного каталога и производных таблиц (которые мы рассмотрим далее), связанных с базой данных.

Обычное табличное пространство с именем USERSPACE1, которое используется для хранения всех определенных пользователем объектов (таких как таблицы, индексы и т.д.) вместе с данными пользователя, данными индексов и данными длинных значений.

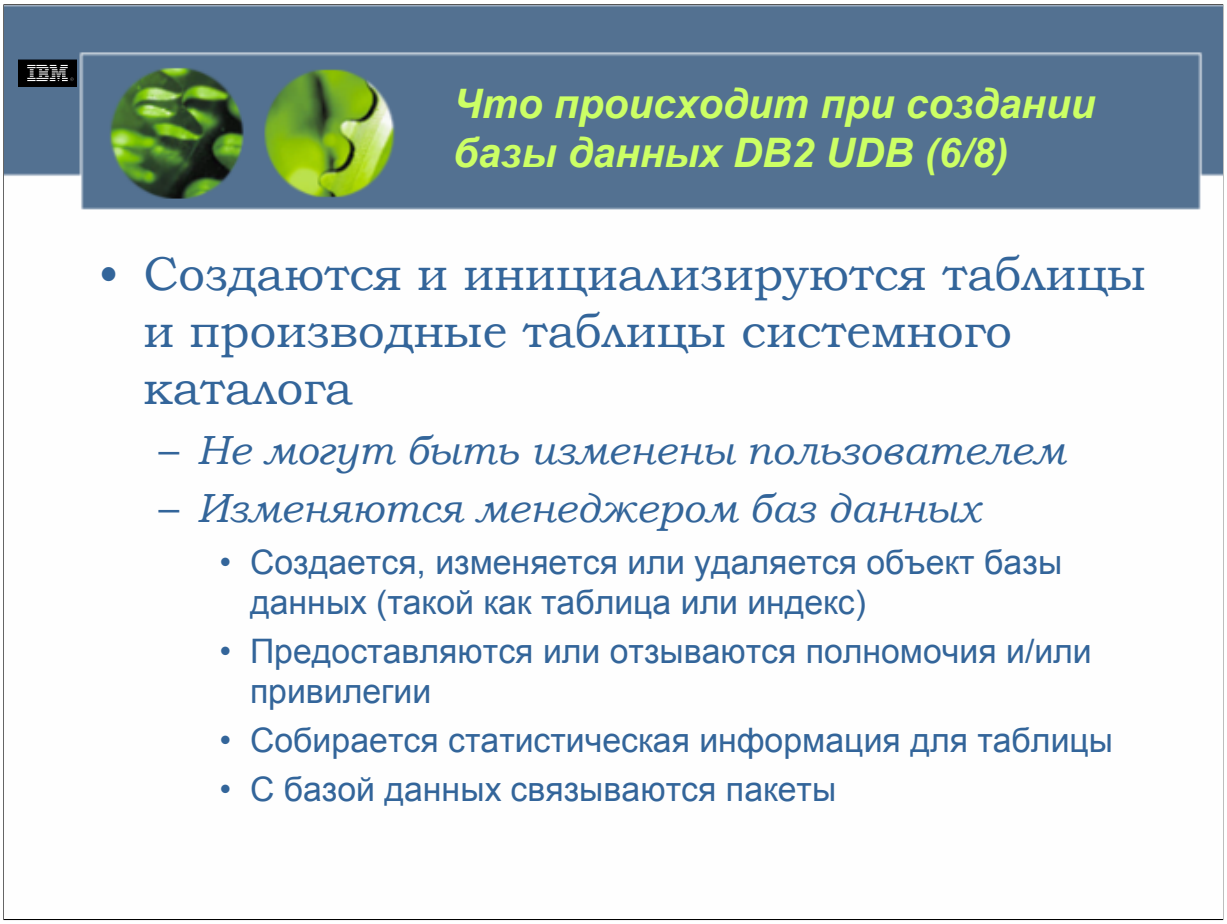
Системное временное табличное пространство с именем TEMPSPACE1, которое используется в качестве области временного хранения для таких операций, как сортировка данных, реорганизация таблиц и создание индексов.

Если не указано другое , все эти три табличных пространства будут табличными пространствами SMS с размером экстента в 32 4Кб страницы; свойства для каждого из этих табличных пространств можно предоставить в качестве входных данных мастера создания базы данных или команды CREATE DATABASE.



WCF03.0

В ходе процесса создания базы данных создаются следующие схемы: SYSIBM, SYSCAT, SYSSTAT и SYSFUN. Владельцем каждой из них становится специальный пользователь с именем SYSIBM.



WCF03.0

После создания табличного пространства SYSCATSPACE в рамках этого табличного пространства конструируется и заполняется специальный набор таблиц, известных как *таблицы системного каталога*. Менеджер баз данных DB2 использует эти таблицы системного каталога для отслеживания такой информации, как определения объектов базы данных, зависимости объектов базы данных, привилегии объектов базы данных, типы данных столбцов и ограничения таблиц. Вместе с таблицами системного каталога создается набор производных таблиц системного каталога, и эти производные таблицы обычно используются при доступе к данным, хранящимся в таблицах системного каталога. Таблицы и производные таблицы системного каталога не могут быть изменены с помощью операторов SQL. Вместо этого они изменяются менеджером баз данных DB2 всякий раз, когда:

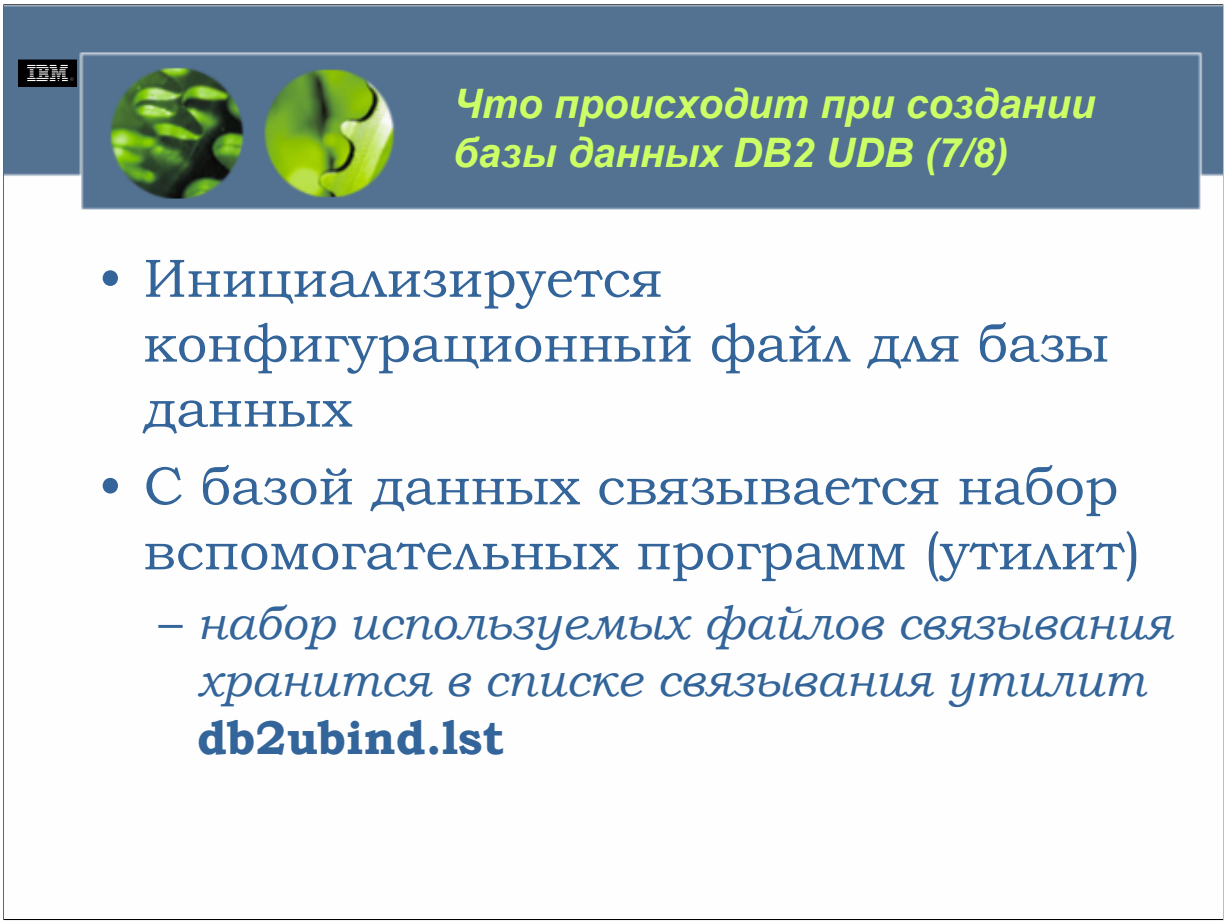
Создается, изменяется или удаляется объект базы данных (такой как таблица или индекс).

Предоставляются или отзываются полномочия и/или привилегии.

Собирается статистическая информация для таблицы.

С базой данных связываются пакеты.

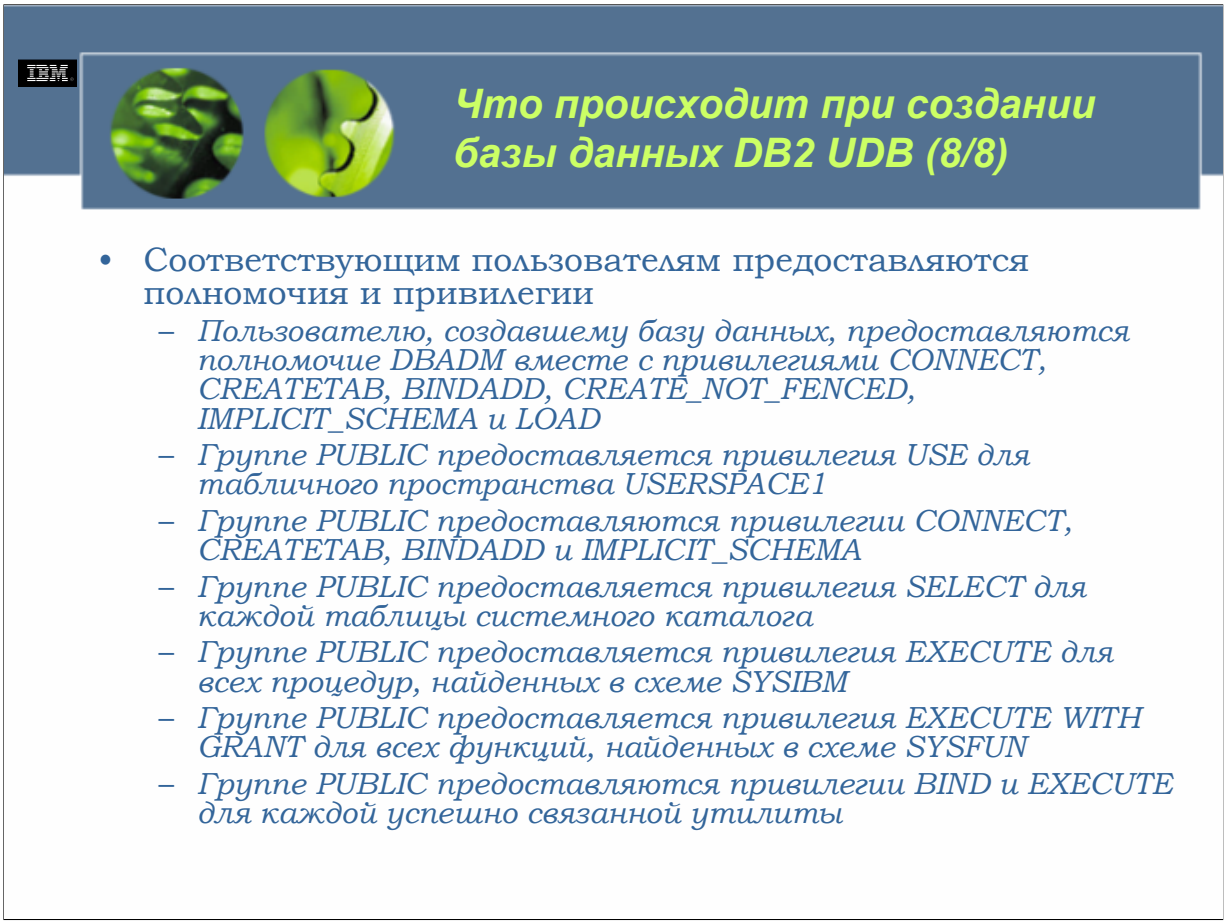
В большинстве случаев при создании объекта в одной или нескольких таблицах системного каталога сохраняются полные свойства объекта базы данных. Однако в некоторых случаях, таких как при определении триггеров или ограничений, вместо этого сохраняются реальные операторы SQL, использованные для создания объекта.



WCF03.0

Некоторые из параметров в конфигурационном файле базы данных (такие, как кодовый набор, территория и последовательность сортировки) будут установлены с использованием значений, которые были указаны в качестве входных данных для мастера создания базы данных или команды CREATE DATABASE. Другим будут назначены системные значения по умолчанию.

Прежде чем некоторые из доступных утилит DB2 UDB смогут работать с базой данных, должны быть созданы пакеты, необходимые для запуска этих утилит. Такие пакеты создаются посредством привязки к базе данных набора определенных файлов связывания менеджера баз данных DB2 (набор используемых файлов связывания хранится в списке связывания утилит *db2ubind.lst*).



Чтобы соединиться и работать с определенной базой данных у пользователя должны быть полномочия и привилегии, необходимые для использования этой базы данных. Поэтому каждый раз при создании новой базы данных предоставляются следующие полномочия и привилегии:

•Пользователю, создавшему базу данных, предоставляются полномочие DBADM вместе с привилегиями CONNECT, CREATETAB, BINDADD, CREATE_NOT_FENCED, IMPLICIT_SCHEMA и LOAD.

•Группе PUBLIC предоставляется привилегия USE для табличного пространства USERSPACE1.

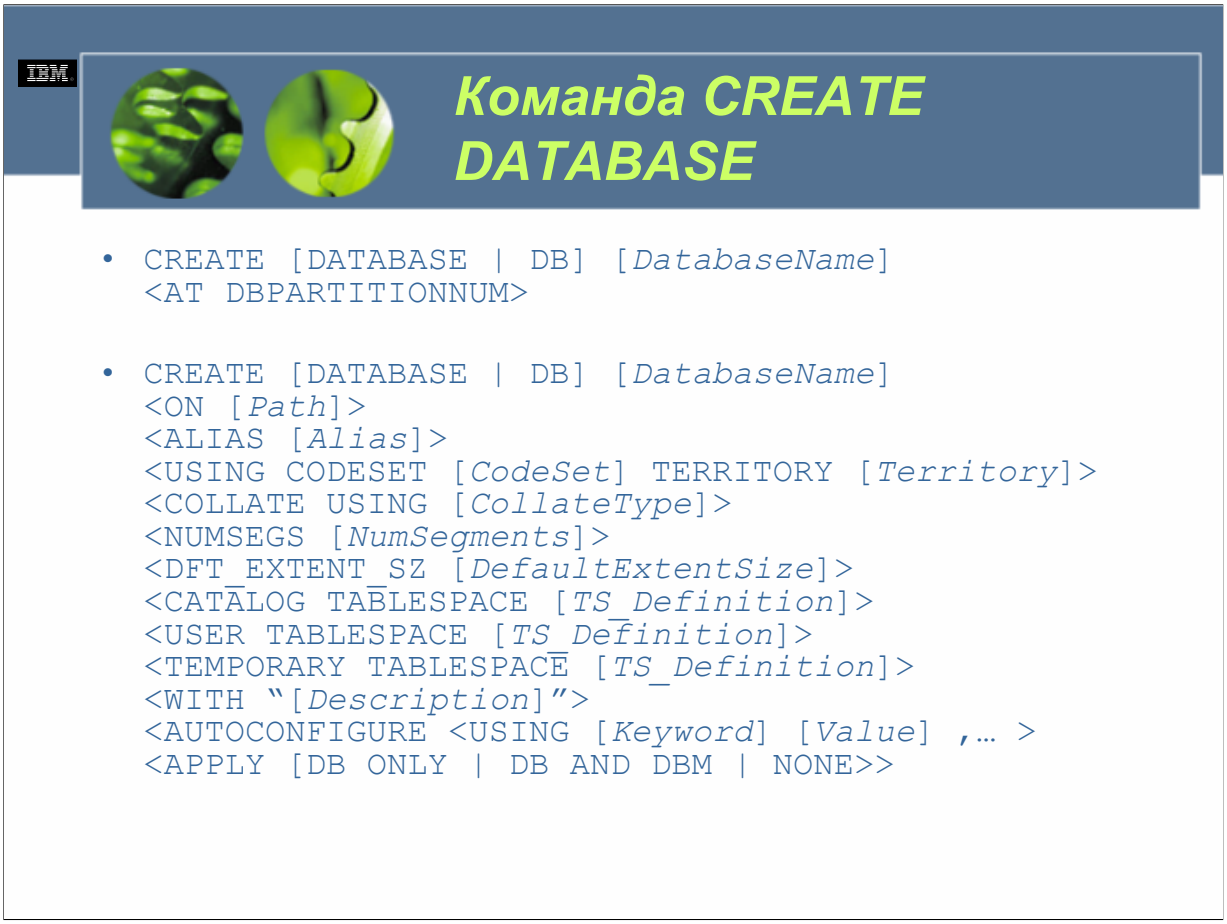
•Группе PUBLIC предоставляются привилегии CONNECT, CREATETAB, BINDADD и IMPLICIT_SCHEMA.

•Группе PUBLIC предоставляется привилегия SELECT для каждой таблицы системного каталога.

•Группе PUBLIC предоставляется привилегия EXECUTE для всех процедур, найденных в схеме SYSIBM.

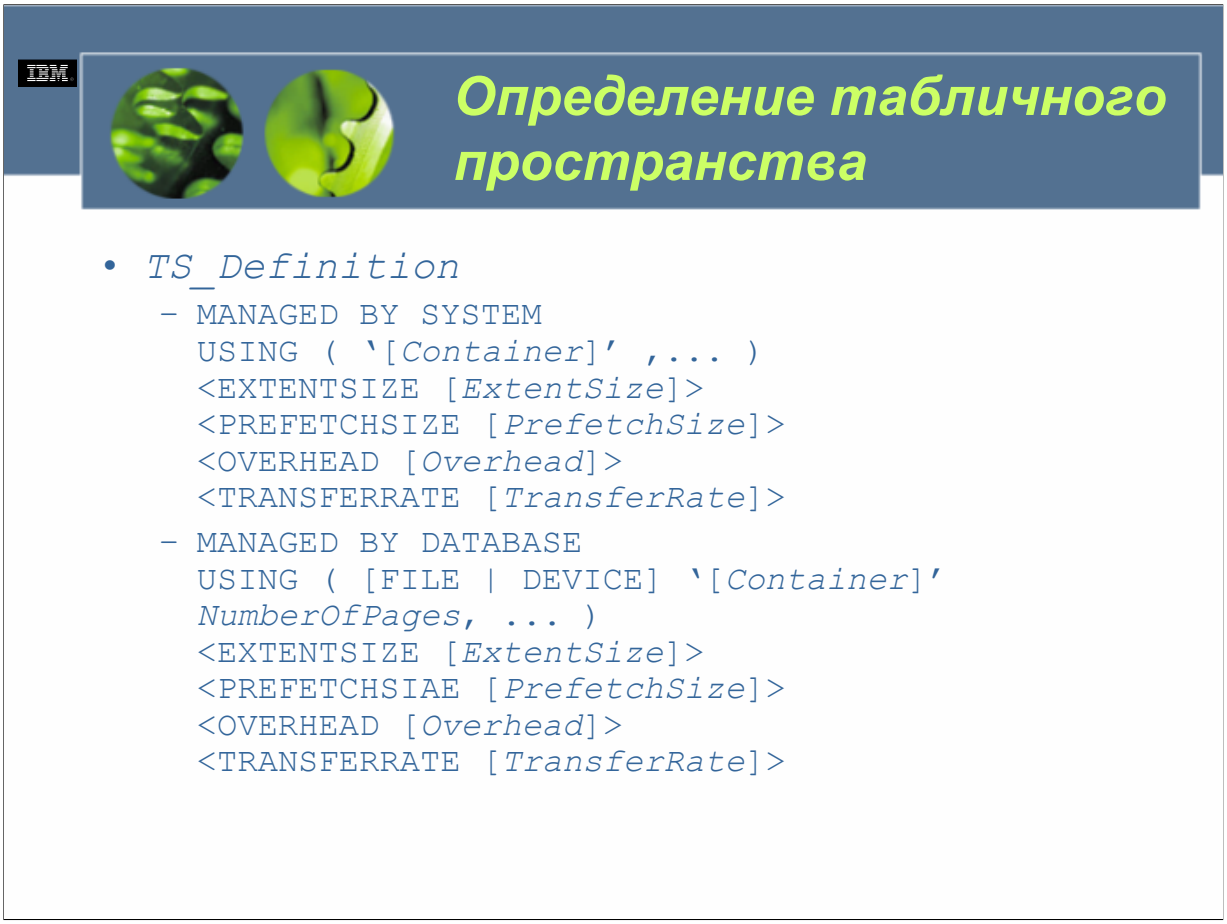
•Группе PUBLIC предоставляется привилегия EXECUTE WITH GRANT для всех функций, найденных в схеме SYSFUN.

•Группе PUBLIC предоставляются привилегии BIND и EXECUTE для каждой успешно связанной утилиты.



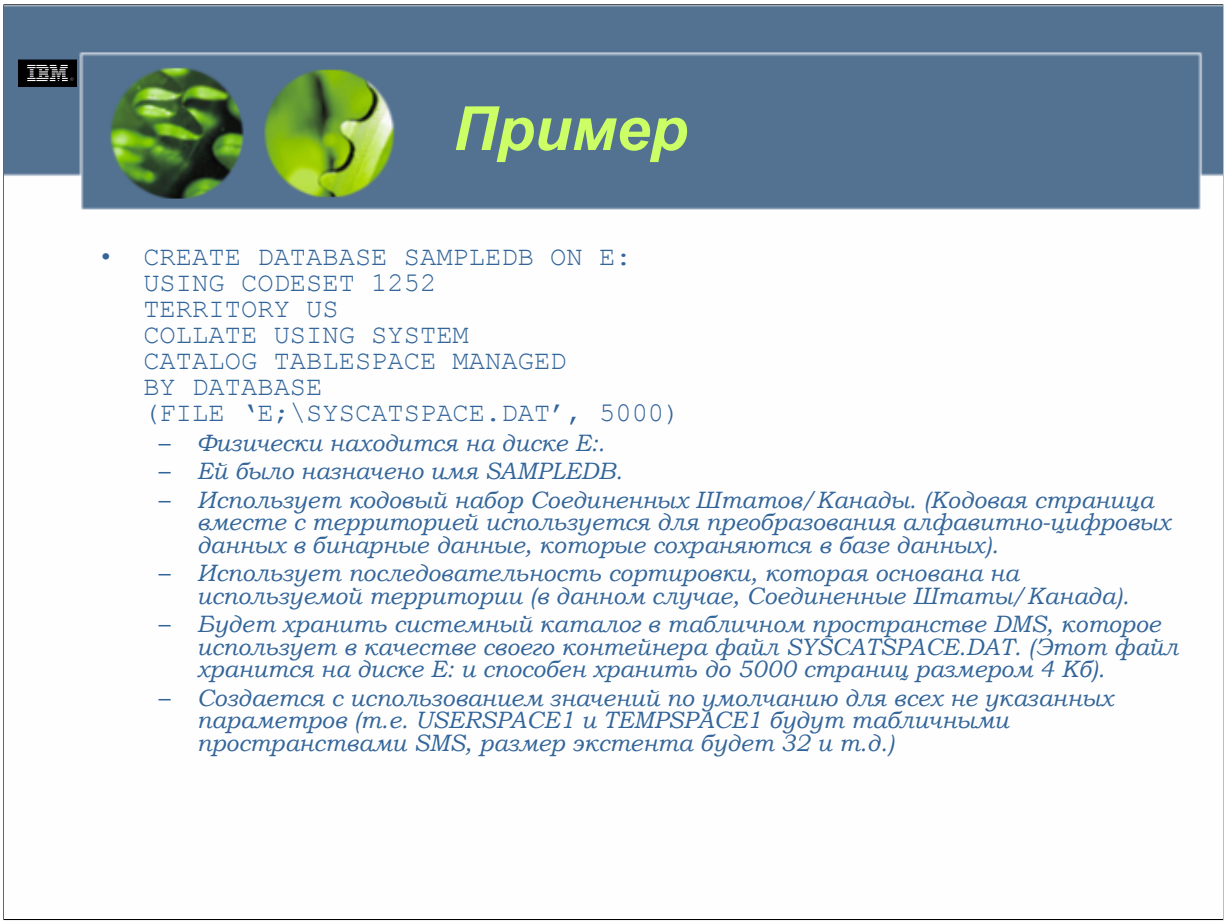
WCF03.0

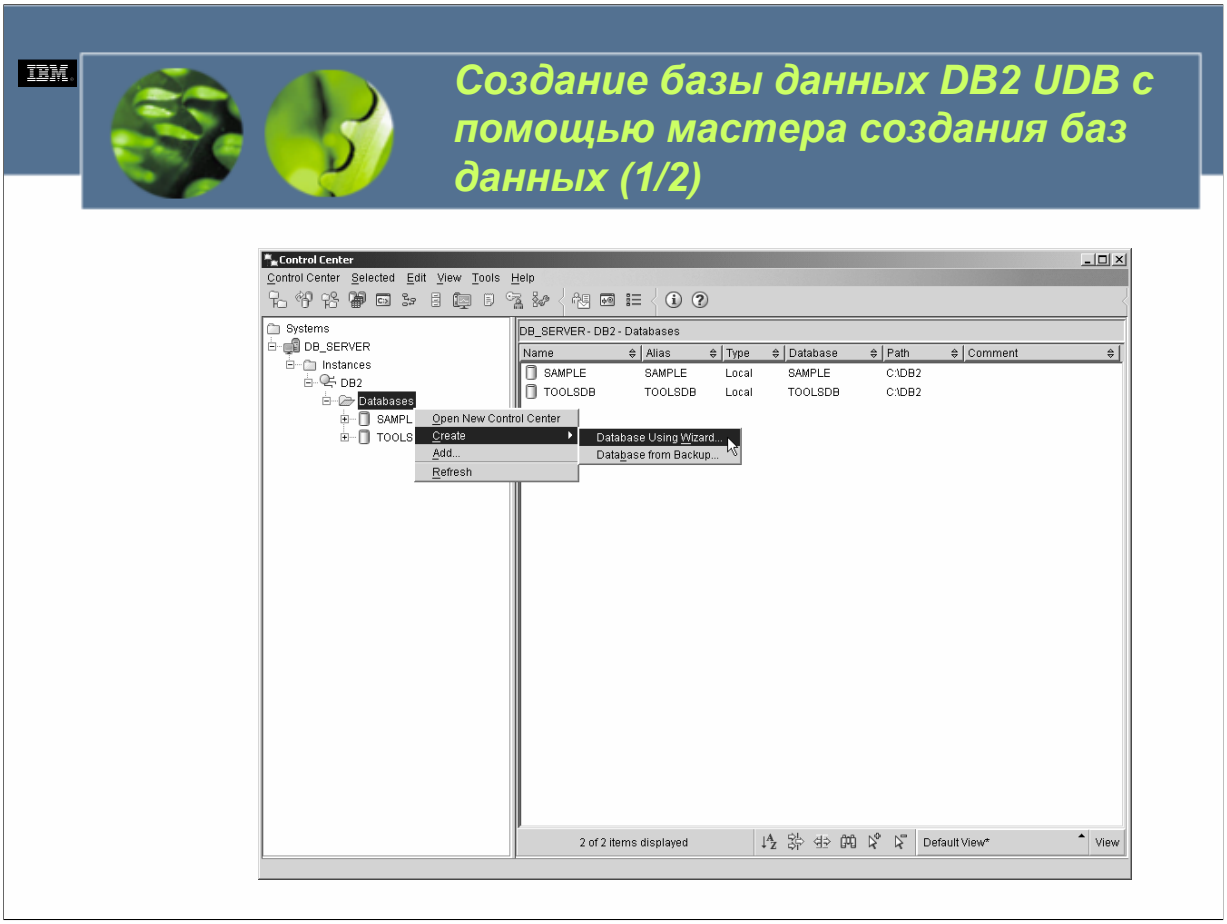
DatabaseName обозначает уникальное имя, которое должно быть назначено создаваемой базе данных. *Path* обозначает место (диск и/или каталог), где должны физически храниться создаваемые иерархия каталогов и файлы, связанные с базой данных. *Alias* определяет алиас, который должен быть связан с создаваемой базой данных. *CodeSet* определяет кодовый набор, который должен использоваться для хранения данных в создаваемой базе данных. (В базе данных DB2 UDB каждый однобайтный символ внутренне представляется в виде уникального числа от 0 до 255. Этот номер называется *кодом (code point)* символа; привязка кодов к каждому символу в определенном наборе символов называется *кодовой страницей*; а термином Международной организации по стандартизации для кодовой страницы является *кодовый набор*). *Territory* обозначает территорию, используемую для хранения данных в создаваемой базе данных. *CollateType* определяет последовательность сортировки (т.е. последовательность, в которой упорядочиваются символы в целях сортировки, объединения и сравнений), которая должна использоваться создаваемой базой данных. (Действительные значения включают COMPATABILITY, IDENTITY, NLSCHAR и SYSTEM). *NumSegments* определяет число сегментов, которые должны быть созданы и использованы для хранения файлов для каждого используемого создаваемой базой данных табличного пространства SMS (только SYSCATSPACE, USERSPACE1 и TEMPSPACE1). *DefaultExtentSize* определяет размер экстента для использования по умолчанию каждый раз, когда создается табличное пространство, а в ходе процесса создания размер экстента не указывается. *Description* комментарий, использующийся для описания элемента базы данных, который будет сделан в каталоге баз данных для создаваемой базы данных. Описание должно быть заключено в двойные кавычки. *Keyword* одно или несколько ключевых слов, распознаваемых командой AUTOCONFIGURE. (действительные значения включают mem_percent, workload_type, nub_stmts, tpm, admin_priority, is_populated, num_local_apps, num_remote_apps, isolation и bp_resizable – за дополнительной информацией о том, как используется команда AUTOCONFIGURE, обратитесь к Справочнику по командам DB2 UDB). *Value* определяет значение, которое должно быть связано с указанным ключевым словом. *TS_Definition* указывает определение, которое должно использоваться для создания табличного пространства, которое будет использоваться для хранения таблиц системного каталога (SYSCATSPACE), объектов, определенных пользователем (USERSPACE1), и/или временных объектов (TEMPSPACE1).



WCF03.0

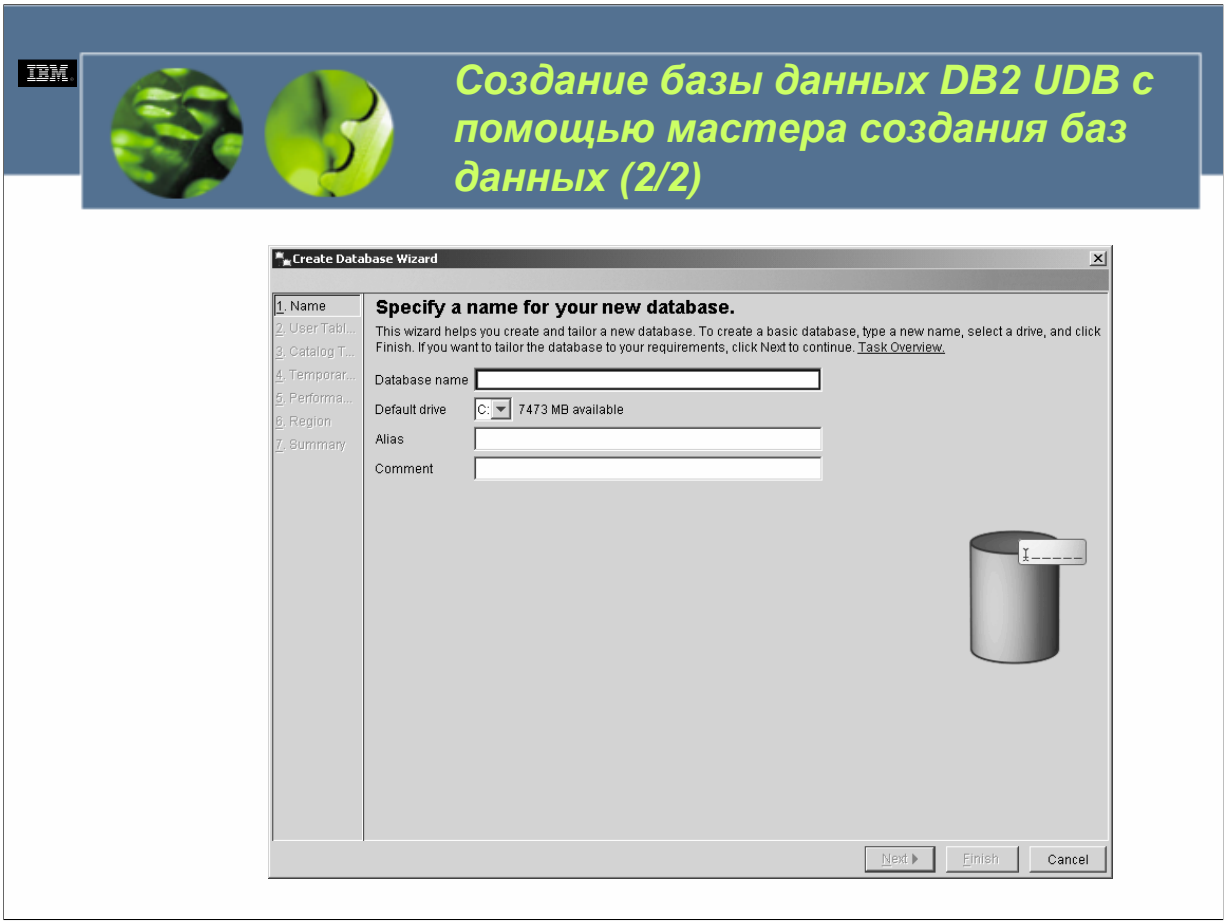
Container определяет один или несколько контейнеров, которые должны быть использованы для хранения данных, которые будут связаны с указанным табличным пространством. Для табличных пространств SMS каждый указанный контейнер должен указывать на действительный каталог; для контейнеров DMS FILE каждый указанный контейнер должен указывать действительный файл; а для контейнеров DMS DEVICE каждый указанный контейнер должен определять существующее устройство. *NumberOfPages* определяет число 4Кб страниц, которые должны использоваться контейнером табличного пространства. *ExtentSize* определяет число 4Кб страниц данных, которые будут записываться на манер карусели в каждый контейнер табличного пространства. *PrefetchSize* определяет число 4 Кб страниц данных, которые будут считываться из указанного табличного пространства при осуществлении предварительной выборки. *Overhead* определяет издержки контроллера ввода/вывода и время задержки поиска диска (в миллисекундах), связанные с контейнерами, принадлежащими указанному табличному пространству. *TransferRate* определяет время в миллисекундах, необходимое для чтения 4 Кб страницы данных из контейнера табличного пространства и сохранения их в памяти.





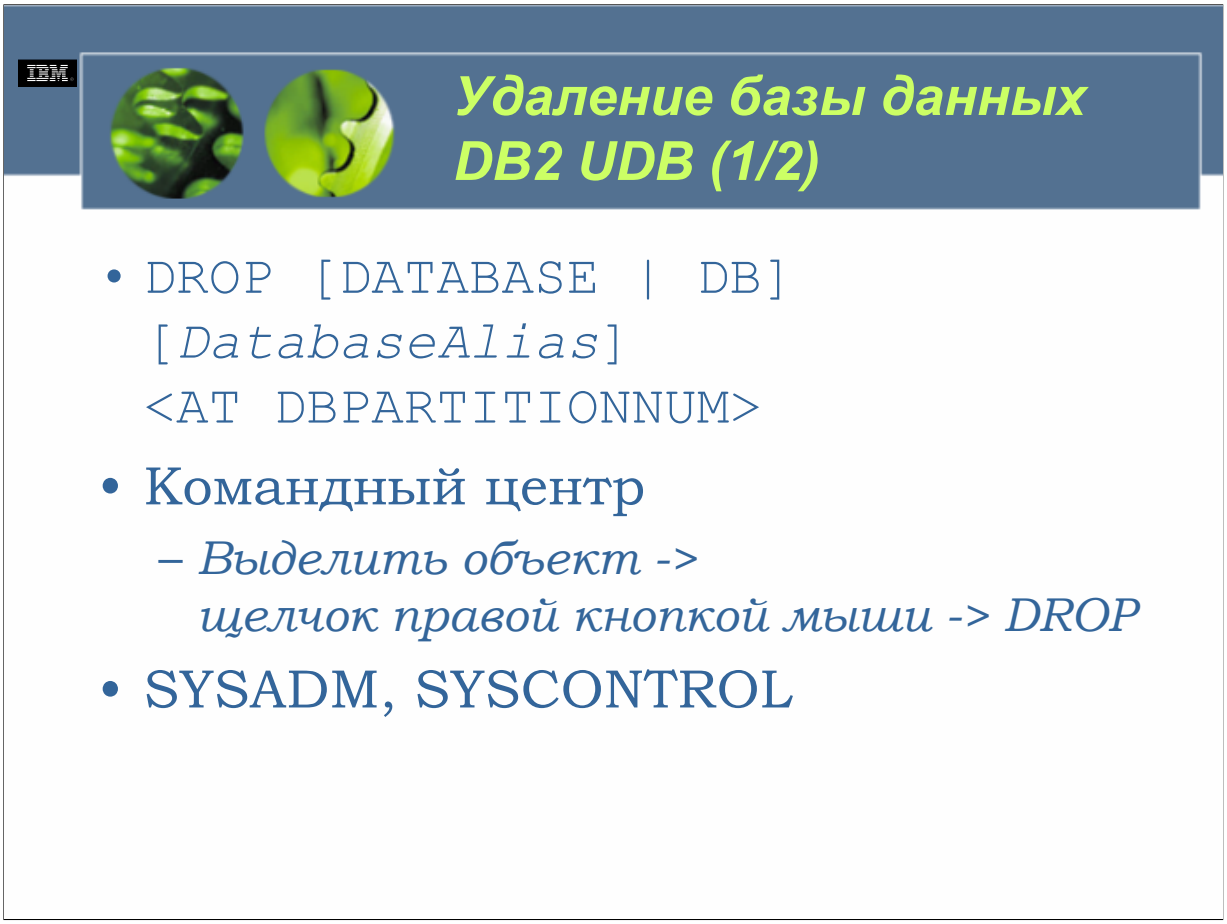
WCF03.0

Выделив объект *Базы данных*, показанный в панели объектов центра управления, и щелкнув правой клавишей мыши, вы можете вызвать меню, содержащее список доступных действий для объектов баз данных. Мастер создания баз данных вызывается путем выбора из меню *Создать (Create)* и затем *Базу данных с помощью мастера… (Database Using Wizard…)*.



WCF03.0

После отображения мастера создания баз данных вы просто следуете инструкциям, показанным на каждой представленной панели, для определения свойств создаваемой базы данных. (Эти же самые свойства могут быть определены посредством различных доступных опций с помощью команды CREATE DATABASE). После предоставления менеджеру баз данных DB2 достаточной для создания базы данных информации кнопка “Finish”, отображенная в правом нижнем углу мастера, становится доступной. При выборе этой кнопки создается указанная база данных.



WCF03.0

Точно также, как есть два способа создания базы данных DB2 UDB, есть два способа ее уничтожения: путем использования центра управления или путем использования команды DROP DATABASE. Выделив показанный в панели объектов центра управления объект, соответствующий базе данных, которая должна быть разрушена, и щелкнув правой кнопкой мыши, вы можете отобразить меню, содержащее список доступных для данной конкретной базы данных опций. Если вы затем выберите из меню элемент Отбросить *(Drop)*, вам будет представлено диалоговое окно для подтверждения, где у вас попросят подтвердить свое решение об удалении базы данных (в этом диалоговом окне будет показано название базы данных, которую вы собираетесь удалить, чтобы предотвратить случайное удаление по ошибке другой базы данных). После того, как вы подтвердите, что указанная база данных должна быть удалена, ее содержимое уничтожается, ее элементы удаляются как из системного, так и из локального каталога баз данных, а контейнеры ее табличных пространств становятся доступными для использования другими базами данных. На рис. 4-5 показаны элементы пунктов меню центра управления, которые должны быть выбраны для отбрасывания (удаления) существующей базы данных.

Базу данных можно также удалить, выполнив команду DROP DATABASE. Синтаксис для данной команды такой:

DROP [DATABASE | DB] [*DatabaseAlias*] <AT DBPARTITIONNUM>

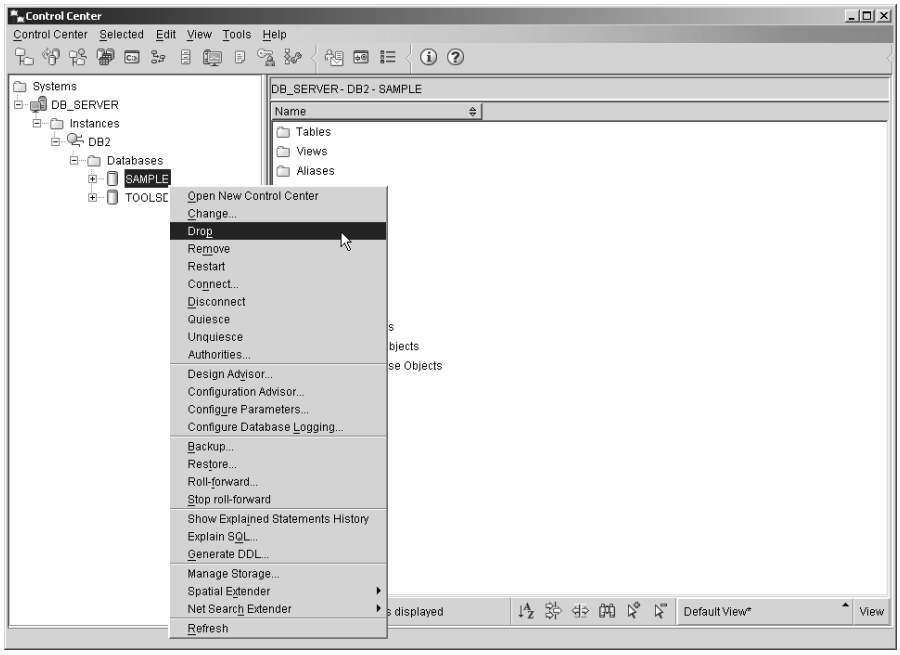
где:

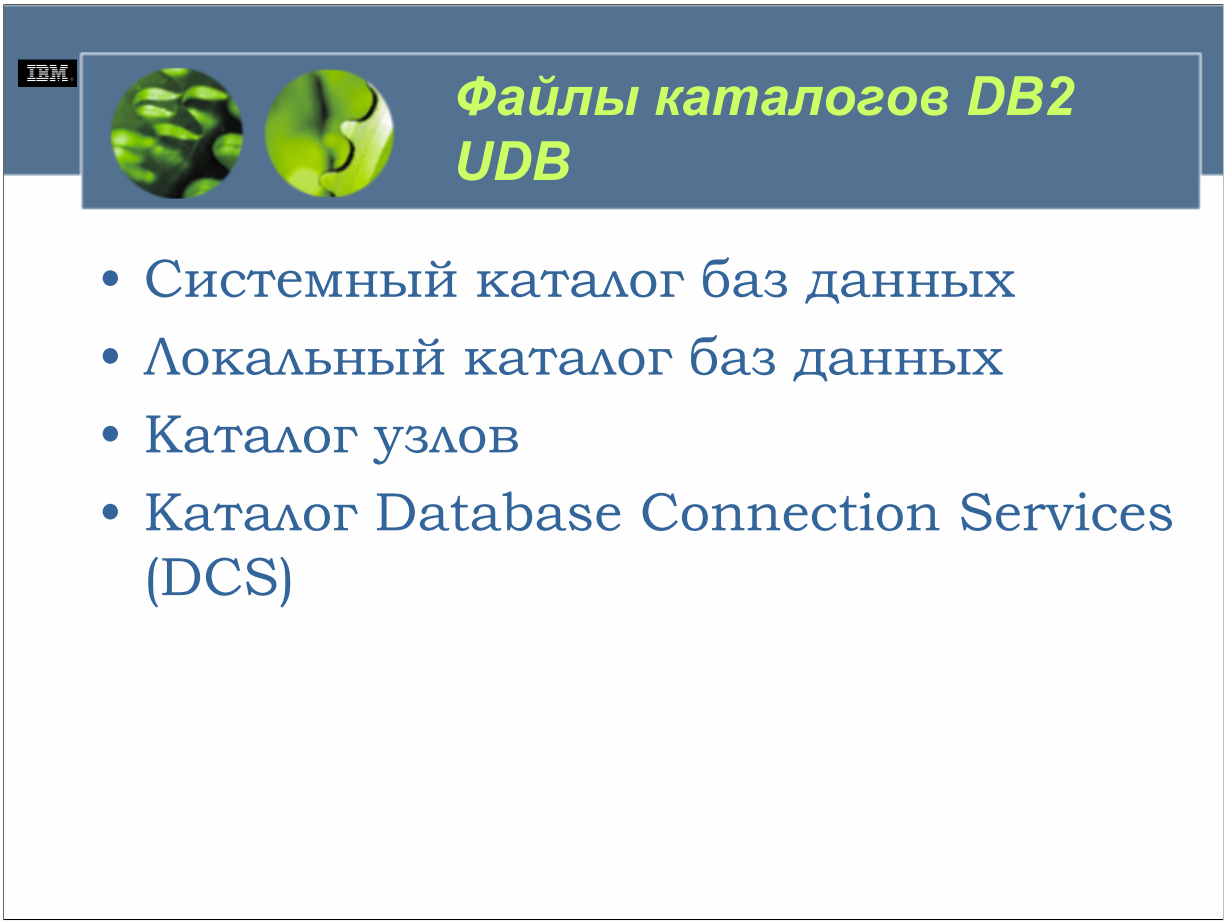
DatabaseAlias указывает алиас, связанный с уничтожаемой базой данных.Так, если бы вам было нужно уничтожить базу данных с именем и алиасом TEST_DB, вы могли бы сделать это, выполнив команду DROP DATABASE, которая выглядит следующим образом:

DROP DATABASE TEST_DB

Важно помнить, что удалять существующую базу данных позволено лишь пользователям с полномочиями системного администратора (SYSADM) или управления системой (SYSCTRL).

Удаление базы данных DB2 UDB (2/2)





WCF03.0

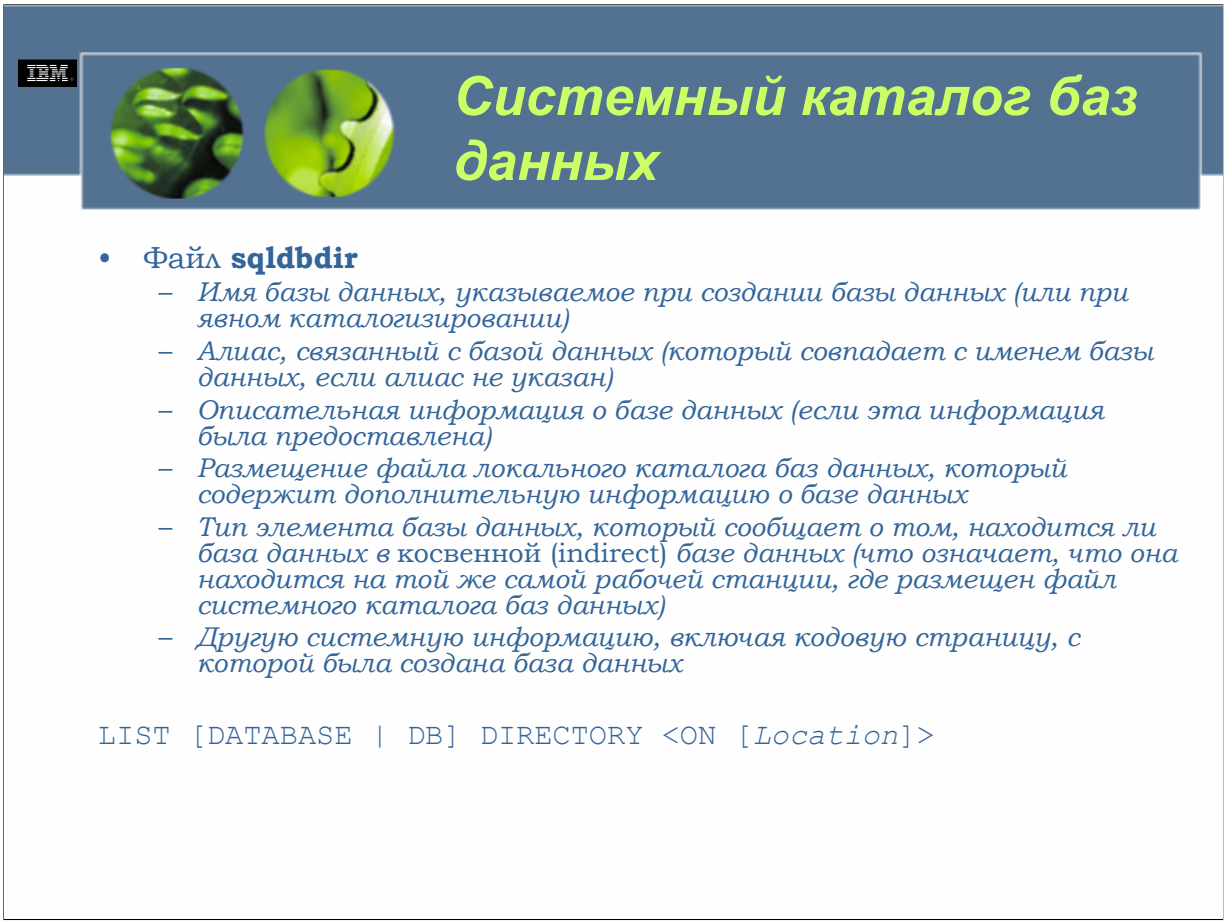
Ранее мы видели, что при создании базы данных она каталогизируется в системном и локальном каталогах баз данных. Так что представляют собой системный и локальный каталоги баз данных и почему база данных должна в них каталогизироваться? Если вы вспомните, когда мы изучали команду CREATE DATABASE, мы видели, что база данных физически может располагаться в любом месте системы. Из-за этого каждый экземпляр менеджера баз данных DB2 должен знать, где физически находятся базы данных, подпадающие по его контроль, а также как установить соединение с этими базами данных от имени пользователей и/или приложений. Для отслеживания этой информации DB2 UDB использует специальный набор файлов, известных как *файлы каталогов* (или *каталоги*). Существует четыре типа каталогов:

•Системный каталог баз данных

•Локальный каталог баз данных

•Каталог узлов

•Каталог Database Connection Services (DCS)



Системный каталог баз данных находится в файле с названием *sqldbdir*, который автоматически создается при создании первой базы данных для экземпляра. Затем информация о новой базе данных записывается в системном каталоге баз данных, и по мере каталогизирования дополнительных баз данных также записывается информация и об этих базах данных. (Базы данных неявно каталогизируются при своем создании; базы данных можно каталогизировать явным образом, используя центр управления или команду CATALOG DATABASE). Каждая запись, сделанная в системном каталоге баз данных, содержит следующую информацию:

Имя базы данных, указываемое при создании базы данных (или при явном каталогизировании).

Алиас, связанный с базой данных (который совпадает с именем базы данных, если алиас не указан).

Описательная информация о базе данных (если эта информация была предоставлена).

Размещение файла локального каталога баз данных, который содержит дополнительную информацию о базе данных.

Тип элемента базы данных, который сообщает о том, находится ли база данных в *косвенной (indirect)* базе данных (что означает, что она находится на той же самой рабочей станции, где размещен файл системного каталога баз данных).

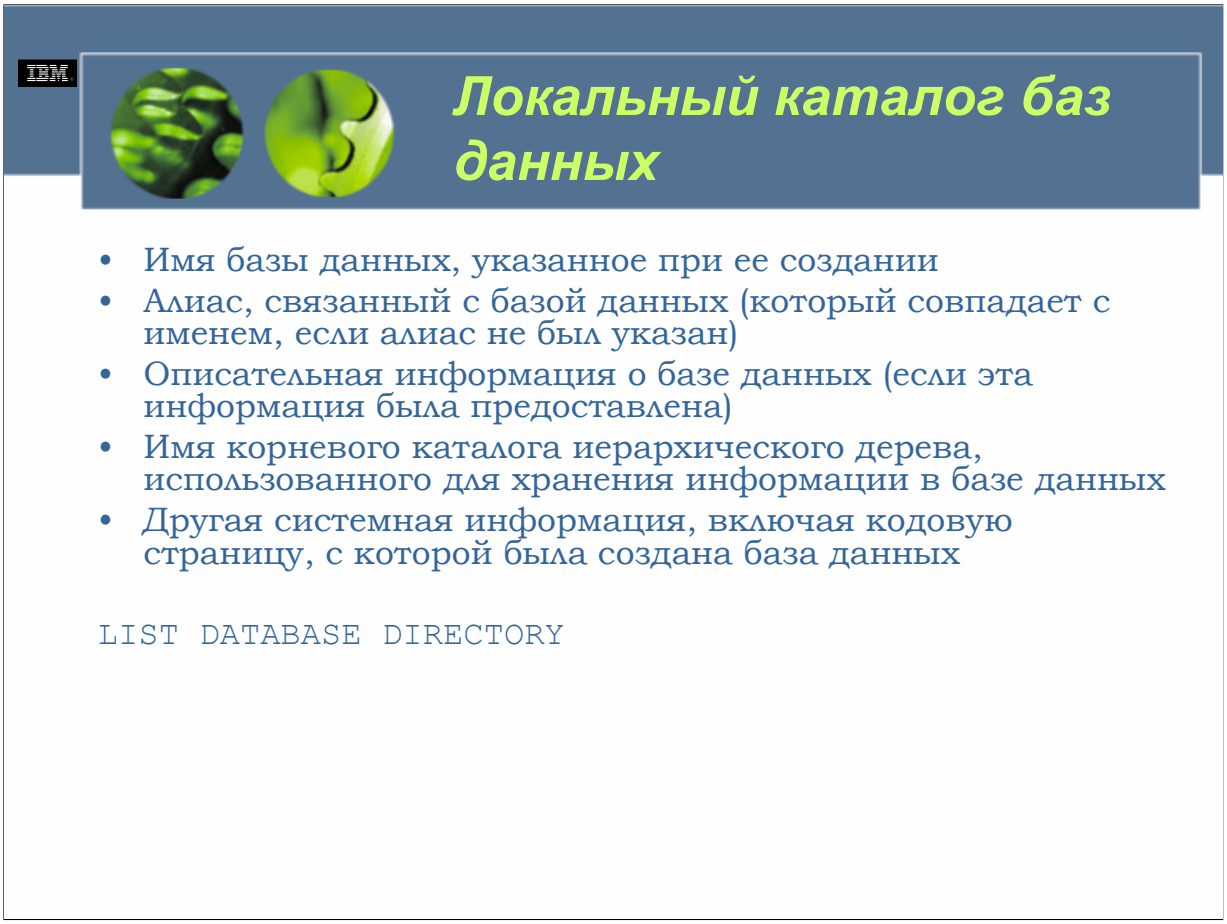
Другую системную информацию, включая кодовую страницу, с которой была создана база данных.

Содержание файла системного каталога баз данных или локального каталога баз данных можно просмотреть, выполнив команду LIST DATABASE DIRECTORY. Синтаксис этой команды следующий:

LIST [DATABASE | DB] DIRECTORY <ON [*Location*]>

где:

Location определяет диск или каталог, где хранятся одна или несколько баз данных.Если при выполнении данной команды место не указано, отображается содержание файла системного каталога баз данных. С другой стороны, если при выполнении данной команды указано место, будет отображено содержание файла локального каталога баз данных, который находится в этом определенном месте.



WCF03.0

Каждый раз, когда база данных DB2 UDB создается в новом месте (т.е. на новом диске или в новом каталоге), в этом же месте создается также файл локального каталога баз данных. Затем информация об этой базе данных записывается в локальном каталоге баз данных, а когда в этом месте создаются другие базы данных, информация о них также записывается в этом локальном каталоге баз данных. Таким образом, в то время как для определенного экземпляра существует лишь один системный каталог баз данных, могут существовать несколько локальных каталогов баз данных в зависимости от того, как в доступном пространстве хранения были распределены базы данных.

Каждый элемент, записанный в локальном каталоге баз данных, содержит следующую информацию:

Имя базы данных, указанное при ее создании.

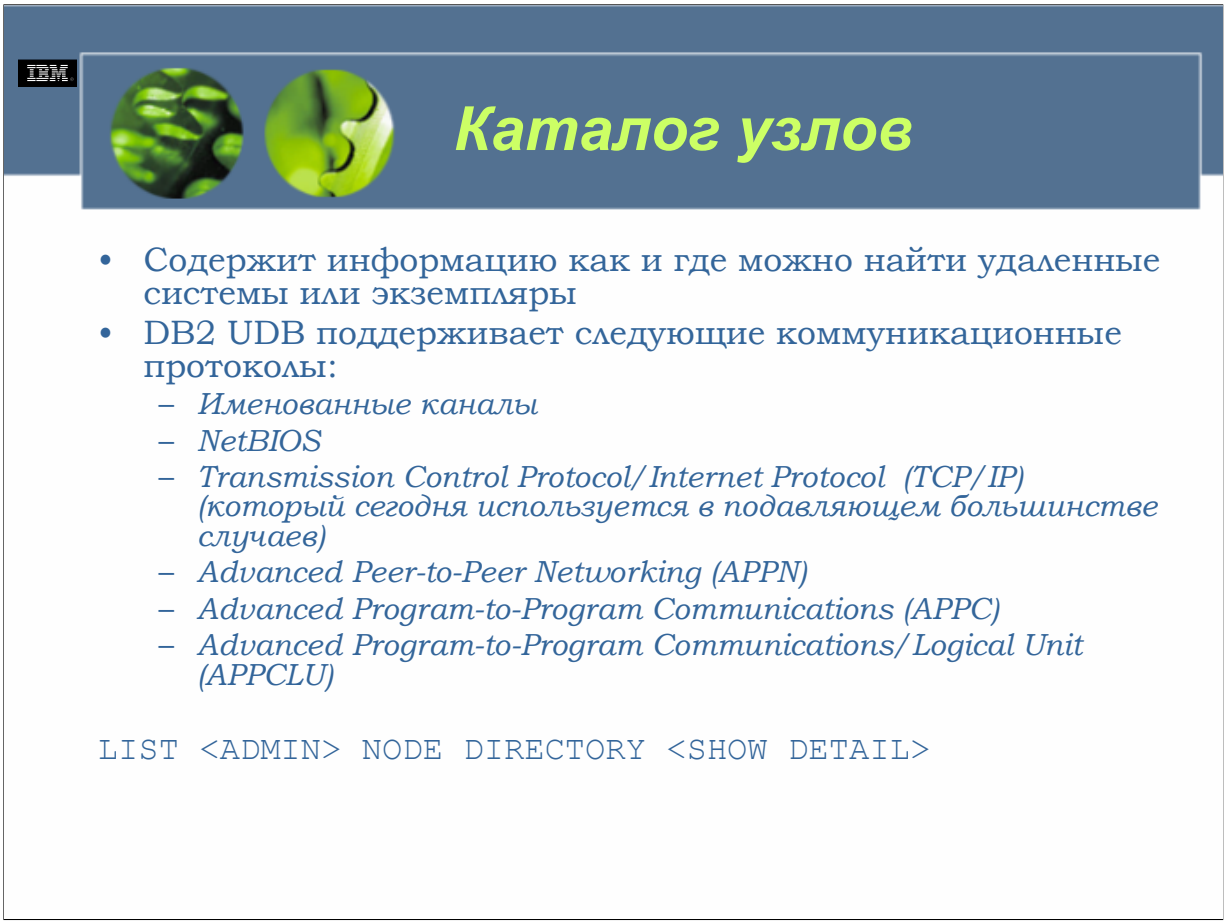
Алиас, связанный с базой данных (который совпадает с именем, если алиас не был указан).

Описательная информация о базе данных (если эта информация была предоставлена).

Имя корневого каталога иерархического дерева, использованного для хранения информации в базе данных.

Другая системная информация, включая кодовую страницу, с которой была создана база данных.

Как упоминалось ранее, содержание файла локального каталога баз данных можно просмотреть, выполнив команду LIST DATABASE DIRECTORY.



WCF03.0

В отличие от системного каталога баз данных и локального каталога баз данных, которые используются для отслеживания того, какие базы данных существуют и где они хранятся, каталог узлов содержит информацию, касающуюся того, как и где можно найти удаленные системы или экземпляры. Файл каталога узлов создается на каждой клиентской рабочей станции в первый раз, когда каталогизируется удаленный сервер или экземпляр. Когда каталогизируются другие удаленные экземпляры/серверы, информация об этих экземплярах/серверах также записывается в каталоге узлов. Затем элементы в каталоге узлов используются в сочетании с элементами в системном каталоге баз данных для установления соединений и подключений к базам данных DB2 UDB, хранящимся на удаленных серверах.

Каждый элемент в каталоге узлов содержит, помимо прочего, информацию о типе коммуникационного протокола, который должен использоваться для взаимодействия между рабочей станцией клиента и сервером удаленной базы данных. DB2 UDB поддерживает следующие коммуникационные протоколы:

•Именованные каналы

•NetBIOS

•Transmission Control Protocol/Internet Protocol (TCP/IP) (который сегодня используется в подавляющем большинстве случаев)

•Advanced Peer-to-Peer Networking (APPN)

•Advanced Program-to-Program Communications (APPC)

•Advanced Program-to-Program Communications/Logical Unit (APPCLU)

Содержание файла каталога узлов можно просмотреть, выполнив команду LIST NODE DIRECTORY. Синтаксис для данной команды следующий:

LIST <ADMIN> NODE DIRECTORY <SHOW DETAIL>

(Если при выполнении данной команды указана опция ADMIN, будет отображена информация об административных серверах).

WCF03.0

DB2 Connect является добавочным продуктом, использующимся для предоставления соединений с серверами приложений DRDA, такими как:

•DB2 для баз данных OS/390 или z/OS на хостах с архитектурой компьютеров System/370 и System/390.

•DB2 для баз данных VM и VSE на хостах с архитектурой компьютеров System/370 и System/390.

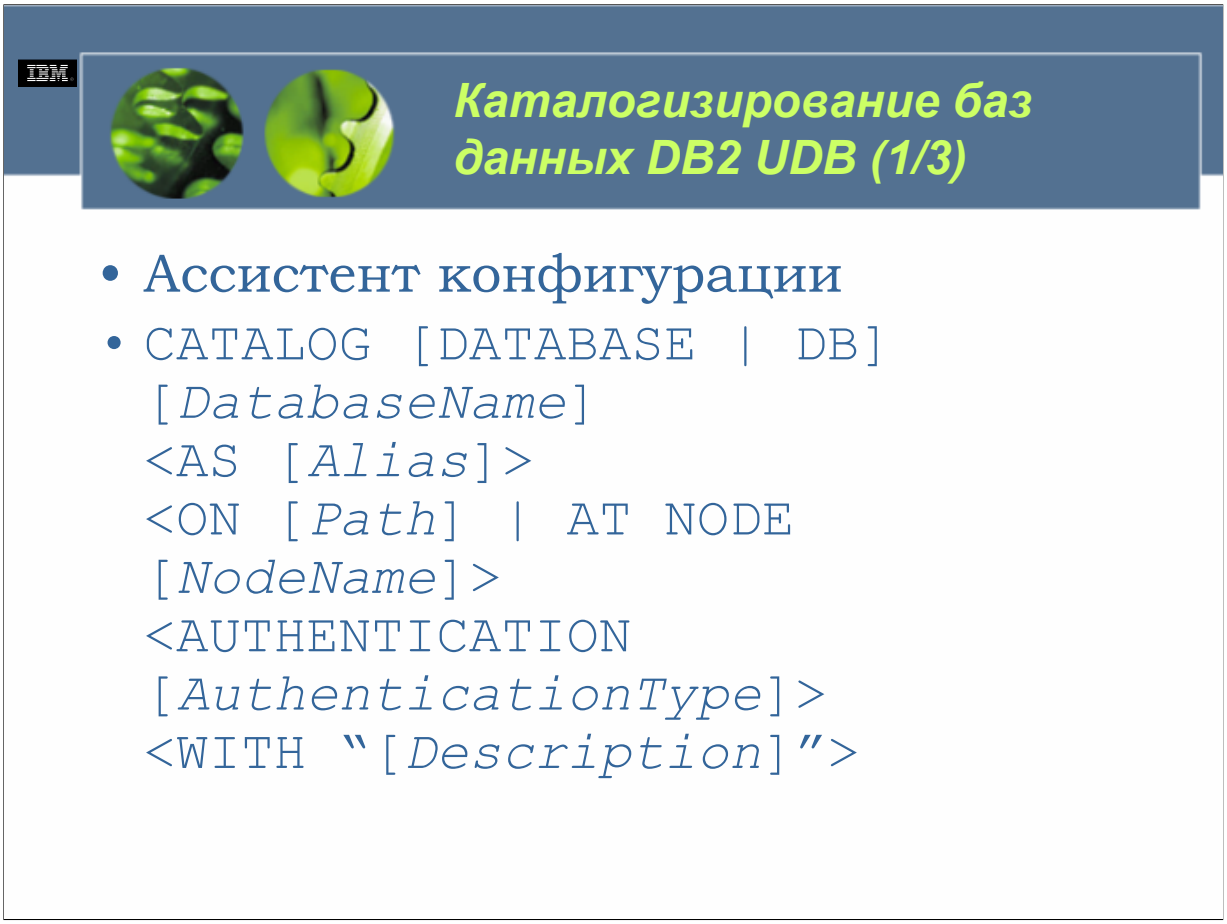
•Базы данных iSeries на компьютерах Application System/400 (AS/400) и iSeries.

Поскольку информация, необходимая для подключения к базам данных хостов DRDA отличается от информации, используемой для подключения к базам данных на основе локальных сетей, информация об удаленном хосте или базах данных iSeries хранится в специальном каталоге, известном как каталог службы соединения базы данных (Database Connection Services – DCS). Каждый раз, когда в каталоге DCS будет найден элемент с именем базы данных, соответствующим имени базы данных, хранящейся в системном каталоге баз данных, указанный реквестер прикладных программ (которым в большинстве случаев является DB2 Connect) будет пересылать выдаваемые запросы SQL удаленному серверу DRDA, где находится база данных.

Содержание файла каталога DCS можно просмотреть, выполнив команду LIST DCS DIRECTORY. Синтаксис данной команды следующий:

LIST DCS DIRECTORY

Важно отметить, что каталог DCS существует лишь в том случае, если был установлен продукт DB2 Connect.



Вы можете также каталогизировать базу данных, выполнив команду CATALOG DATABASE. Синтаксис для этой команды следующий:

CATALOG [DATABASE | DB] [*DatabaseName*]

<AS [*Alias*]>

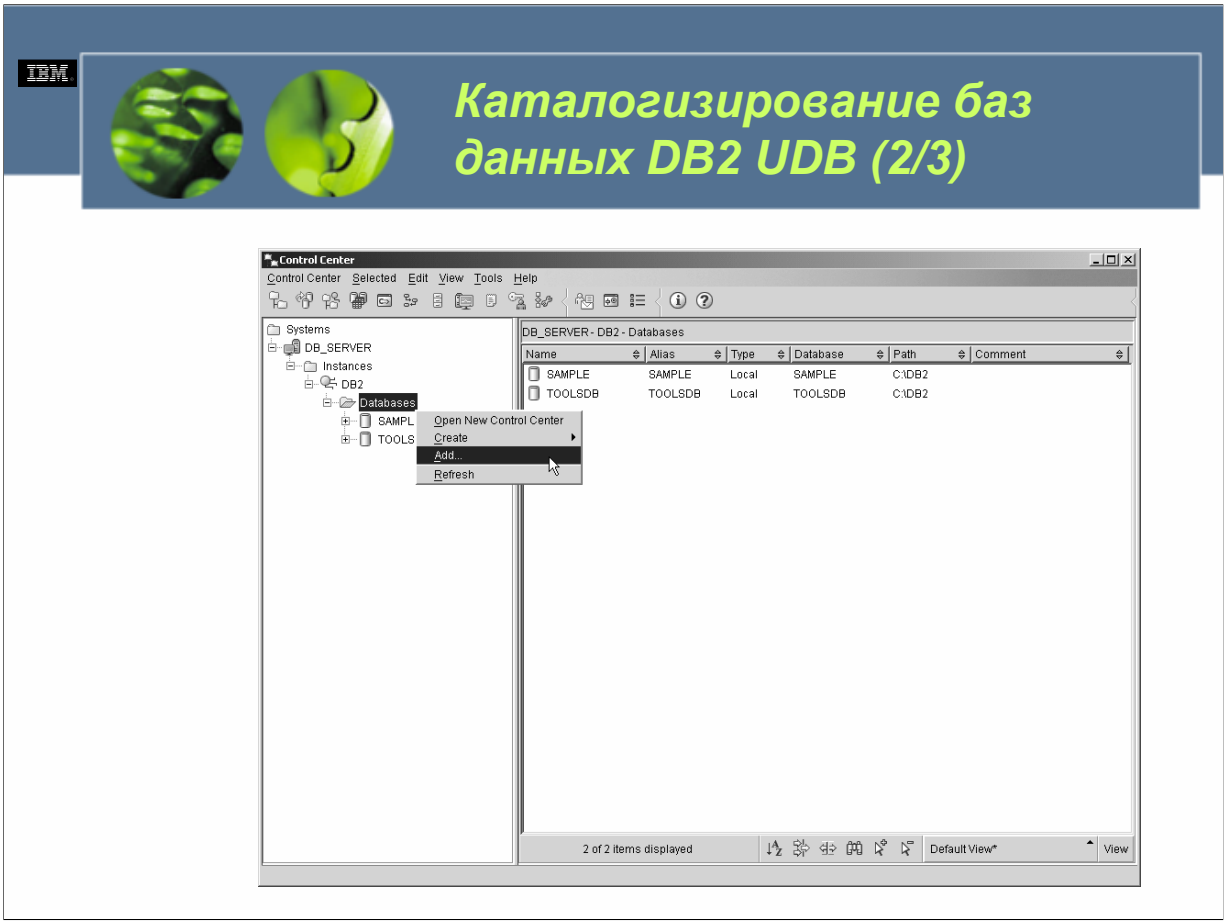
<ON [*Path*] | AT NODE [*NodeName*]>

<AUTHENTICATION [*AuthenticationType*]>

<WITH "[*Description*]">

где:

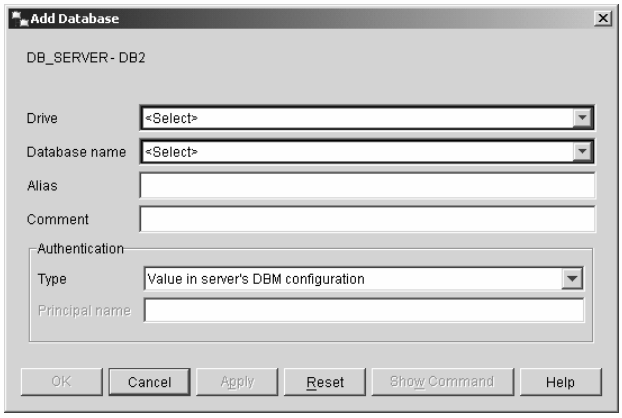
DatabaseName указывает имя, назначенное базе данных, которая должна быть каталогизирована. *Alias* указывает алиас, который должен быть связан с каталогизируемой базой данных. *Path* указывает место (диск и/или каталог), где физически хранятся иерархия каталогов и файлы, связанные с каталогизируемой базой данных. *NodeName* указывает узел, где находится каталогизируемая база данных. Указанное имя узла должно соответствовать элементу в файле каталога узлов (т.е. должно соответствовать каталогизированному узлу). *AuthenticationType* определяет, где и как должна происходить аутентификация, когда пользователь пытается получить доступ к базе данных. (Действительные значения включают: SERVER, CLIENT, SERVER_ENCRYPT и KERBEROS TARGET PRINCIPAL *PrincipalName*, где *PrincipalName* является полным именем принципала Kerberos для целевого сервера). *Description* комментарий, используемый для описания элемента базы данных, который будет сделан в каталоге баз данных для каталогизируемой базы данных. Описание должно быть заключено в двойные кавычки.

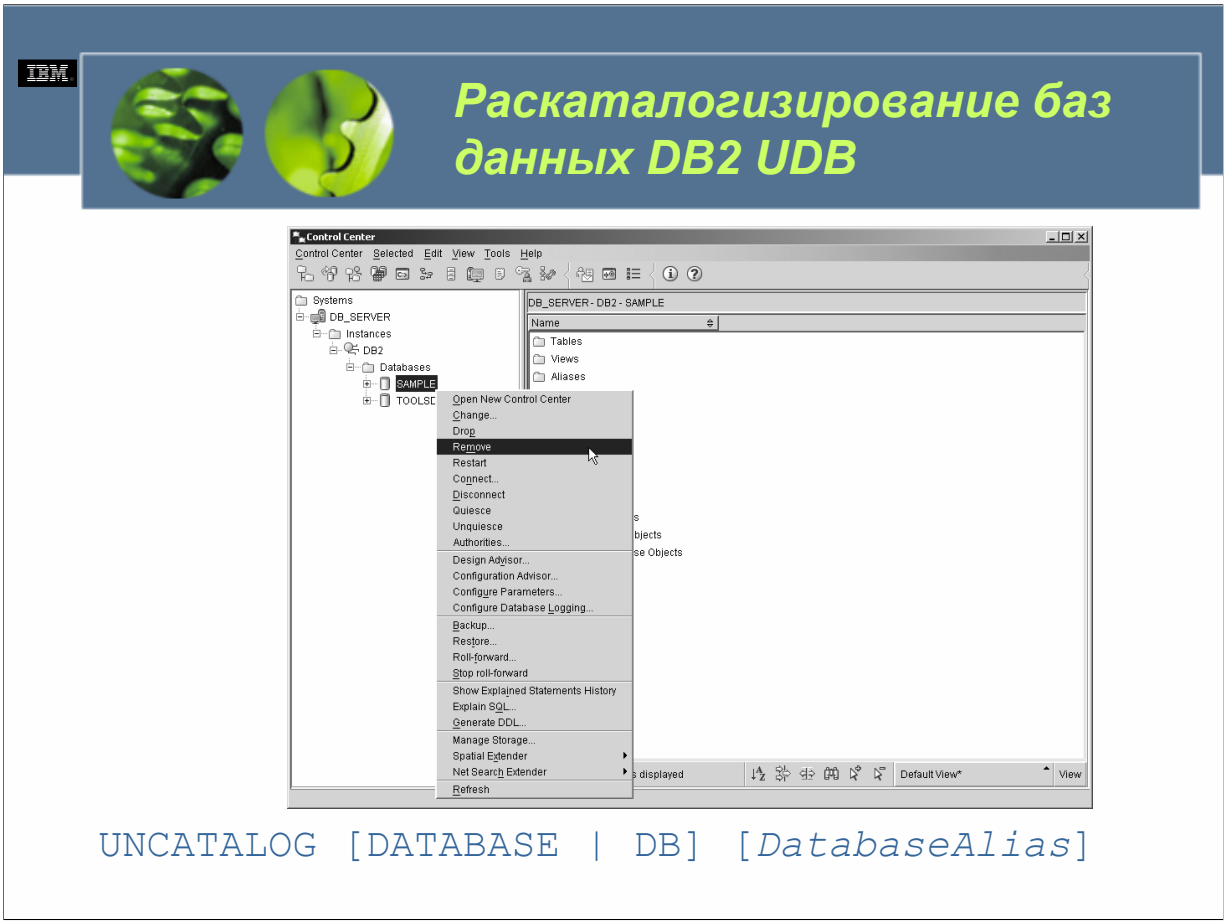


WCF03.0

Путем выделения объекта *Базы данных*, отображенного в панели объектов центра управления, и щелчка правой клавиши мыши вы вызываете меню, содержащее список доступных для объектов баз данных опций. Диалог, используемый для каталогизации баз данных (диалог добавления базы данных) вызывается путем выбора из данного меню *Добавить… (Add…)*.

Каталогизирование баз данных DB2 UDB (3/3)





WCF03.0

Также, как есть два различных способа каталогизации базы данных DB2 UDB, когда ассистент конфигурирования недоступен, есть два способа ее раскаталогизирования: путем использования центра управления или путем исполнения команды UNCATALOG DATABASE. Выделив представленный в панели объектов центра управления объект, соответствующий базе данных, которая должна быть удалена из каталога баз данных, и щелкнув правой кнопкой мыши, вы можете отобразить меню, содержащее список доступных для данной конкретной базы данных опций. Если затем вы выберите из меню элемент *Удалить (Remove)*, вам будет представлен диалог подтверждения, где у вас попросят подтвердить свое решение раскаталогизировать базу данных (в этом диалоге будет отображено название базы данных, чтобы предотвратить случайное раскаталогизирование другой базы данных). После подтверждения того, что указанная база данных должна быть раскаталогизирована, ее элемент удаляется как из системного, так и локального каталога баз данных; однако сама база данных не уничтожается, также, как и ее контейнеры табличных пространств не становятся доступными для использования другими базами данных.

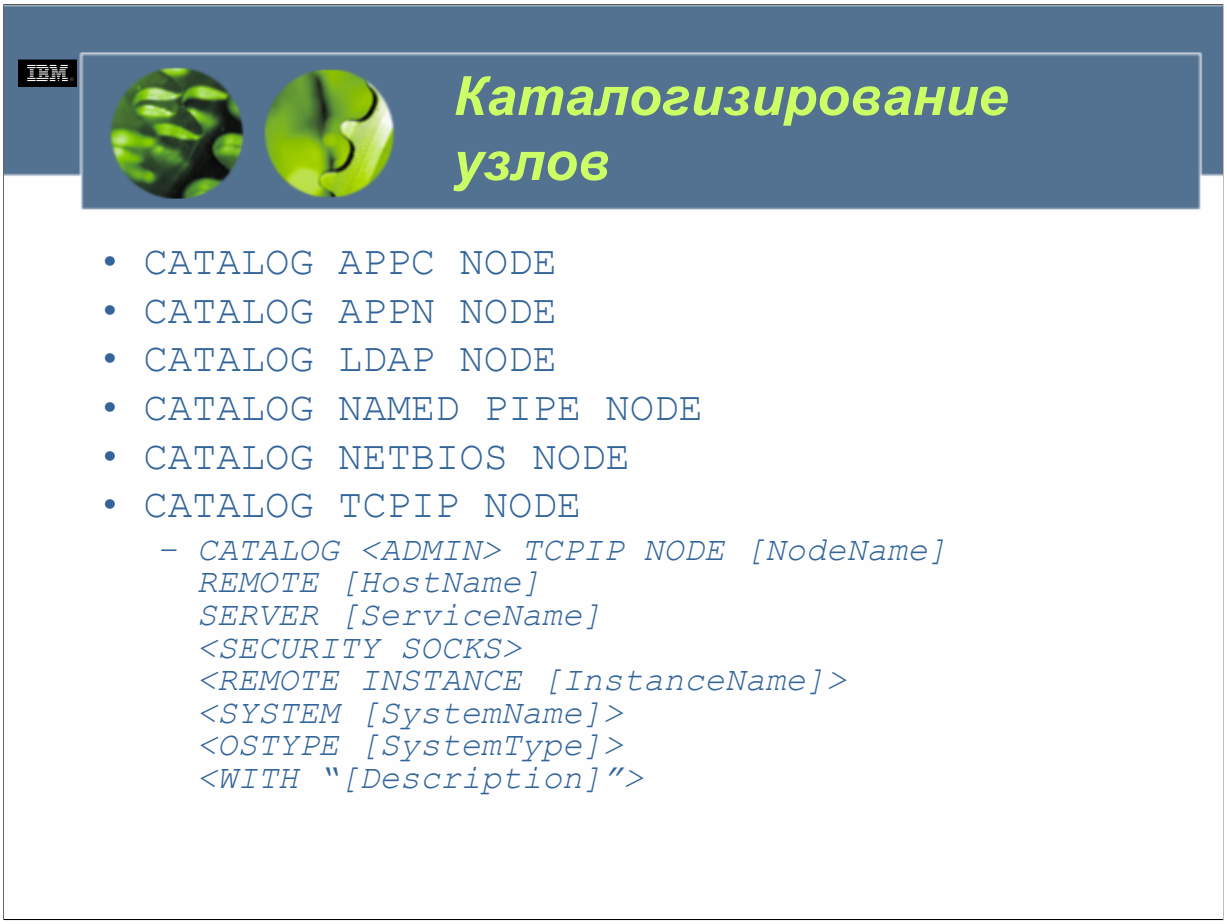
Базу данных можно также раскаталогизировать, выполнив команду UNCATALOG DATABASE. Синтаксис команды следующий:

UNCATALOG [DATABASE | DB] [*DatabaseAlias*]

где:

DatabaseAlias определяет алиас, связанный с раскаталогизируемой базой данных.Так, если бы вам было нужно раскаталогизировать базу данных, имеющую имя и алиас TEST_DB, вы могли бы сделать это, выполнив команду UNCATALOG DATABASE, которая выглядела бы так:

UNCATALOG DATABASE TEST_DB



Процесс, используемый для каталогизирования узлов (серверов), значительно отличается от процесса, используемого для каталогизирования баз данных. Вместо явного каталогизирования по мере необходимости узлы обычно каталогизируются неявно каждый раз, когда с использованием ассистента конфигурирования каталогизируется удаленная база данных. Однако если вы хотите явным образом каталогизировать определенный узел, вы можете сделать это, выполнив команду CATALOG … NODE, которая соответствует коммуникационному протоколу, который будет использоваться для доступа к каталогизируемому серверу. Доступно несколько форм команды CATALOG … NODE, включая:

•CATALOG APPC NODE

•CATALOG APPN NODE

•CATALOG LDAP NODE

•CATALOG NAMED PIPE NODE

•CATALOG NETBIOS NODE

•CATALOG TCPIP NODE

Синтаксис для всех этих команд очень сходен, причем главным отличием является то, что многие из опций, доступных с каждой из них, являются специфическими для коммуникационного протокола, для которого приспособлена команда. Поскольку TCP/IP несомненно является наиболее обычным используемым сегодня коммуникационным протоколом, давайте рассмотрим синтаксис для этой формы команды CATALOG … NODE.

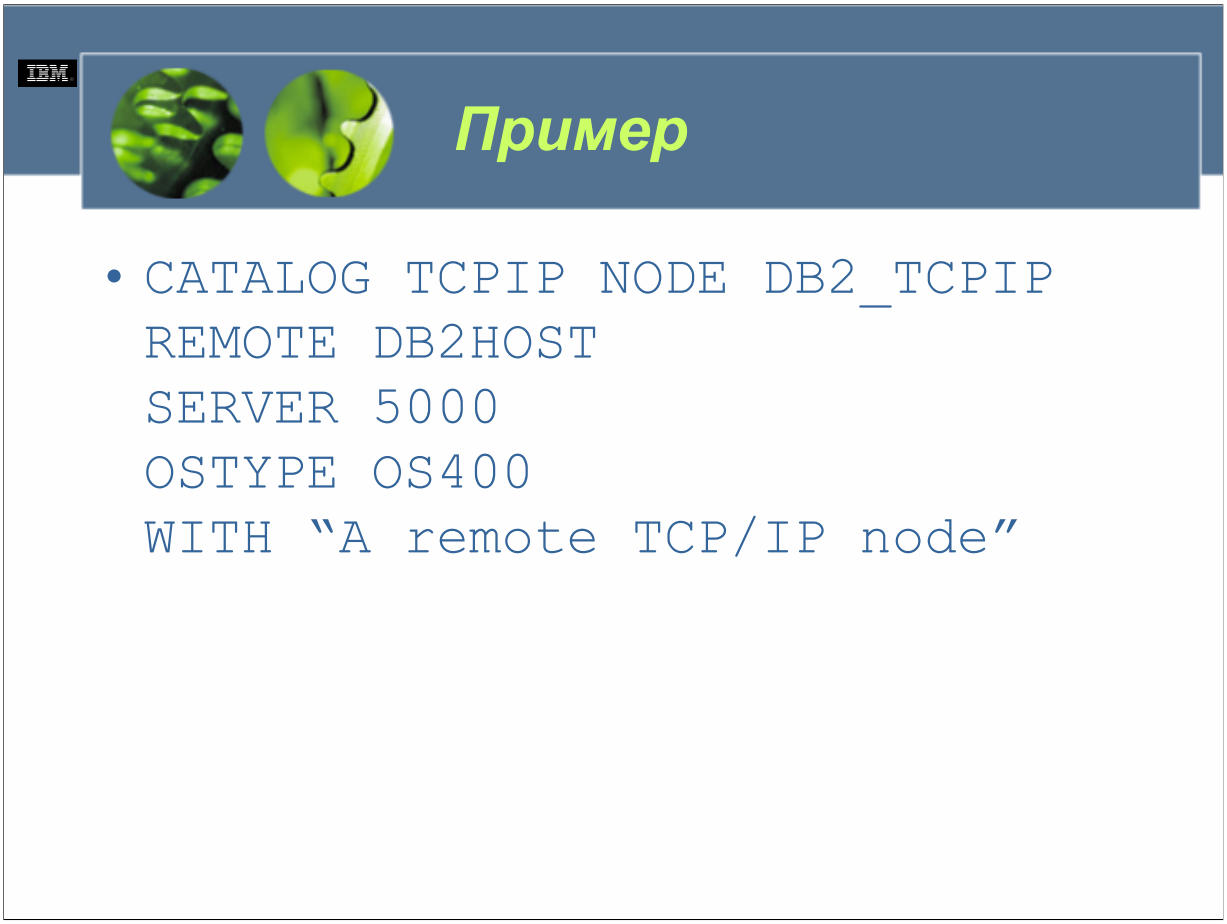
Синтаксис для команды CATALOG TCPIP NODE следующий:

CATALOG <ADMIN> TCPIP NODE [*NodeName*] REMOTE [*HostName*] SERVER [*ServiceName*] <SECURITY SOCKS>

<REMOTE INSTANCE [*InstanceName*]> <SYSTEM [*SystemName*]> <OSTYPE [*SystemType*]> <WITH "[*Description*]">

где:

*NodeName*Определяет алиас, который должен быть связан с каталогизируемым узлом. Это произвольное имя, созданное на рабочей станции пользователя и использующееся для идентификации узла.*HostName*Обозначает имя хоста в том виде, как оно известно в сети TCP/IP. (Это имя сервера, на котором находится удаленная база данных, с которой вы пытаетесь взаимодействовать). *ServiceName*Идентифицирует имя службы или номер порта, с которым будет взаимодействовать экземпляр менеджера баз данных DB2 . *InstanceName*Обозначает имя экземпляра сервера, с которым должно быть установлено соединение. *SystemName*Указывает имя системы DB2, которая используется для идентификации рабочей станции сервера. *SystemType*Обозначает тип операционной системы, использующейся на рабочей станции сервера. (Действительные значения включают: AIX, WIN, HPUX, SUN, OS390, OS400, VM, VSE, SNI, SCO и LINUX). *Description*Комментарий, используемый для описания элемента узла, который будет сделан в каталоге узла для каталогизируемого узла. Описание должно быть заключено в двойные кавычки.



WCF03.0

Если бы вам было нужно каталогизировать узел для рабочей станции AS/400 сервера с именем DB2HOST и назначить ему алиас DB2_TCPIP, вы могли бы сделать это, выполнив команду CATALOG TCPIP NODE, которая выглядит примерно следующим образом:

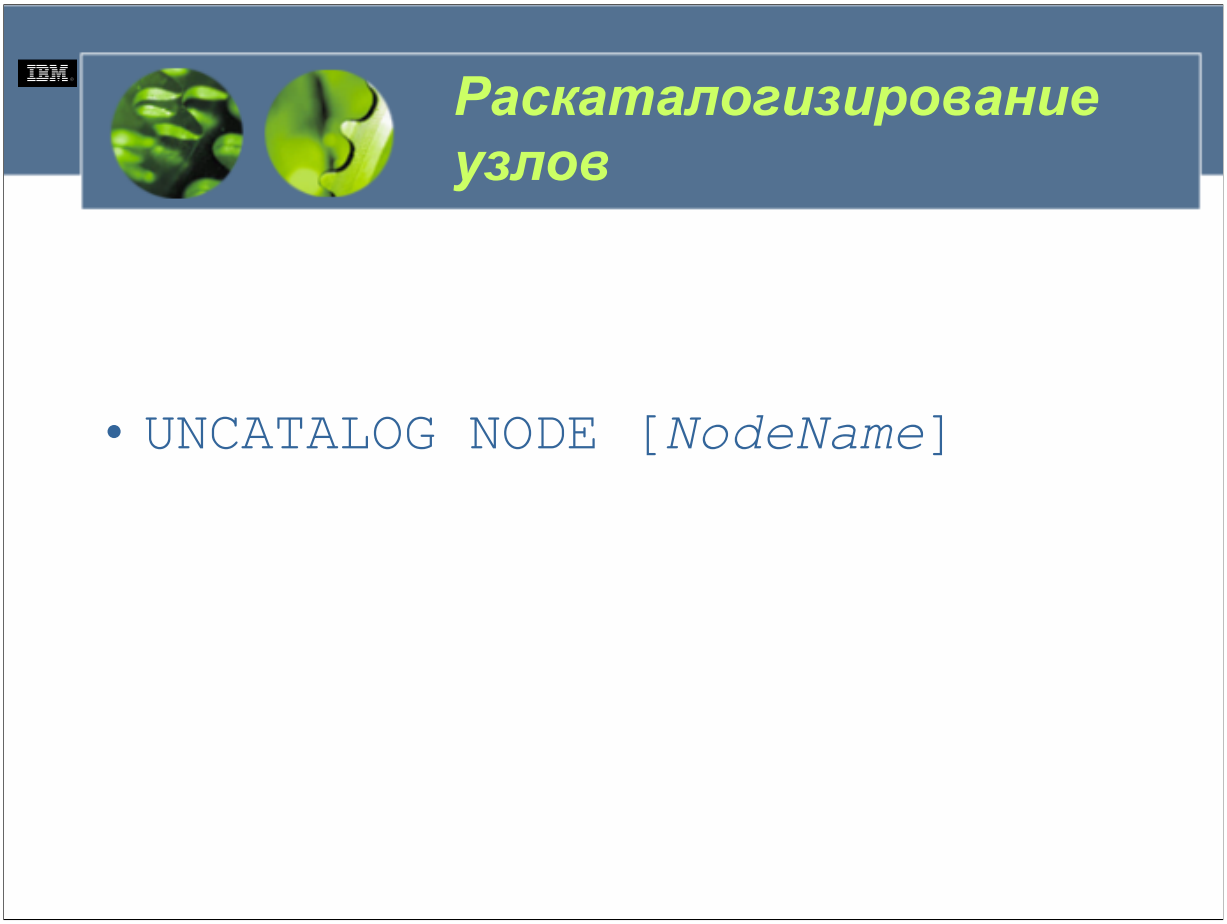
CATALOG TCPIP NODE DB2_TCPIP

REMOTE DB2HOST

SERVER 5000

OSTYPE OS400

WITH “A remote TCP/IP node”



WCF03.0

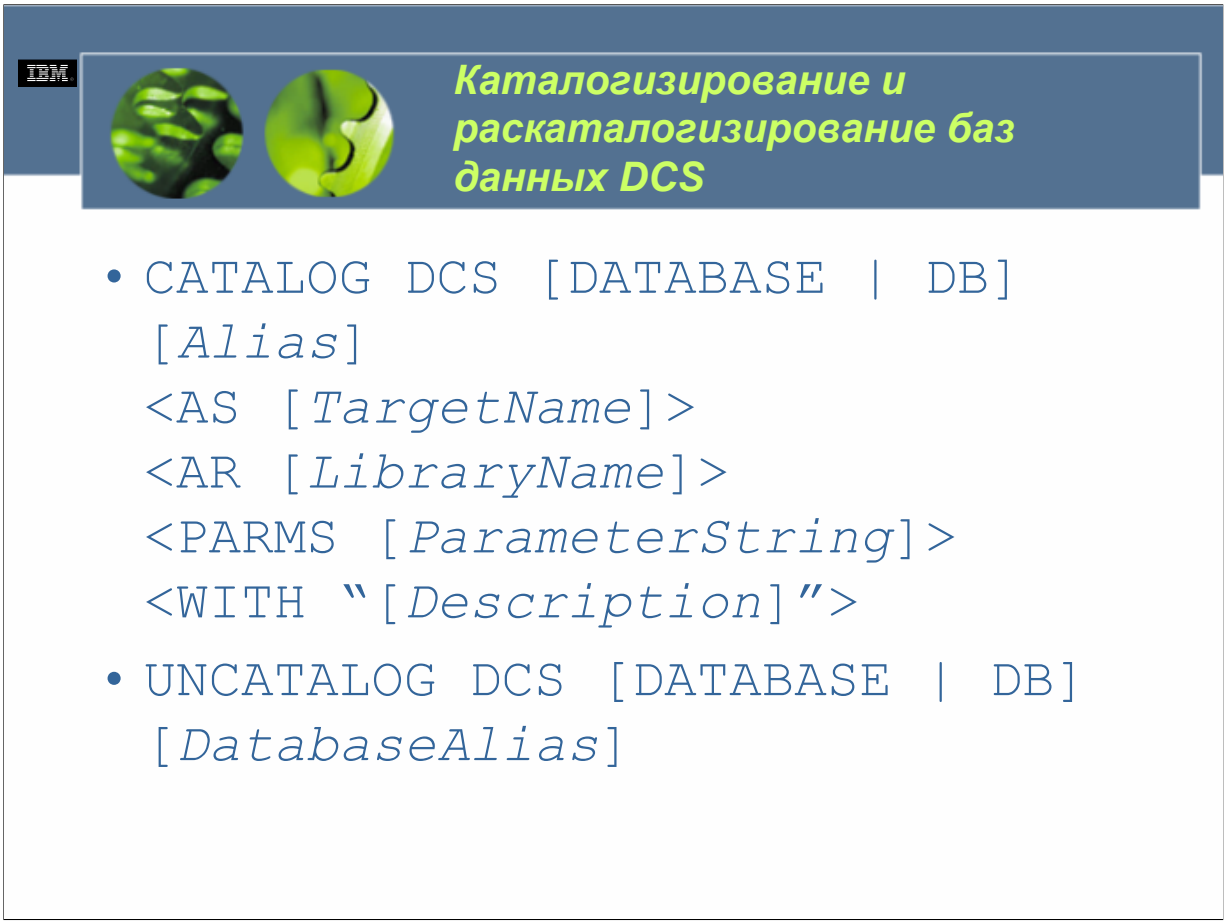
Независимо от того, какая команда CATALOG … NODE была использована для его каталогизирования, узел может быть раскаталогизирован посредством выполнения команды UNCATALOG NODE. Синтаксис для команды следующий:

UNCATALOG NODE [*NodeName*]

где:

*NodeName*Определяет алиас, связанный с раскаталогизируемым узлом.Так, если бы вам было нужно раскаталогизировать узел, который был каталогизирован в предыдущем примере, вы могли бы сделать это, выполнив команду UNCATALOG NODE, которая выглядит следующим образом:

UNCATALOG NODE DB2_TCPIP



WCF03.0

Помимо того факта, что не может использоваться ни центр управления, ни ассистент конфигурирования, процесс, используемый для каталогизирования базы данных Database Connection Services (DCS), очень сходен с процессом, используемым для каталогизирования обычной базы данных DB2 UDB. База данных DCS каталогизируется посредством выполнения команды CATALOG DCS DATABASE. Синтаксис для команды следующий:

CATALOG DCS [DATABASE | DB] [*Alias*]

<AS [*TargetName*]>

<AR [*LibraryName*]>

<PARMS [*ParameterString*]>

<WITH "[*Description*]">

где:

Alias определяет алиас для целевой базы данных для каталогизирования. Это имя должно совпадать с элементом в системном каталоге баз данных, связанном с удаленным узлом. *TargetName* определят имя целевого хоста или базы данных iSeries, которые каталогизируются. *LibraryName* указывает имя загруженной библиотеки реквестора прикладных программ (AR) и используется для доступа к удаленной базе данных, перечисленной в каталоге DCS. *ParameterString* определяет строку параметров, которые должны быть переданы AR при его вызове. Строка параметров должна быть заключена в двойные кавычки. *Description* комментарий, используемый для описания элемента базы данных, который будет сделан в каталоге DCS для каталогизируемой базы данных. Описание должно быть заключено в двойные кавычки (“). Так, если бы вам было нужно каталогизировать базу данных DCS с именем TEST_DB, которая является базой данных DB2 для z/OS, вы могли бы сделать это, выполнив команду CATALOG DCS DATABASE, которая выглядит примерно так:

CATALOG DCS DATABASE TEST_DB

AS DSN_DB

WITH “DB2 z/OS database”

Помните, что элемент для базы данных TEST_DB должен был бы также существовать в системном каталоге баз данных, прежде чем элемент, который эта команда создает в каталоге баз данных DCS, мог бы быть использован для соединения с базой данных z/OS.

Элементы в каталоге баз данных DCS можно удалить, выполнив команду UNCATALOG DCS DATABASE. Синтаксис для этой команды следующий:

UNCATALOG DCS [DATABASE | DB] [*DatabaseAlias*]

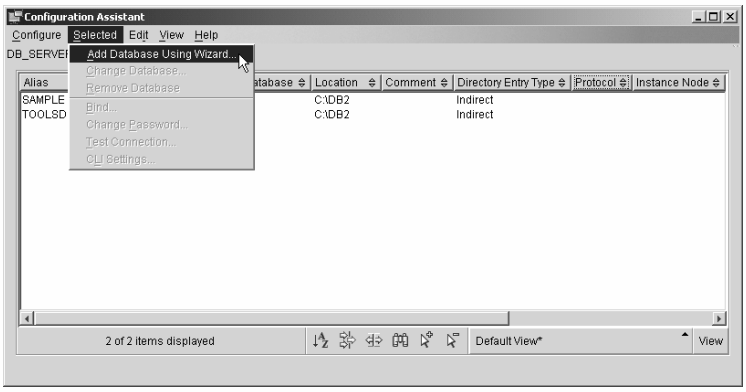
где:

DatabaseAlias обозначает алиас, связанный с раскаталогизируемой базой данных DCS. Таким образом, если бы вам было нужно раскаталогизировать базу данных DCS, каталогизированную в предыдущем примере, вы могли бы сделать это, выполнив команду UNCATALOG DCS DATABASE, которая выглядит следующим образом:

UNCATALOG DCS DATABASE TEST_DB

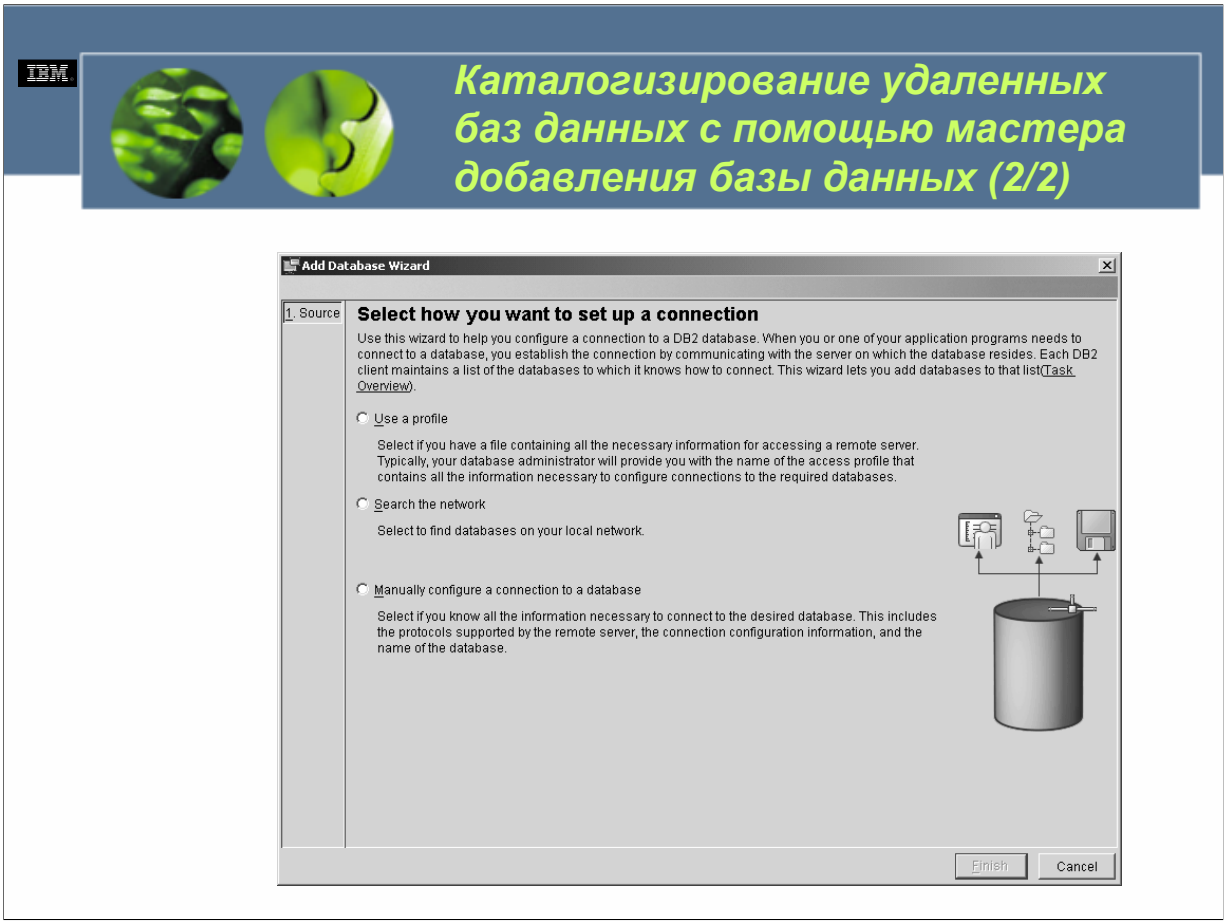
Каталогизирование удаленных баз данных с помощью мастера добавления базы данных (1/2)

- Автоматизированное конфигурирование с использованием профиля доступа
- Автоматизированное конфигурирование с использованием поиска в сети
- Ручное конфигурирование



WCF03.0

Как вы можете видеть, каталогизирование локальной базы данных является сравнительно простым процессом. Однако каталогизирование удаленной базы данных несколько более сложно, потому что как база данных, так и узел, на котором хранится база данных, должны каталогизироваться вместе. (Элемент в системном каталоге баз данных должен иметь соответствующий элемент в каталоге узлов). Вот почему был создан мастер добавления базы данных (не путать с диалогом добавления базы данных, который вызывается из центра управления). Этот инструмент, который запускается из ассистента конфигурирования, предназначен для сбора информации о свойствах баз данных, которые должны быть каталогизированы, затем использования этой информации для создания соответствующих элементов в каталогах баз данных и узлов.



WCF03.0

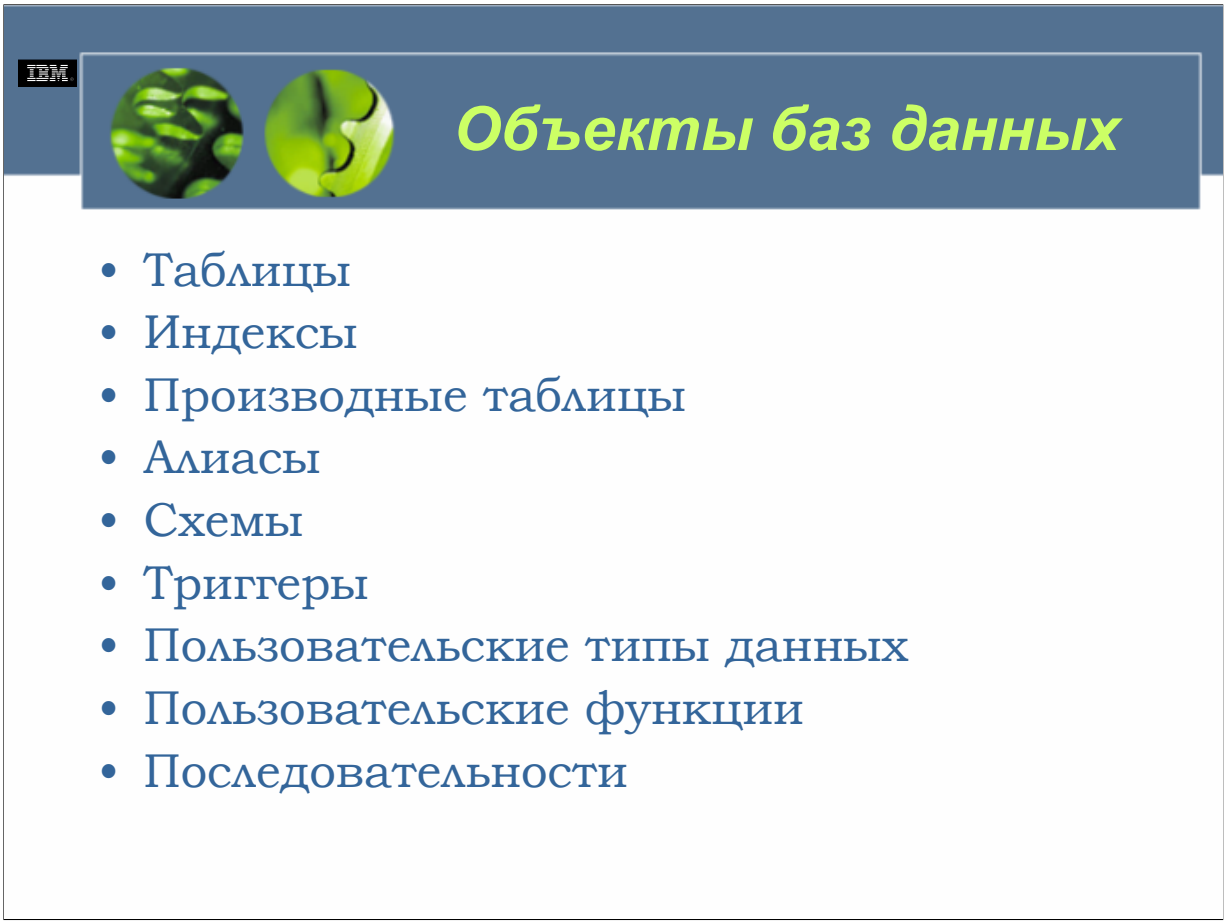
Первым, что вы заметите после запуска мастера добавления базы данных, является то, что имеется три метода, которые вы можете использовать для каталогизирования базы данных. Это:

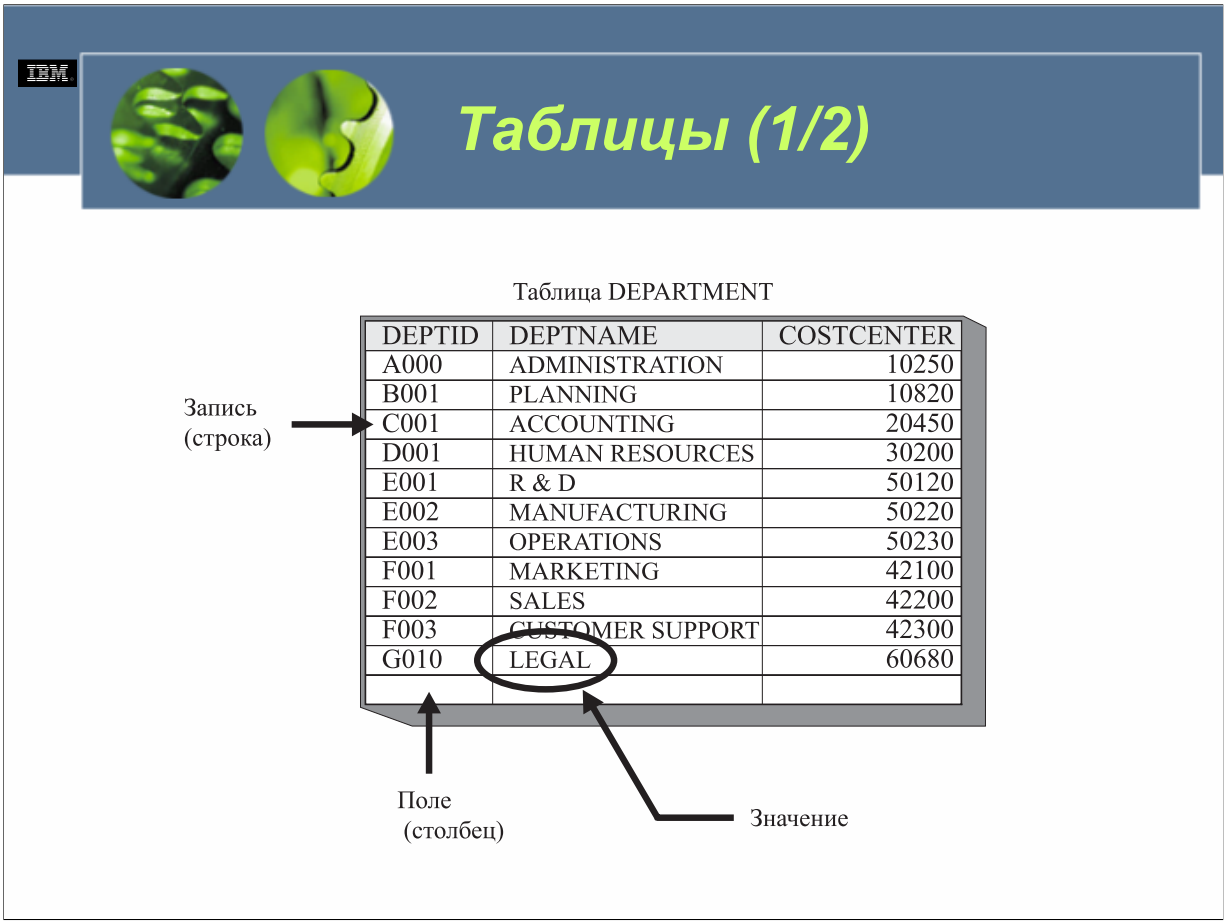
•Автоматизированное конфигурирование с использованием профиля доступа

•Автоматизированное конфигурирование с использованием поиска в сети

•Ручное конфигурирование

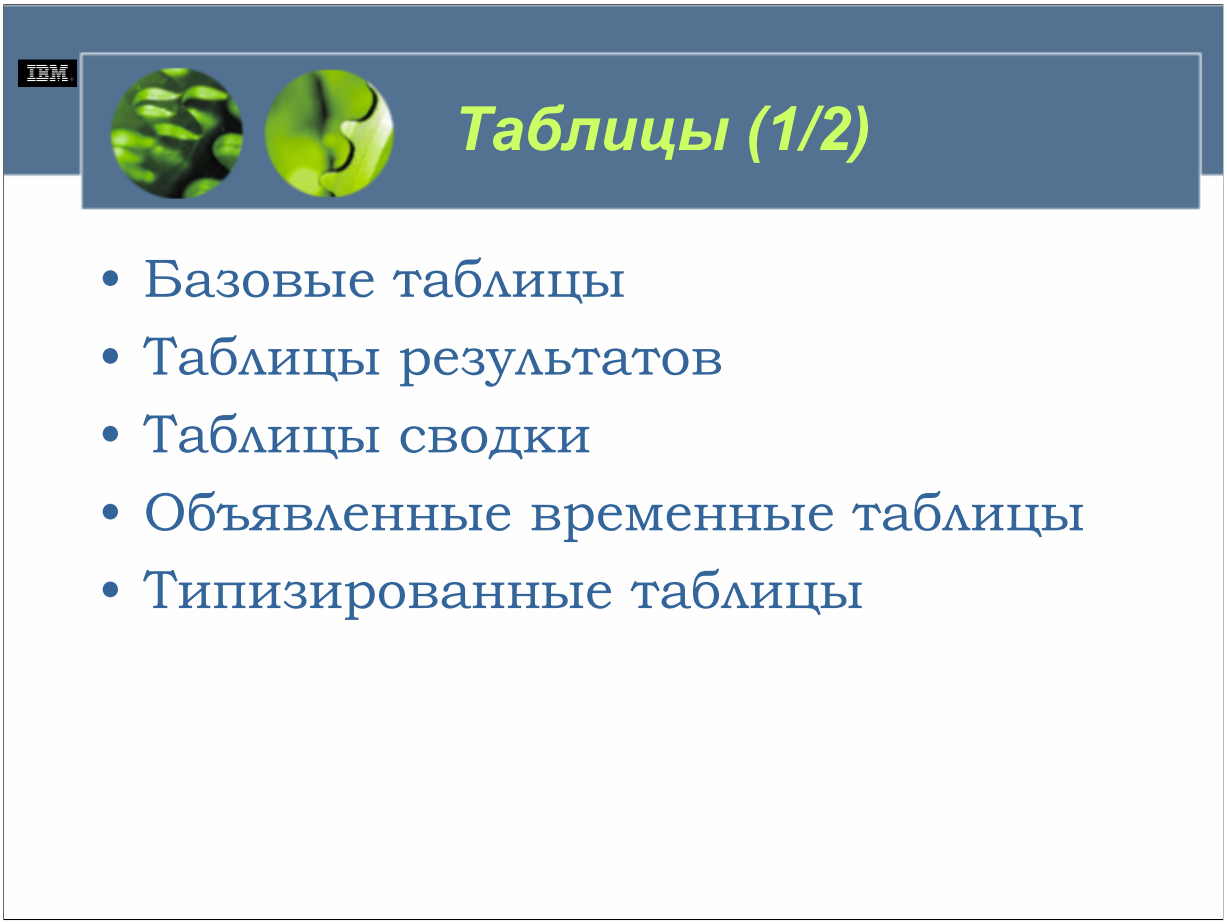
С точки зрения пользователя использование профиля доступа или поиска в сети является простейшим способом каталогизирования базы данных; ручное конфигурирование требует знания того, где размещена база данных физически, вместе со свойствами коммуникационного протокола, используемого как на клиенте, так и на сервере. Однако прежде, чем пользователь может воспользоваться преимуществами любой из доступных возможностей автоматизированного конфигурирования, администратор базы данных должен либо создать набор профилей доступа, либо настроить службы поиска DB2 на каждом соответствующем сервере DB2 UDB.





WCF03.0

Таблица является логическим объектом базы данных, который действует в качестве главного хранилища в базе данных. Таблицы представляют данные в виде набора неупорядоченных строк с фиксированным числом столбцов. Каждый столбец содержит данные одного и того же типа данных или одного из его подтипов, а каждая строка содержит набор значений для каждого доступного столбца. Обычно столбцы в таблице логически связаны, а между двумя или более таблицами могут быть определены дополнительные отношения. Представление хранения строки называется *записью*, представление хранения столбца называется *полем*, а каждое пересечение строки и столбца называется *значением*.



WCF03.0

В DB2 UDB доступны пять типов таблиц:

Базовые таблицы. Пользовательские таблицы, предназначенные для хранения постоянных данных пользователя.

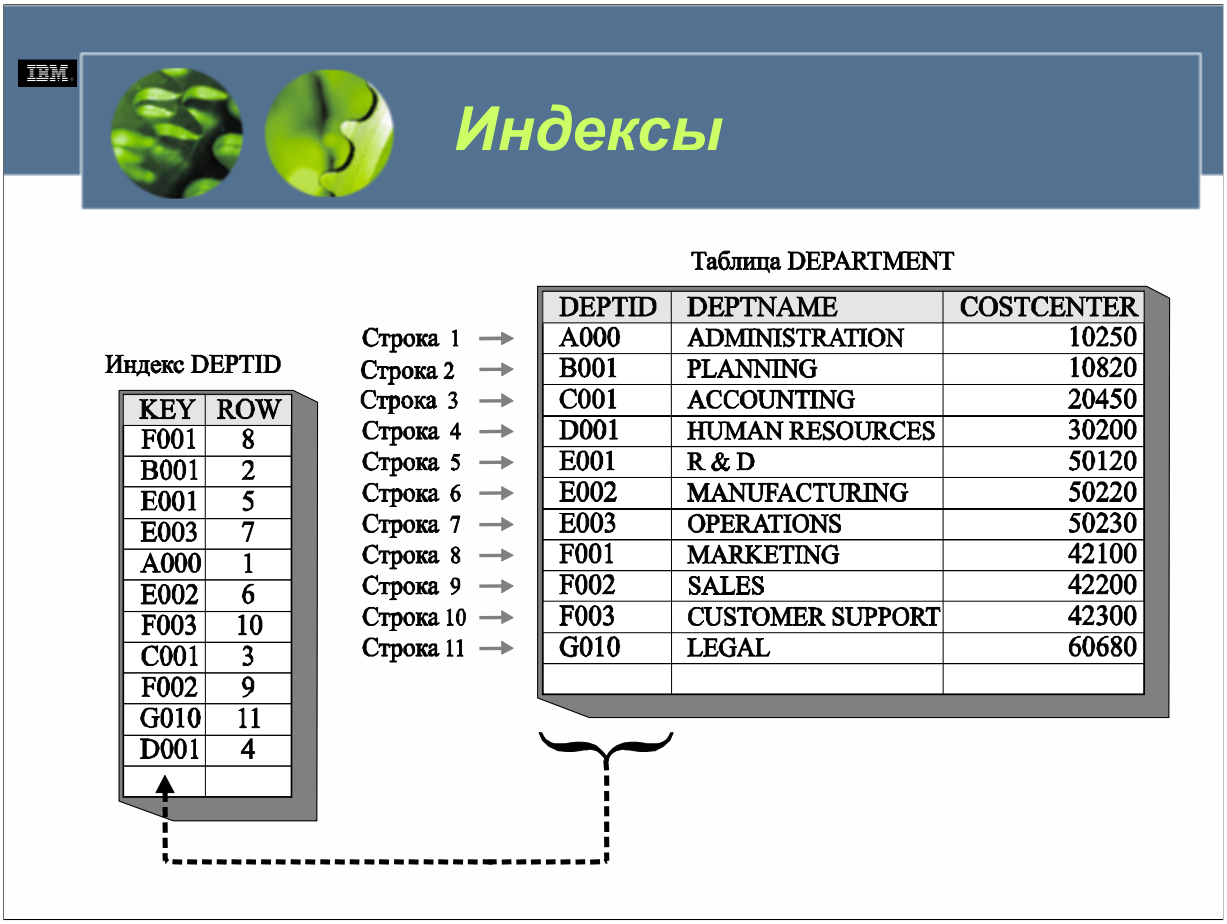
Таблицы результатов. Таблицы, определенные менеджером баз данных DB2 и заполненные строками, полученными из одной или нескольких базовых таблиц в ответ на запрос.

Таблицы сводки. Пользовательские таблицы, определения столбцов которых основаны на результатах запроса, который также используется для заполнения таблицы. Таблицы сводки используются для повышения производительности запроса.

Объявленные временные таблицы. Пользовательские таблицы, использующиеся для временного хранения непостоянной информации от имени одного приложения. Объявленные временные таблицы при необходимости создаются приложением явно и уничтожаются неявно, когда создавшее их приложение завершает свою последнюю связь с базой данных.

Типизированные таблицы. Пользовательские таблицы, определения столбцов которых основаны на атрибутах пользовательского структурированного типа данных.

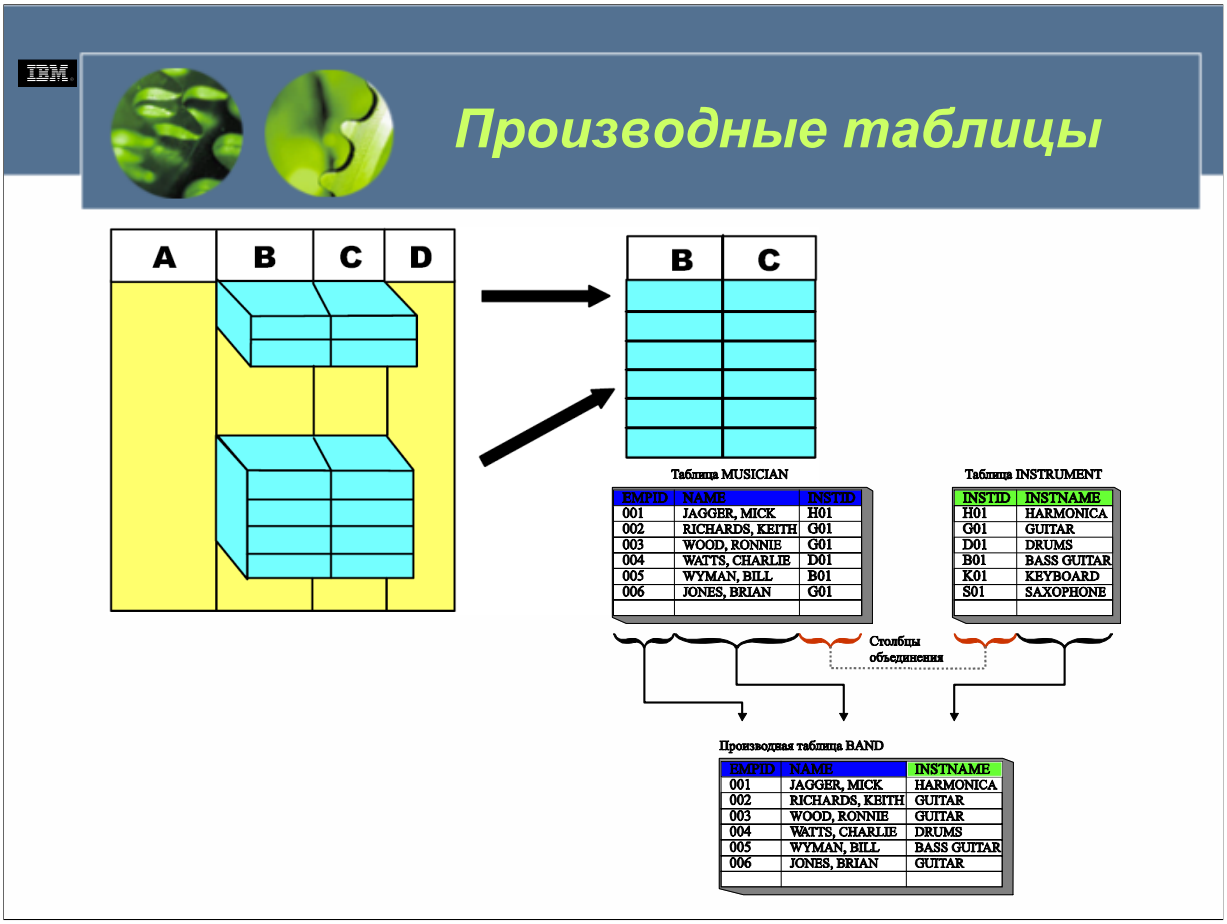
Данные, связанные с базовыми таблицами, таблицами сводки и типизированными таблицами, физически хранятся в табличных пространствах – в ходе процесса создания таблицы указывается действительное используемое табличное пространство.



WCF03.0

Индекс является объектом, содержащим упорядоченный набор указателей, которые ссылаются на строки в базовой таблице. Каждый индекс основан на одном или более столбцах в базовой таблице, на которую он указывает, однако они хранятся в качестве отдельных сущностей.

Индексы используются главным образом для обеспечения уникальности записи и чтобы помочь менеджеру баз данных DB2 быстро находить записи в ответ на запрос. Индексы могут также предоставлять больший параллелизм в многопользовательской среде – поскольку записи можно находить быстрее, запрошенные блокировки не нужно долго удерживать. Однако за эти преимущества приходится платить: каждый раз при использовании индексов необходимо дополнительное пространство хранения, и производительность может в действительности снижаться, когда в базовую таблицу добавляются новые данные и изменяются существующие. В обоих случаях проделываемые операции должны применяться как к базовой таблице, так и ко всем соответствующим индексам.

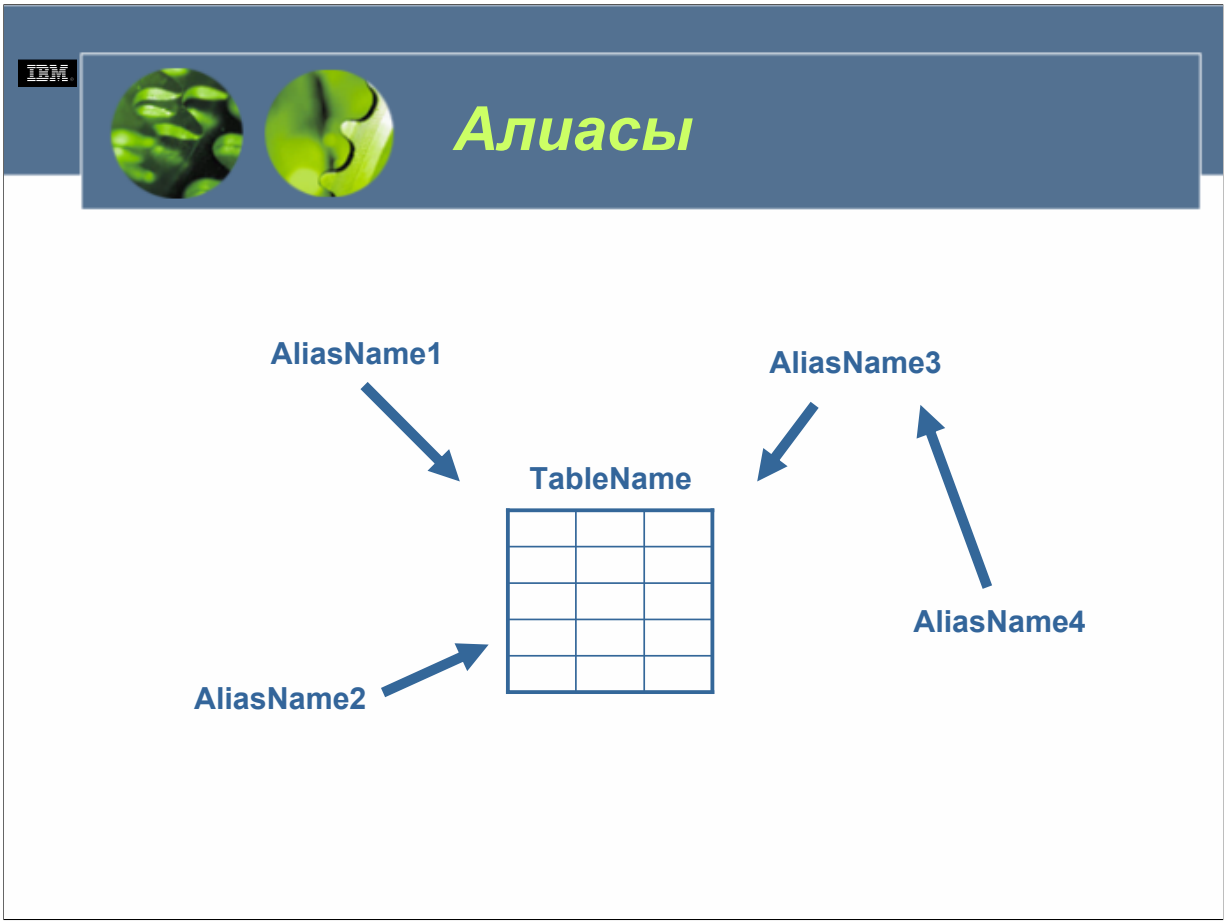


WCF03.0

Производные таблицы используются для предоставления другого способа просмотра данных, хранящихся в одной или нескольких базовых таблицах. В сущности производная таблица является именованной спецификацией таблицы результатов, которая заполняется каждый раз, когда на производную таблицу осуществляется ссылка в операторе SQL. Подобно базовым таблицам, производные таблицы можно представить составленными из столбцов и строк. И в большинстве случаев данные можно получить из производной таблицы таким же способом, как из обычной таблицы. Однако то, может или нет производная таблица использоваться в операциях вставки, изменения или удаления, зависит от того, как она была определена – производные таблицы могут определяться как с возможностью вставки, изменения , удаления, так и с доступом только для чтения.

Хотя производные таблицы выглядят похожими на базовые таблицы, они не содержат фактических данных. Вместо этого производные данные ссылаются на данные, хранящиеся в других базовых таблицах. В базе данных реально хранится лишь само определение производной таблицы. (Фактически, когда производятся изменения данных, представленных в производной таблице, реально изменяются данные, хранящиеся в базовых таблицах, на которые производная таблица ссылается).

Поскольку производные таблицы позволяют различным пользователям видеть различные представления одних и тех же данных, они часто используются для контроля за доступом к данным. Например, представьте, что у вас есть таблица, содержащая информацию обо всех служащих, работающих в определенной компании. Менеджерам можно было бы дать доступ к этим таблицам, используя производную таблицу, которая позволяет видеть информацию лишь о тех служащих, которые работают в их отделе. Членам отдела заработной платы, с другой стороны, можно было бы дать доступ к таблице, используя производную таблицу, которая позволяет им видеть лишь ту информацию, которая нужна для создания платежных ведомостей служащих. Обоим наборам пользователей предоставляется доступ к одной и той же таблице; однако поскольку каждый пользователь работает со своей производной таблицей, кажется, что они работают со своими собственными таблицами.



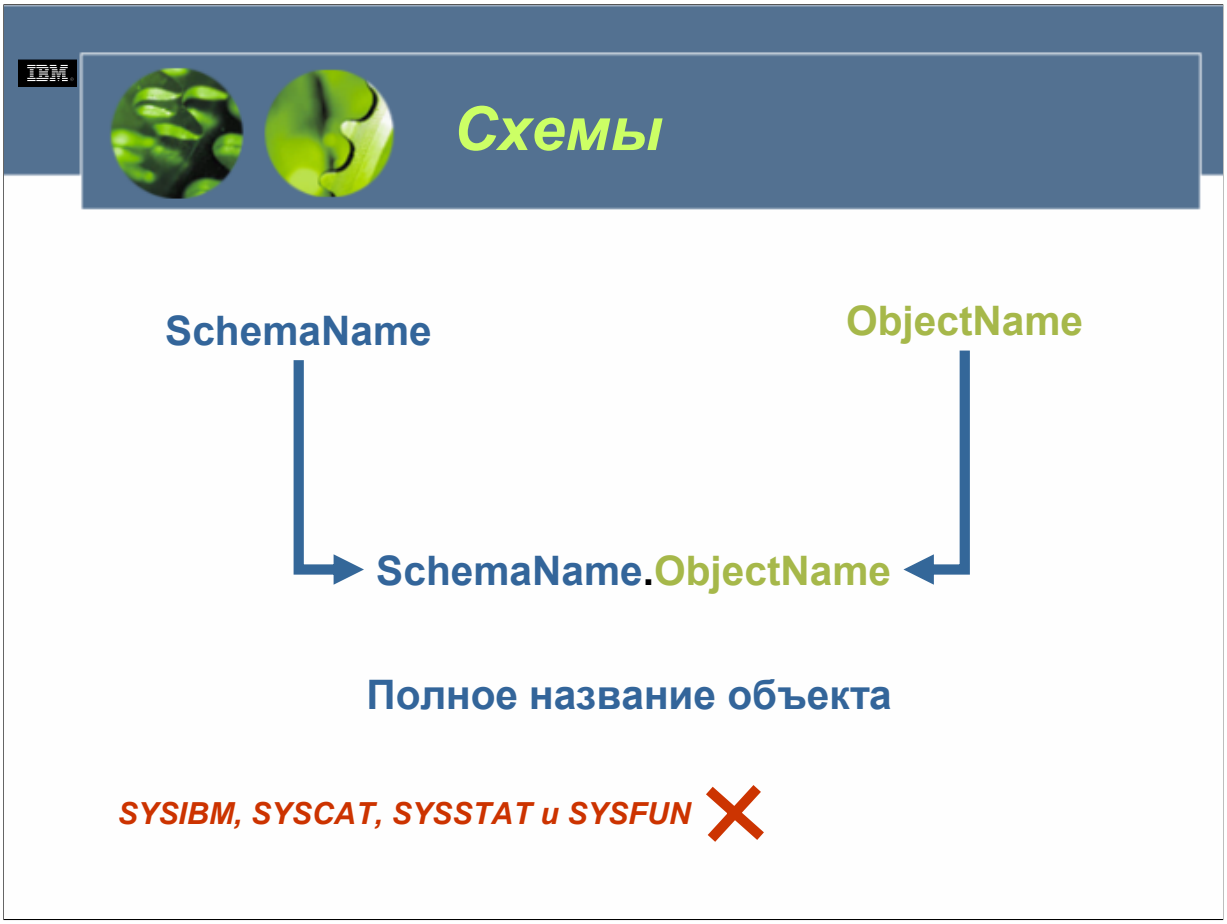
WCF03.0

Алиас является альтернативным именем для таблицы или производной таблицы. (Алиасы могут создаваться также для *псевдонимов*, которые ссылаются на таблицы данных или производные таблицы, расположенные в системах объединения). Алиасы могут использоваться для ссылки на любую таблицу или производную таблицу, на которую можно ссылаться по ее первичному имени. Однако алиас не может использоваться в каждом контексте, в котором может использоваться первичное имя таблицы или производной таблицы. Например, алиас не может использоваться в условии проверки проверочного ограничения, а также не может использоваться для ссылки на пользовательскую временную таблицу.

Аналогично таблицам и производным таблицам алиас может быть создан, уничтожен, и с ним могут быть связаны комментарии. Однако в отличие от таблиц (но аналогично производным таблицам) алиасы могут ссылаться на другие алиасы, используя процесс, известный как *сцепление (chaining)*.

Алиасы являются публично представляемыми именами, поэтому для их использования не требуется специальных полномочий или привилегий. Однако доступ к таблице или производной таблице, на которую ссылается алиас, по-прежнему имеет требования авторизации, связанные с этими типами объектов.

Так почему же может понадобиться использовать алиас вместо действительного имени таблицы/производной таблицы? Предположим, вам нужно разработать приложение, которое взаимодействует с таблицей с именем EMPLOYEES, которая находится в базе данных платежных ведомостей вашей компании. В ходе процесса разработки вам бы хотелось запустить приложение для тестовой таблицы EMPLOYEES; затем, когда разработка завершена, приложение будет нужно запустить для производственной таблицы EMPLOYEES. Используя алиас вместо имени базовой таблицы во всех ссылках на таблицы, делаемых приложением, вы можете быстро изменить приложение таким образом, чтобы оно работало с производственной таблицей EMPLOYEES вместо тестовой таблицы EMPLOYEES, просто изменив имя таблицы, на которое ссылается алиас.

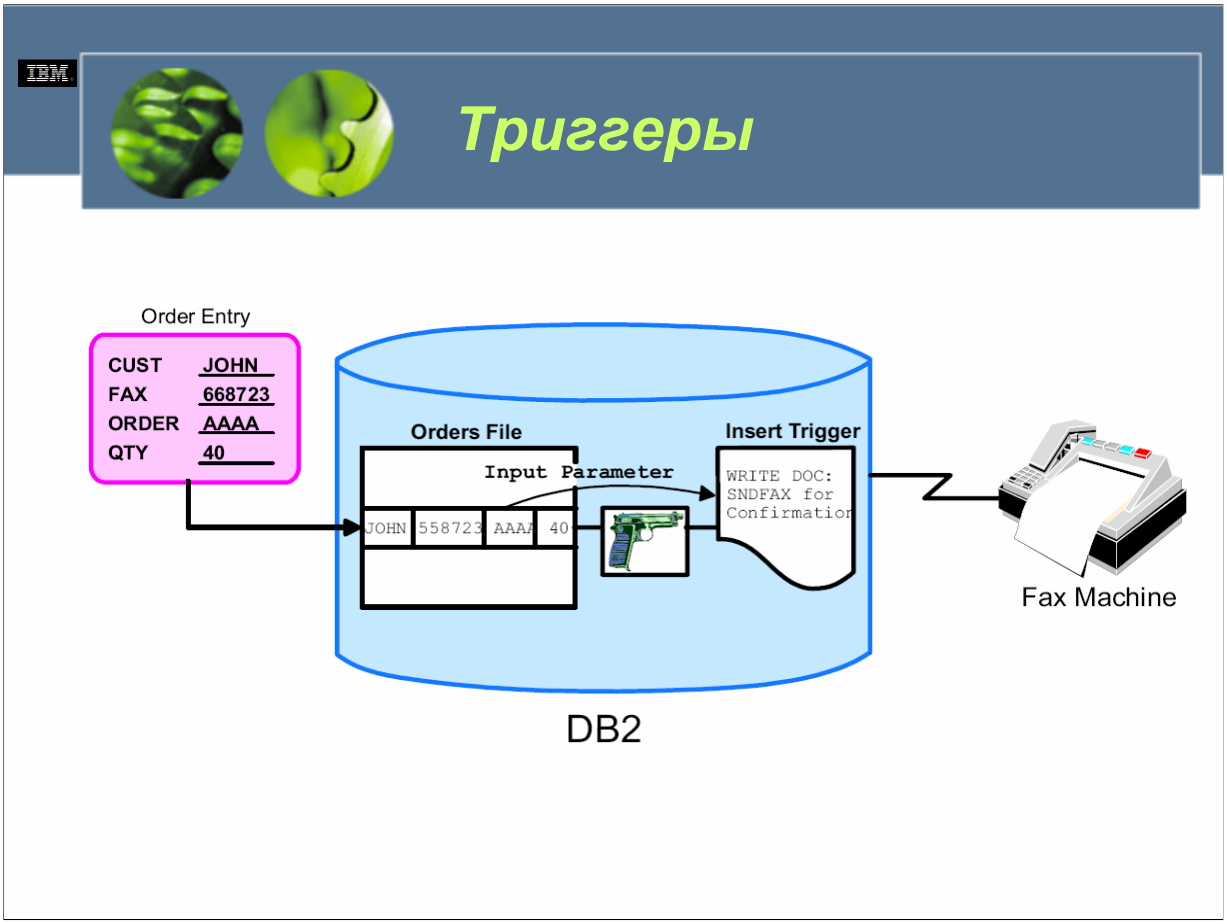


WCF03.0

Схемы являются объектами, которые используются для логической классификации и группировки других объектов в базе данных. Поскольку схемы сами являются объектами, у них есть связанные с ними привилегии, которые позволяют владельцу схемы контролировать то, какие пользователи могут создавать, изменять или уничтожать объекты внутри нее.

Большинство объектов в базе данных именуются с использованием соглашения по именованию, составленному из двух частей. Первая (левая) часть имени называется *именем схемы* или *спецификатором (qualifier)*, а вторая (правая) часть называется *именем объекта*. Синтаксически эти две части соединяются и отделяются точкой (например, HR.EMPLOYEE). Каждый раз, когда создается объект, специфицируемый именем схемы, он связывается со схемой, имя которой предоставлено. (Если имя схемы не указано, объект связывается со схемой по умолчанию, которой обычно является ID пользователя, создавшего объект). Некоторые имена схем, такие как имена, назначенные четырем схемам, неявно создаваемым при создании базы данных (SYSIBM, SYSCAT, SYSSTAT и SYSFUN), зарезервированы и не могут использоваться.

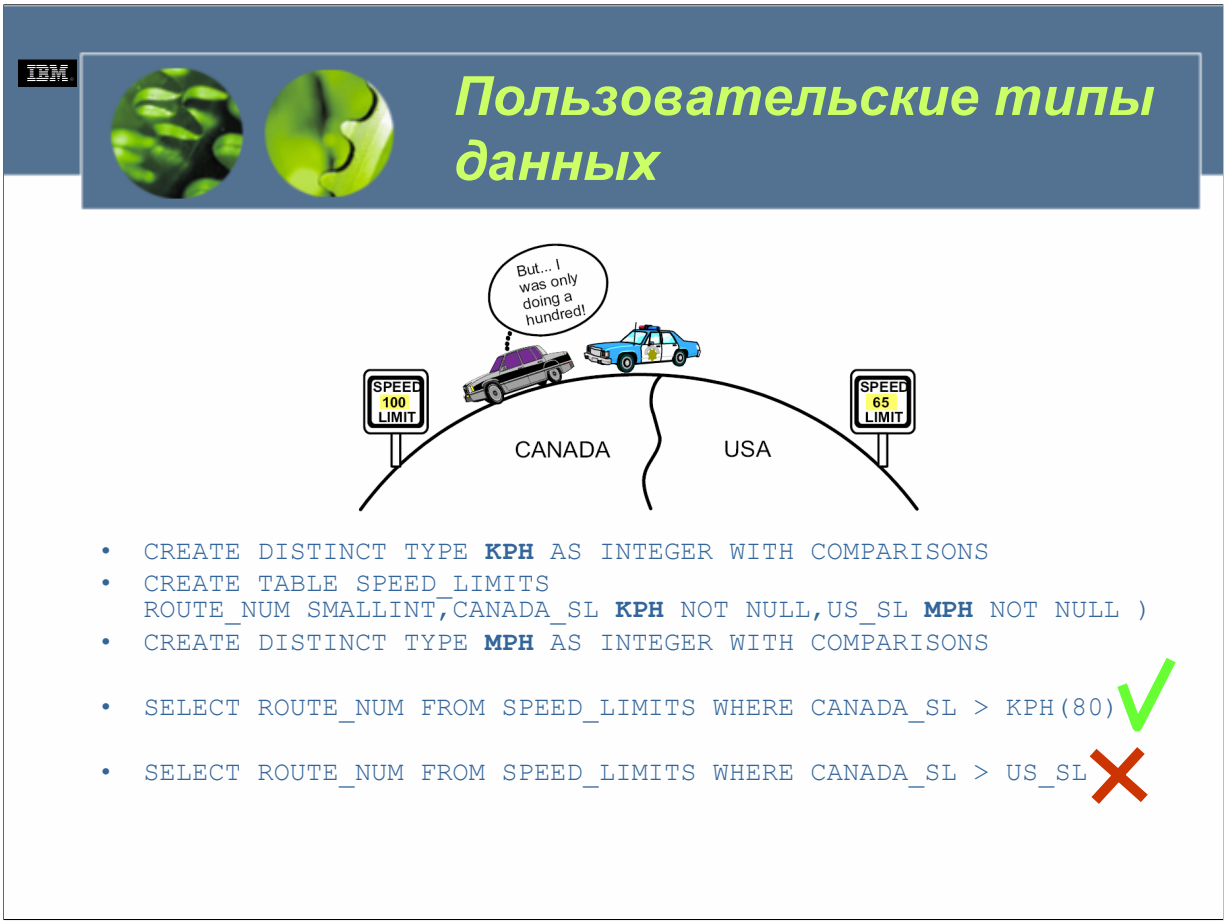
Схемы обычно создаются неявно, когда создаются другие объекты. (Когда создается новая база данных, всем пользователям предоставляются привилегии IMPLICIT_SCHEMA. Это позволяет любому пользователю создавать объекты в любой еще не существующей схеме). Однако схемы можно также создавать и явно.



WCF03.0

Триггер используется для определения набора действий, которые должны выполняться каждый раз, когда для определенной таблицы выполняется операция вставки, изменения или удаления. Триггеры могут использоваться вместе с реляционными и проверочными ограничениями для обеспечения правил целостности данных и бизнес-правил. (Правилом целостности данных могло бы быть то, что каждый раз, когда удаляется запись для служащего из таблицы, в которой хранится информация о сотрудниках, соответствующая запись будет удалена из таблицы, в которой хранится информация о платежных ведомостях. Бизнес-правилом могло бы быть то, что жалованье сотрудника не может быть увеличено более чем на 10 процентов). Триггеры могут также использоваться для изменения других таблиц, автоматического создания или преобразования значений для вводимых и/или обновляемых строк или вызова функций для выполнения специальных задач.

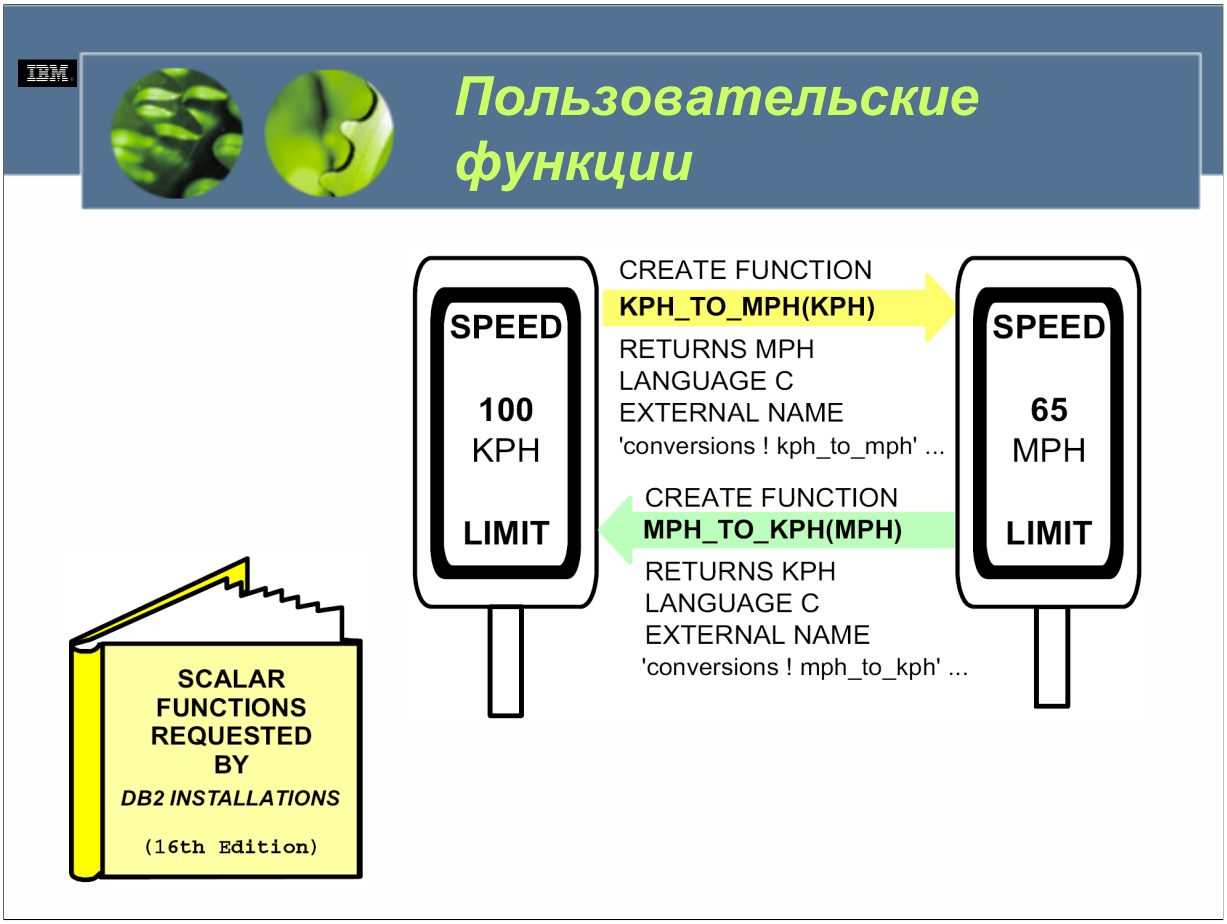
Путем использования триггеров логика, необходимая для осуществления таких бизнес-правил, может быть размещена непосредственно в базе данных, а приложения, которые работают с базой данных, могут концентрироваться исключительно на хранении данных, управлении данными и получении данных. И путем хранения логики, необходимой для обеспечения правил целостности данных и бизнес-правил непосредственно в базе данных, ее можно изменять при изменении правил целостности данных и бизнес-правил без необходимости перекодирования и перекомпилирования приложений.



WCF03.0

Как предполагает имя, пользовательские типы данных (UDT) являются типами данных, которые создаются (и именуются) пользователем базы данных. Пользовательский тип данных может быть особым типом данных, который разделяет общее представление с одним из встроенных типов данных, предоставленных DB2 UDB, или он может быть структурированным типом, состоящим из последовательности именованных атрибутов, у каждого из которых свой собственный тип данных. Структурированные типы данных могут также создаваться в виде подтипов других структурированных типов, определяя таким образом иерархию типов.

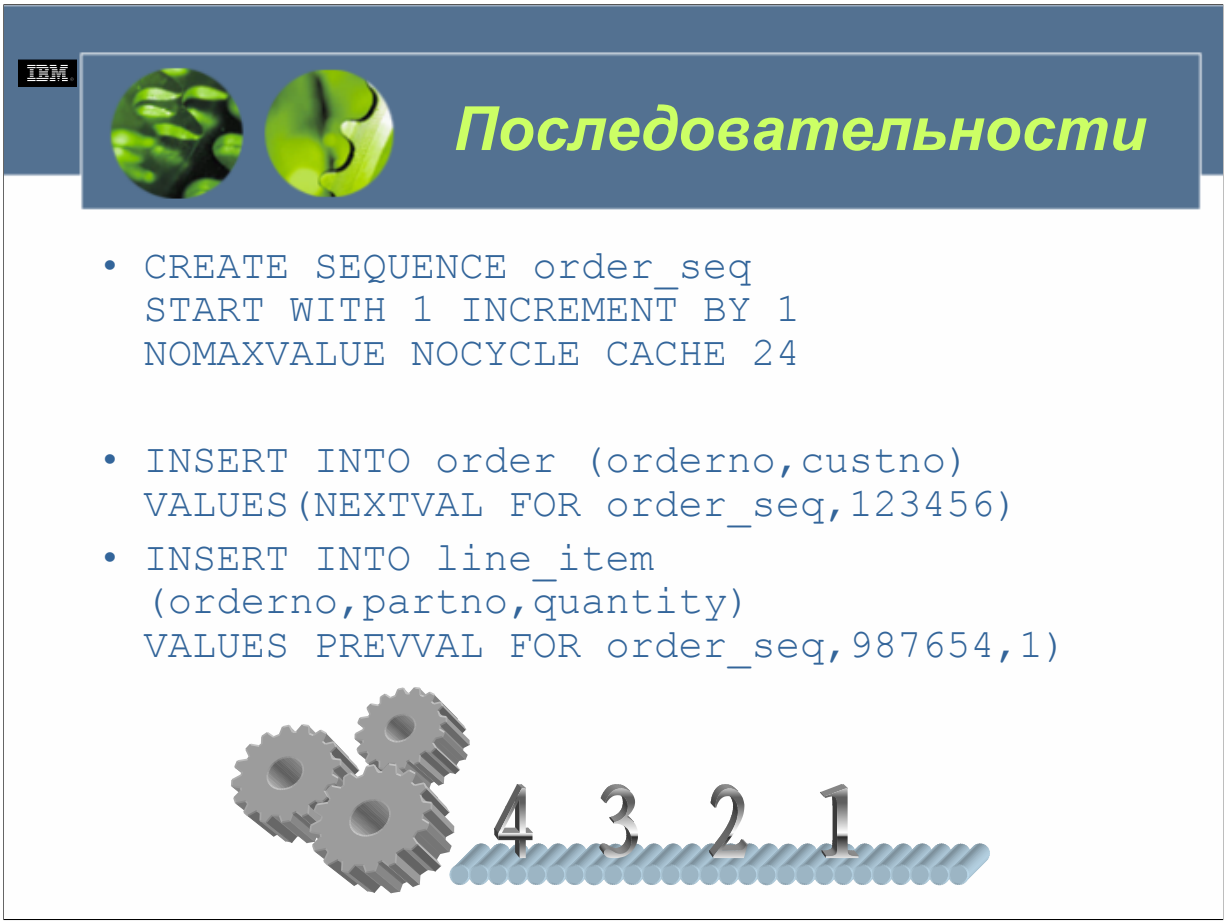
Пользовательские типы данных поддерживают строгую типизацию данных, что означает, что даже хотя они могут разделять одно и то же представление с другими встроенными или пользовательскими типами данных, значение одного пользовательского типа данных совместимо лишь со значениями того же самого типа (или других пользовательских типов данных в пределах одной и той же иерархии типов данных). В результате пользовательские типы данных не могут использоваться в качестве аргументов для большинства доступных встроенных функций. Вместо этого должны быть разработаны пользовательские функции (или методы) с аналогичными функциональными возможностями, когда необходимы эти возможности.



WCF03.0

Пользовательские функции (UDF) являются специальными объектами, использующимися для расширения и усиления поддержки, предоставляемой встроенными функциями, доступными в DB2 UDB. Подобно пользовательским типам данных, пользовательские функции (или методы) создаются и именуются пользователем базы данных. Пользовательская функция может быть внешней функцией, написанной на языке программирования высокого уровня, или функцией с источником, реализация которой наследуется от какой-нибудь другой уже существующей функции.

Аналогично всем встроенным функциям пользовательские функции подразделяются на *скалярные*, *функции столбца* или *табличные* по своей природе. Скалярные функции возвращают единственное значение и могут быть указаны в операторе SQL каждый раз, когда может быть использовано обычное выражение. (Примером скалярной функции является встроенная функция SUBSTR()). Функции столбца возвращают ответ с одним значением из множества аналогичных значений (столбца) и также могут указываться в операторе SQL каждый раз, когда может быть использовано обычное выражение. (Примером функции столбца является встроенная функция AVG()). Табличные функции возвращают таблицу оператору SQL, который ссылается на нее, и могут указываться лишь в условии FROM оператора SELECT. Табличные функции используются для работы с данными, которые не находятся в базе данных DB2 UDB и/или для преобразования таких данных в формат, напоминающий формат таблицы DB2. (Примером табличной функции является встроенная функция SNAPSHOT_TABLE).



WCF03.0

Последовательность является объектом, который используется для автоматической генерации значений данных. В отличие от столбца идентификации, который используется для генерации значений данных для определенного столбца в таблице, последовательность не привязана к какому-либо конкретному столбцу или определенной таблице. Вместо этого последовательность ведет себя подобно уникальному счетчику, который находится вне базы данных, с тем исключением, что он не представляет тех же самых проблем одновременности и производительности, которые возникают при использовании внешних счетчиков.

Все последовательности имеют следующие свойства:

•Генерируемые значения могут быть любого точного числового типа с масштабом ноль (SMALLINT, BIGINT, INTEGER и DECIMAL).

•Последовательные значения могут отличаться на любое определенное значение приращения. Значением приращения по умолчанию является 1.

•Значения счетчика имеют возможность восстановления. (Значения счетчиков реконструируются из журнала, когда необходимо восстановление).

•Генерируемые значения для повышения производительности могут кэшироваться.

Кроме того, последовательности генерируют значения одним из трех способов:

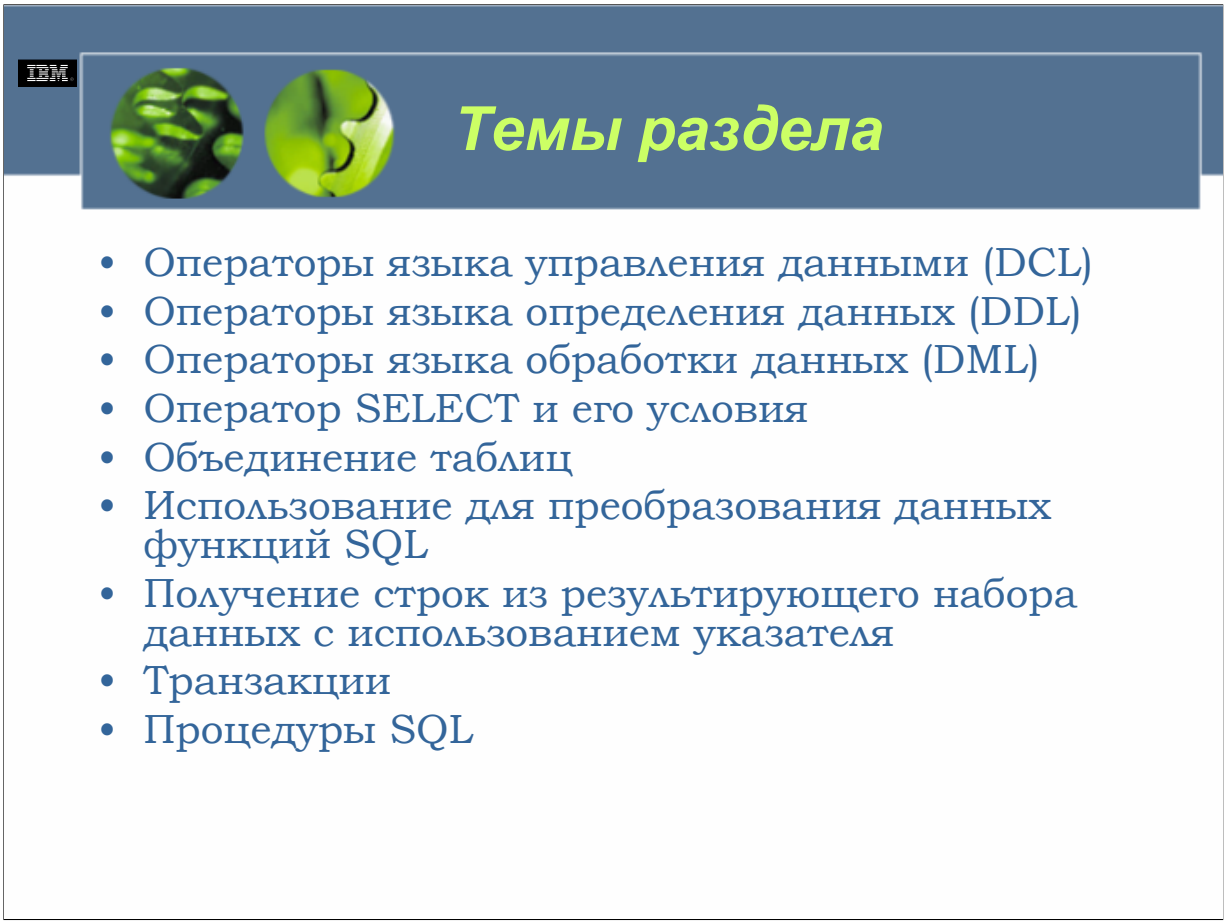
•Увеличением или уменьшением на определенное количество, без границ

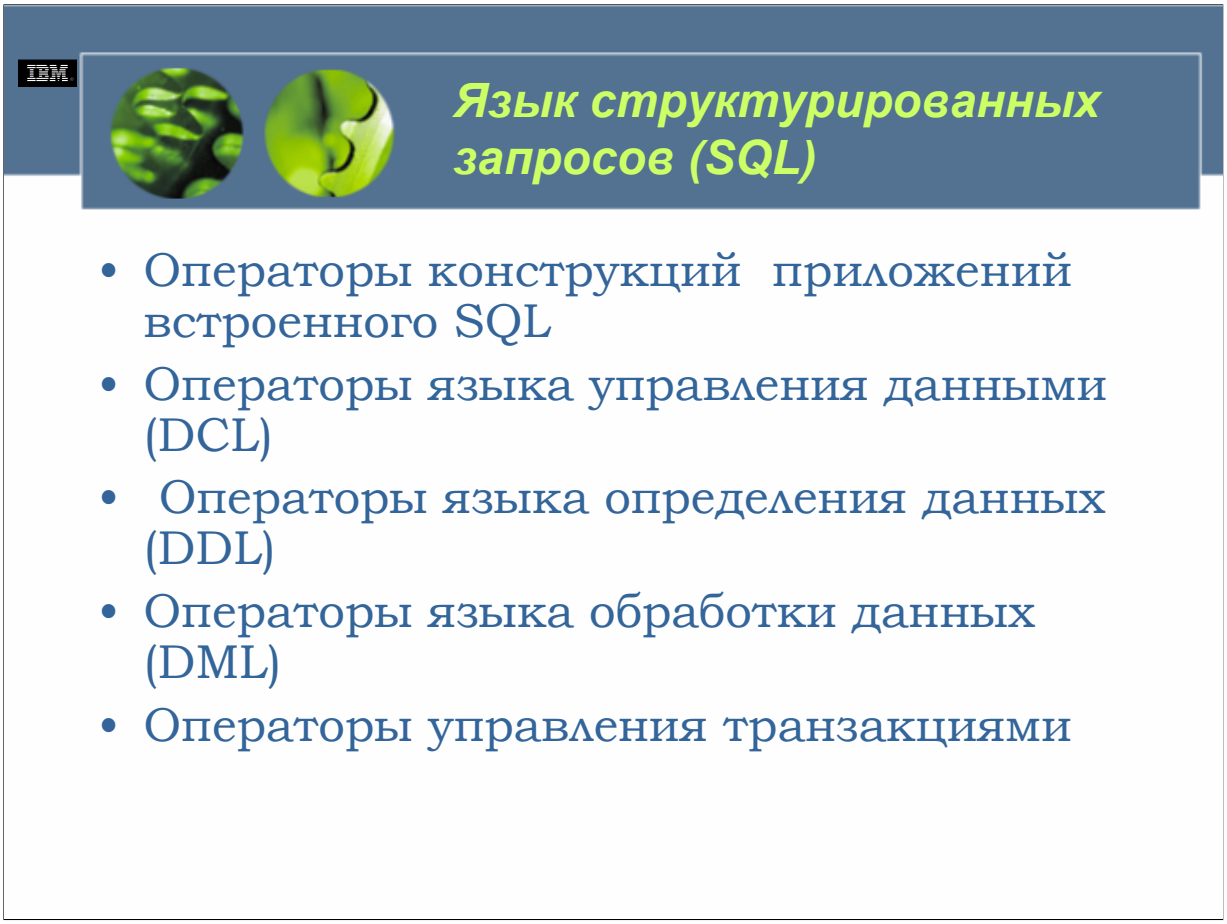
•Увеличением или уменьшением на определенное количество до определенного пользователем предела и остановкой

•Увеличением или уменьшением на определенное количество до определенного пользователем предела, затем циклическим переходом на начало и новым стартом.

Для содействия использованию последовательностей в операциях SQL доступно два выражения: PREVVAL и NEXTVAL. Выражение PREVVAL возвращает последнее сгенерированное значение для определенной последовательности, а выражение NEXTVAL возвращает следующее значение для указанной последовательности.

WCF03.0





WCF03.0

Язык структурированных запросов (SQL) является стандартизированным языком, использующимся для работы с объектами базы данных и данными, которые они содержат. Используя SQL, вы можете определять, изменять и удалять объекты базы данных, а также добавлять, удалять и получать значения данных. Одной из сильных сторон SQL является то, что он может использоваться различными способами: операторы SQL можно выполнять интерактивно, используя такие утилиты, как центр управления и процессор командной строки, их можно поместить прямо в сценарии оболочки UNIX или пакетные файлы Windows, и их можно внедрить в файлы с исходным кодом на языке программирования высокого уровня, которые прекомпилируются/компилируются для создания приложения баз данных. (Поскольку SQL является по назначению не процедурным, он не является действительным языком программирования; поэтому большинство приложений встроенного SQL строятся путем комбинирования контроля решений и последовательностей высокоуровневого языка программирования с возможностями хранения, обработки и получения данных SQL).

Подобно большинству других языков, SQL имеет определенный синтаксис и набор элементов языка. Большинство операторов SQL можно категоризировать в соответствии с функциями, для выполнения которых они были предназначены; операторы SQL обычно попадают в одну из следующих категорий:

•**Операторы конструкций приложений встроенного SQL.** Операторы SQL, использованные с единственной целью построения приложений встроенного SQL.

•**Операторы языка управления данными (DCL)**. Операторы SQL, предназначенные для предоставления и отзыва полномочий и привилегий.

•**Операторы языка определения данных (DDL)**. Операторы SQL, использующиеся для создания, изменения и удаления объектов баз данных.

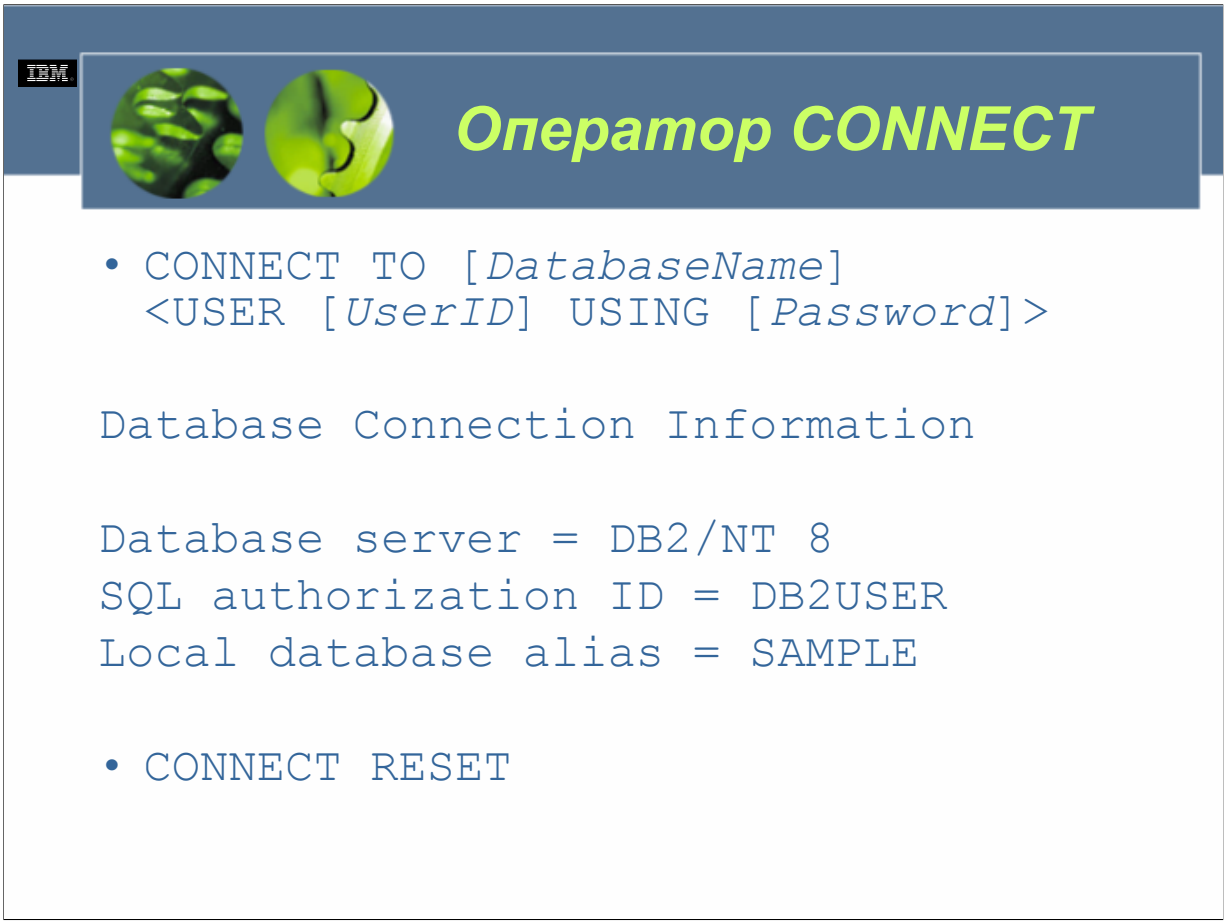
•**Операторы языка обработки данных (DML)**. Операторы SQL, использующиеся для сохранения данных и для получения или удаления данных из объектов баз данных.

•**Операторы управления транзакциями**. Операторы SQL, использующиеся для установления и завершения соединений с базами данных и активных транзакций.

Для того, чтобы сдать сертификационный экзамен по Основам семейства DB2 UDB V8.1 (экзамен 700), вам не нужно знакомство с операторами конструкций приложений встроенного SQL. Однако вы должны знать, как используются более общие операторы DCL, DDL и DML, и вы должны быть знакомы с доступными операторами управления транзакциями. Держа это в уме, данная глава будет фокусироваться на представлении вам наиболее типичных операторов SQL, использующихся для создания объектов баз данных и преобразования данных.

WCF03.0

Вы можете вспомнить, что в главе 3 «Безопасность» мы видели, что для контроля доступа к менеджеру баз данных DB2 для экземпляра, к одной или нескольким базам данных под управлением этого экземпляра и к различным объектам баз данных используются полномочия и привилегии. Пользователям разрешается работать лишь с теми объектами, для которых им были предоставлены соответствующие полномочия и привилегии; полномочия и привилегии, которыми они обладают, определяют также, что они могут и что не могут делать с объектами, к которым имеют доступ. Полномочия и привилегии предоставляются с использованием оператора SQL GRANT. Аналогичным образом, полномочия и привилегии отзываются с использованием оператора SQL REVOKE. Операторы GRANT и REVOKE, вместе с оператором SQL CONNECT, составляют основную массу доступных в DB2 UDB операторов языка управления данными (DCL).



WCF03.0

Прежде чем пользователь сможет получить доступ к данным, хранящимся в базе данных DB2 UDB (или сделать в этом отношении что-то еще с базой данных), он должен сначала установить соединение с этой базой данных. В некоторых случаях соединение с базой данных может быть установлено с использованием центра управления; однако в большинстве случаев соединение с базой данных устанавливается путем выполнения оператора SQL CONNECT. Базовый синтаксис для этого оператора следующий:

CONNECT TO [*DatabaseName*]

<USER [*UserID*] USING [*Password*]>

где:

DatabaseName указывает имя, связанное с базой данных, соединение с которой устанавливается. *UserID* указывает ID аутентификации (или ID пользователя), назначенный пользователю, пытающемуся установить соединение с базой данных. *Password* указывает пароль, назначенный пользователю, пытающемуся установить соединение с базой данных. Таким образом, для того, чтобы пользователю с ID аутентификации “db2user” и паролем “ibmdb2” установить соединение с базой данных SAMPLE, было бы нужно выполнить оператор CONNECT, который выглядит примерно следующим образом:

CONNECT TO SAMPLE USER db2user USING idbdb2

И как только оператор CONNECT успешно выполнен, вы могли бы увидеть примерно следующее сообщение:

Database Connection Information

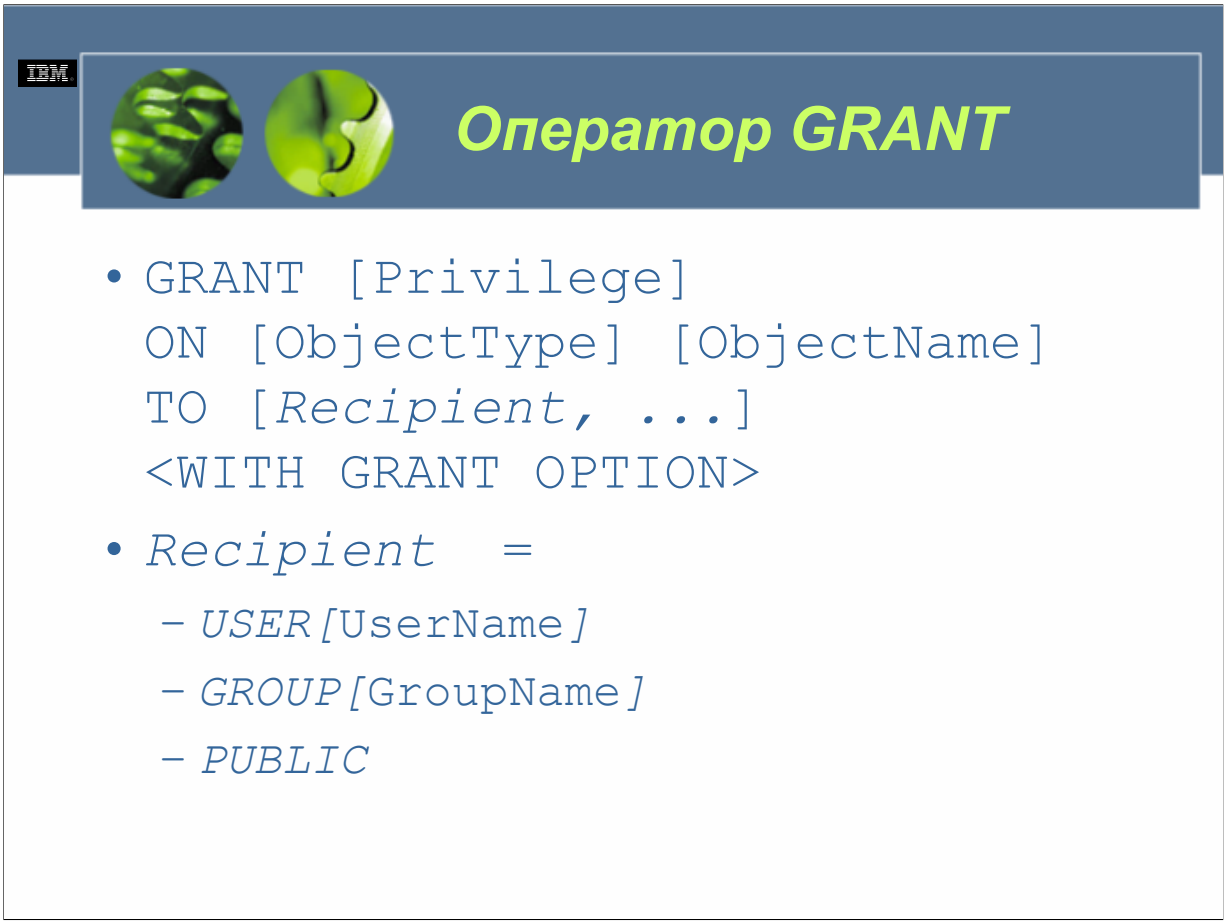
Database server = DB2/NT 8

SQL authorization ID = DB2USER

Local database alias = SAMPLE

После установления соединения с базой данных оно будет оставаться действительным до тех пор, пока не будет завершено явным образом или пока приложение, установившее соединение, не завершится. Соединения с базами данных можно завершить явным образом в любое время, выполнив специальную форму оператора CONNECT. Синтаксис для этой формы оператора CONNECT следующий:

CONNECT RESET



WCF03.0

В главе 3 «Безопасность» мы видели, что полномочия и привилегии можно предоставить отдельному пользователю или группе пользователей явным образом, выполнив оператор SQL GRANT. Доступно несколько разновидностей оператора GRANT, и соответствующая форма для использования определяется объектом базы данных, для которого должны быть предоставлены полномочия и привилегии. (Объекты, для которых могут быть предоставлены полномочия и привилегии, включают базы данных, схемы, табличные пространства, таблицы, индексы, производные таблицы, пакеты, процедуры, последовательности, серверы и псевдонимы). В общем базовый синтаксис для оператора GRANT выглядит примерно следующим образом:

GRANT [Privilege] ON [ObjectType] [ObjectName]

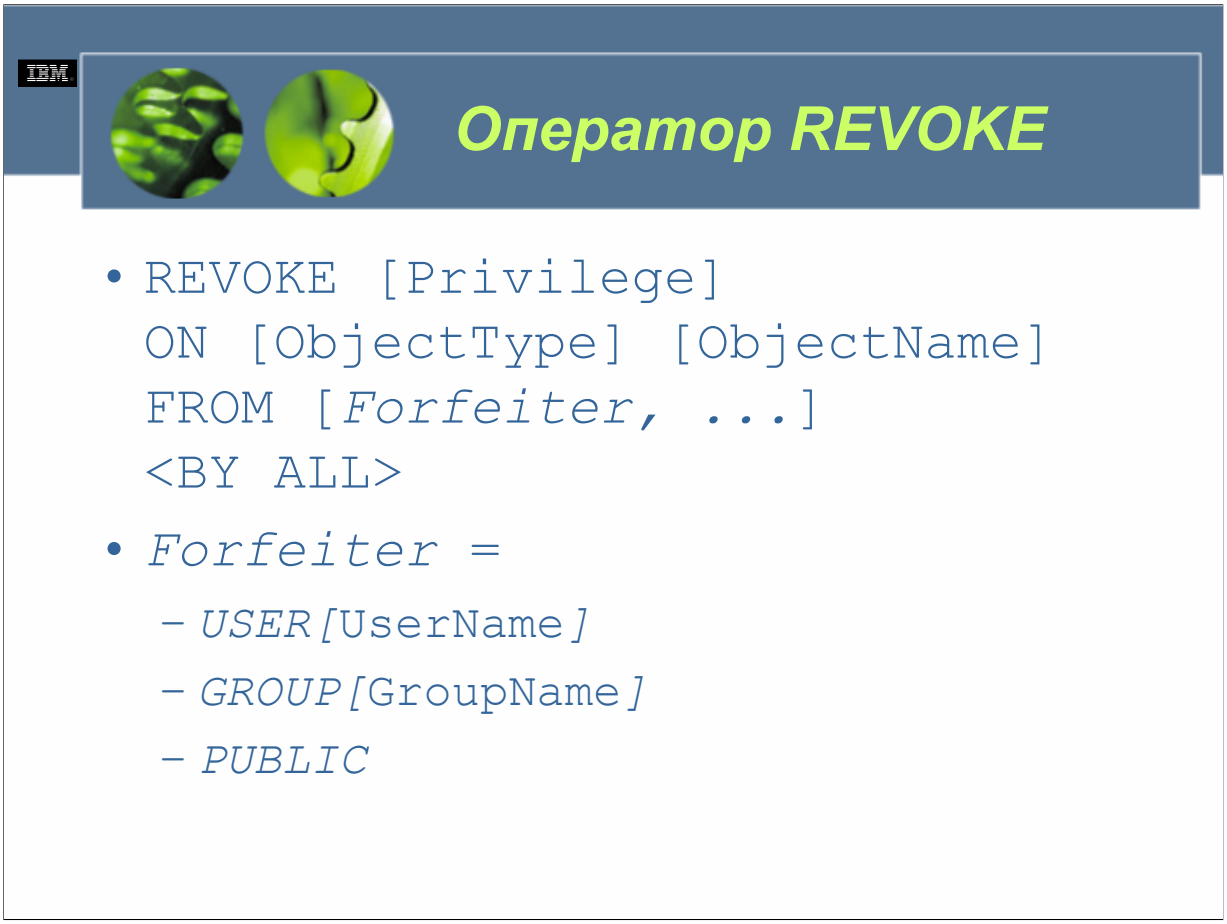
TO [*Recipient, ...*]

<WITH GRANT OPTION>

где:

Privilege обозначает одно или несколько полномочий или привилегий, которые должны быть предоставлены одному или нескольким пользователям и/или группам. *ObjectType* обозначает тип объекта, для которого должны быть предоставлены одно или несколько полномочий или привилегий. *ObjectName* указывает имя, связанное с объектом, для которого должны быть предоставлены одно или несколько полномочий или привилегий. *Recipient* указывает имена пользователей и/или групп, которые должны получить указанные привилегии объекта. Указанное для этого параметра значение может быть любой комбинацией следующих значений: USER[*UserName*] определяет конкретного пользователя, которому должны быть предоставлены указанные привилегии. GROUP[*GroupName*] определяет конкретную группу, которой должны быть предоставлены указанные привилегии. PUBLIC определяет, что указанные привилегии должны быть предоставлены специальной группе PUBLIC. (Все пользователи являются членами группы PUBLIC).Если с оператором указано условие WITH GRANT OPTION, пользователю и/или группе, получающей указанные привилегии, дается возможность предоставлять вновь полученные привилегии (за исключением привилегии CONTROL) другим пользователям.

За дополнительной информацией о доступных полномочиях и привилегиях и за примерами того, как можно использовать оператор SQL GRANT, обратитесь к главе 3 «Безопасность».



WCF03.0

Полномочия и привилегии отдельного пользователя или группы пользователей могут быть отозваны явным образом путем выполнения оператора SQL REVOKE. Как и в случае с оператором SQL GRANT, доступно несколько разновидностей оператора REVOKE, и подходящая для использования форма определяется объектом базы данных, для которого отзываются полномочия и привилегии. В общем базовый синтаксис для оператора REVOKE выглядит примерно следующим образом:

REVOKE [Privilege] ON [ObjectType] [ObjectName]

FROM [*Forfeiter, ...*]

<BY ALL>

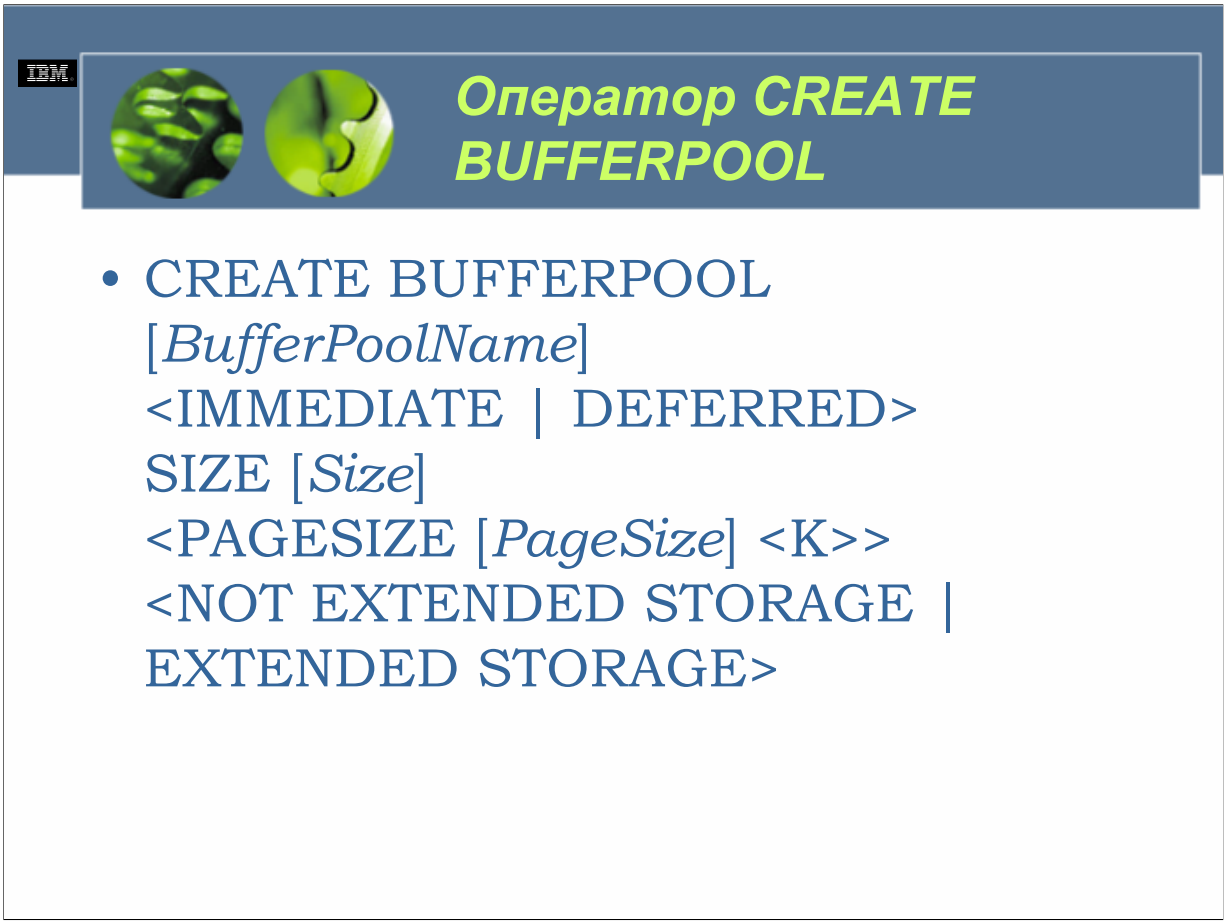
где:

*Privilege о*бозначает одно или несколько полномочий или привилегий, которые должны быть отобраны у одного или нескольких пользователей и/или групп. *ObjectType о*бозначает тип объекта, для которого отзывается одно или несколько полномочий или привилегий. *ObjectName у*казывает имя, связанное с объектом, для которого отзывается одно или несколько полномочий или привилегий. *Forfeiter у*казывает имена пользователей и/или групп, которые должны лишиться полномочия DBADM или указанных привилегий. Как и в случае оператора GRANT, указанное для этого параметра значение может быть любой комбинацией следующих значений: USER[*UserName*] определяет конкретного пользователя, у которого отбираются указанные привилегии. GROUP[*GroupName*] определяет конкретную группу, у которой отбираются указанные привилегии. PUBLIC определяет, что указанные привилегии должны быть отозваны у специальной группы PUBLIC. (Все пользователи являются членами группы PUBLIC). Условие BY ALL является необязательным и предусмотрено в качестве любезности для администраторов, знакомых с синтаксисом оператора SQL REVOKE DB2 для OS/390. Независимо от того, включено оно или нет, результат всегда будет одним и тем же – указанные привилегии будут отозваны у всех указанных пользователей и/или групп независимо от того, кто их первоначально предоставил.

Опять-таки за дополнительной информацией о доступных полномочиях и привилегиях и за примерами того, как можно использовать оператор REVOKE, обращайтесь к главе 3 «Безопасность».

WCF03.0

Когда база данных создается впервые, она не может использоваться для хранения данных, поскольку кроме таблиц системного каталога с доступом только для чтения и производных таблиц, которые создаются по умолчанию, не существует других объектов данных. Вот когда вступают в игру операторы языка определения данных (DDL). Операторы языка определения данных представляют собой набор операторов SQL, которые используются для определения и создания объектов в базе данных, которые будут использоваться как для хранения пользовательских данных, так и для повышения производительности доступа к данным. Различные формы операторов CREATE и ALTER вместе с оператором DROP составляют набор доступных в DB2 UDB операторов языка определения данных (DDL).



WCF03.0

Буферный пул является областью основной памяти, которая была выделена менеджеру баз данных DB2 с целью кэширования страниц данных таблиц и индексов по мере их чтения с диска. По умолчанию создается один буферный пул (с именем IBMDEFAULTBP) для определенной базы данных, когда она впервые создается. На платформах Linux и UNIX этот буферный пул имеет размер 1000 4Кб (килобайтных) страниц; на платформах Windows этот буферный пул имеет размер 250 4Кб страниц. Можно создать дополнительные буферные пулы, выполнив оператор SQL CREATE BUFFERPOOL. Базовый синтаксис для этого оператора следующий:

CREATE BUFFERPOOL [*BufferPoolName*]

<IMMEDIATE | DEFERRED>

SIZE [*Size*]

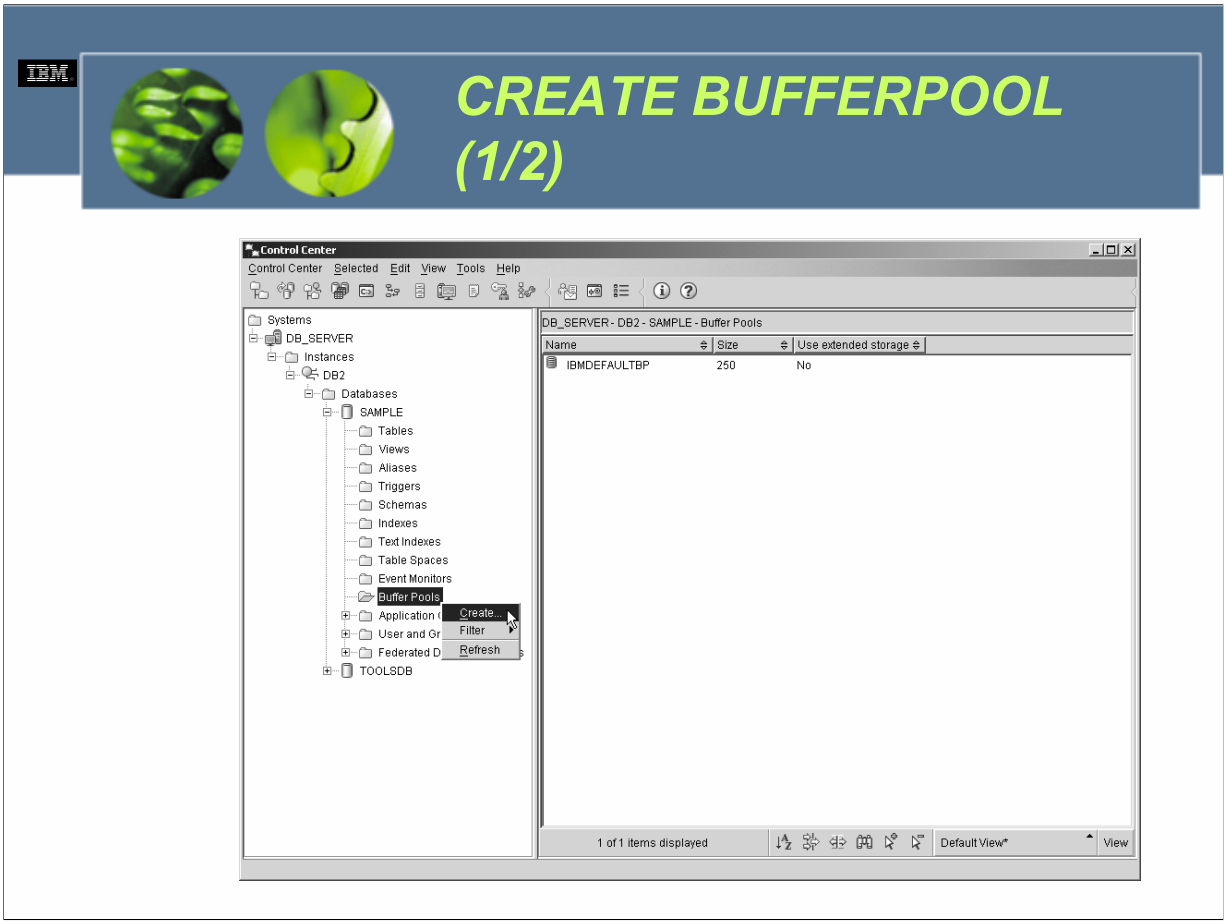
<PAGESIZE [*PageSize*] <K>>

<NOT EXTENDED STORAGE | EXTENDED STORAGE>

где:

BufferPoolName определяет имя, которое должно быть назначено создаваемому буферному пулу. *Size* определяет число страниц (с размером *PageSize*), которые должны быть выделены создаваемому буферному пулу. *PageSize* определяет размер, который должна иметь каждая страница, используемая создаваемым буферным пулом. (Действительные размеры включают 4096, 8192, 16384 или 32768 байтов – если предоставлен суффикс К (Килобайты), этот параметр должен равняться 4, 8, 16 или 32). Если не указано иное, страницы, используемые буферными пулами, имеют размер 4 Кб. (Важно отметить, что размер страницы, используемый буферным пулом, определяет, какие табличные пространства могут с ним использоваться; буферный пул может использоваться лишь табличным пространством с соответствующим размером страницы).

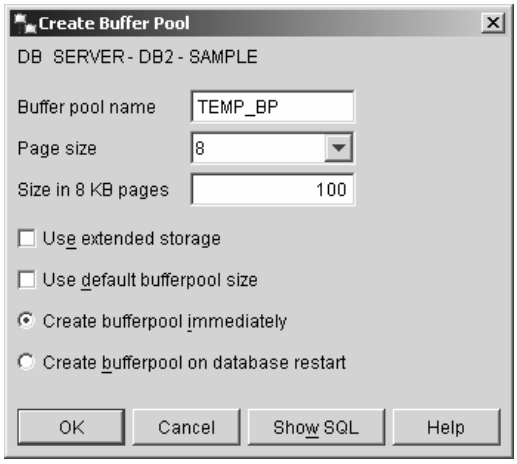
Если с оператором CREATE BUFFERPOOL указано условие IMMEDIATE, буферный пул будет создан немедленно, если достаточно памяти, если нет, будет сгенерировано предупреждающее сообщение, и процесс создания буферного пула будет протекать так, как если бы было указано условие DEFERRED. Если указано условие DEFERRED, буферный пул не будет создан до тех пор, пока не будут завершены все соединения с базой данных, для которой должен быть создан буферный пул. Если не указано ни одно из этих условий, буферный пул не будет создан до тех пор, пока не будут завершены все соединения с базой данных, для которой должен быть создан буферный пул.

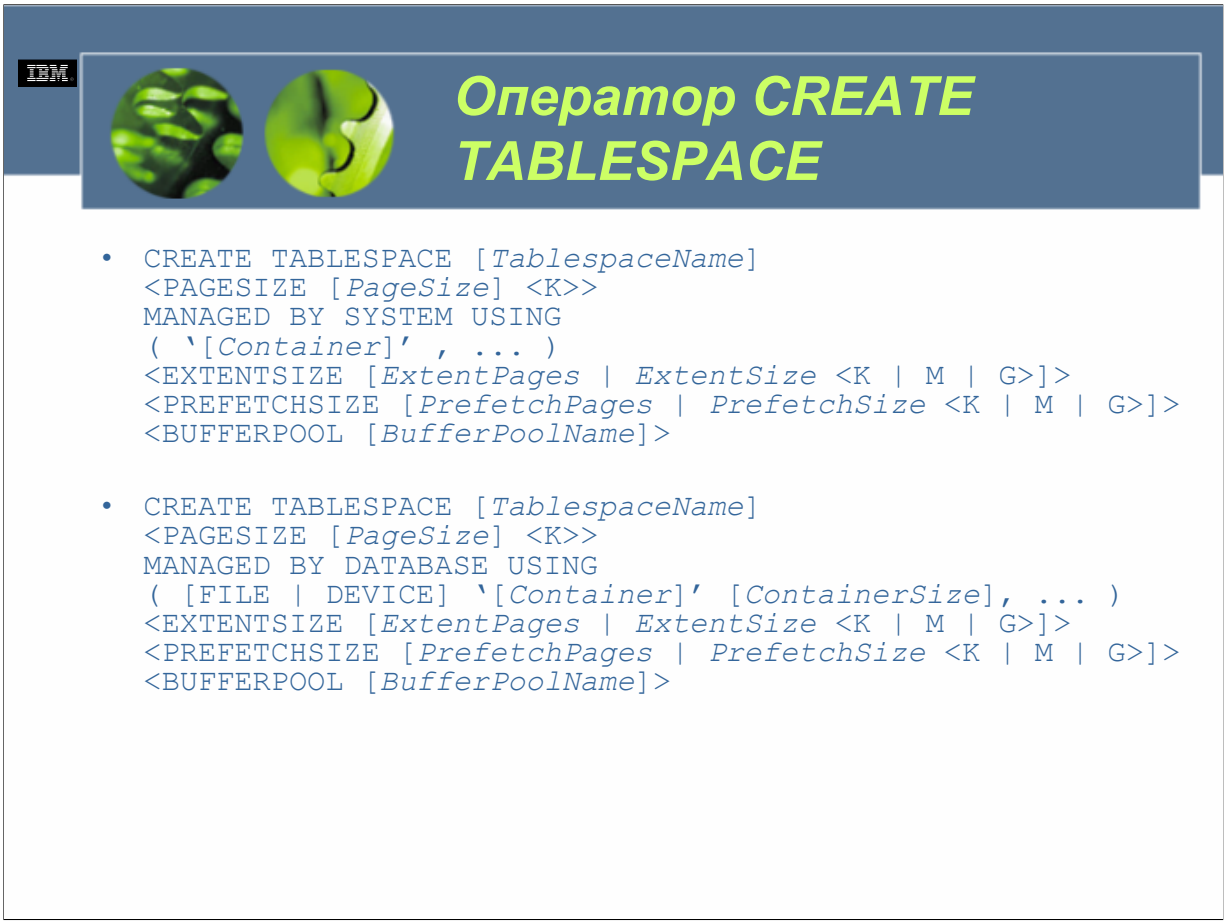


WCF03.0

Буферные пулы можно также создавать, используя диалог Создание буферного пула (Create Buffer Pool), который можно вызвать, выбрав соответствующее действие из меню *Буферные пулы (Buffer Pools)* в центре управления.

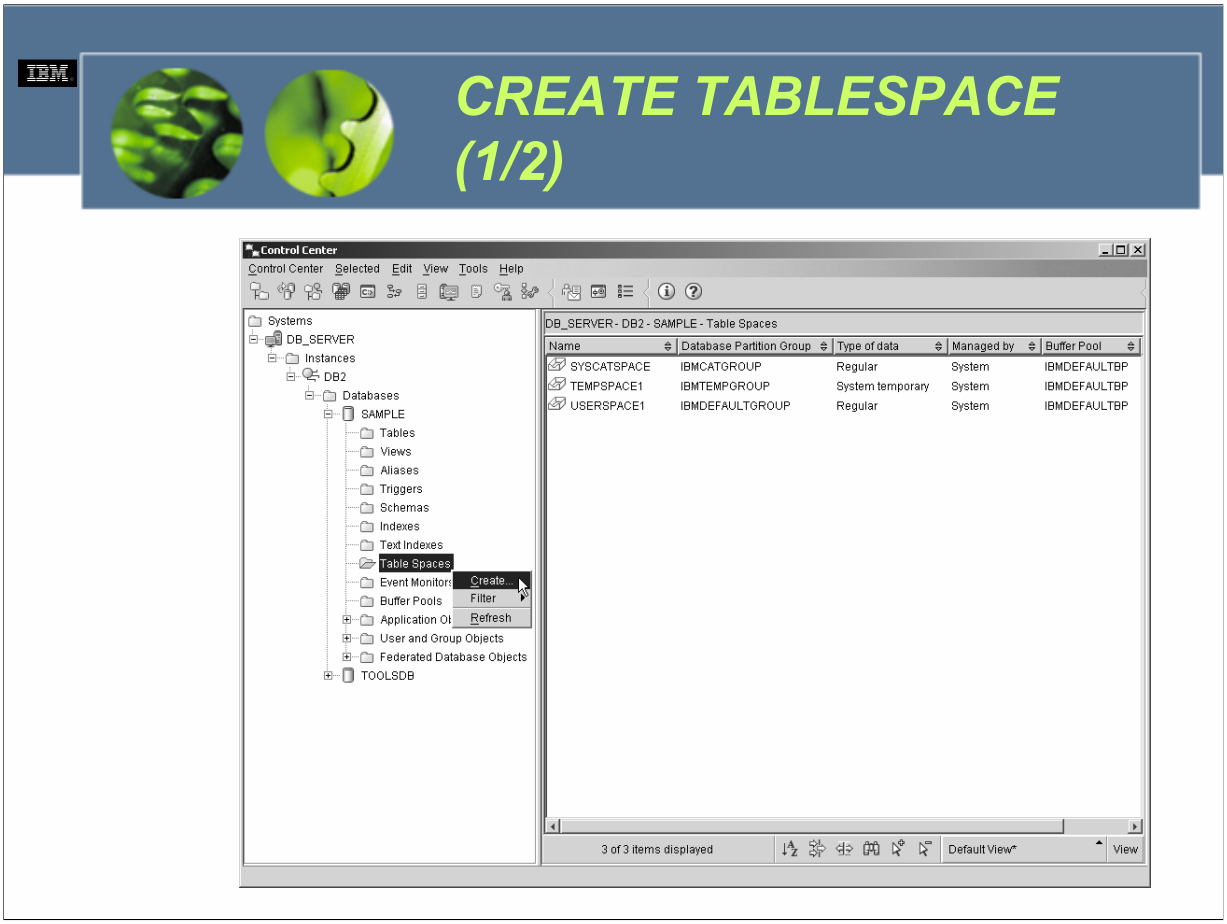
CREATE BUFFERPOOL (2/2)





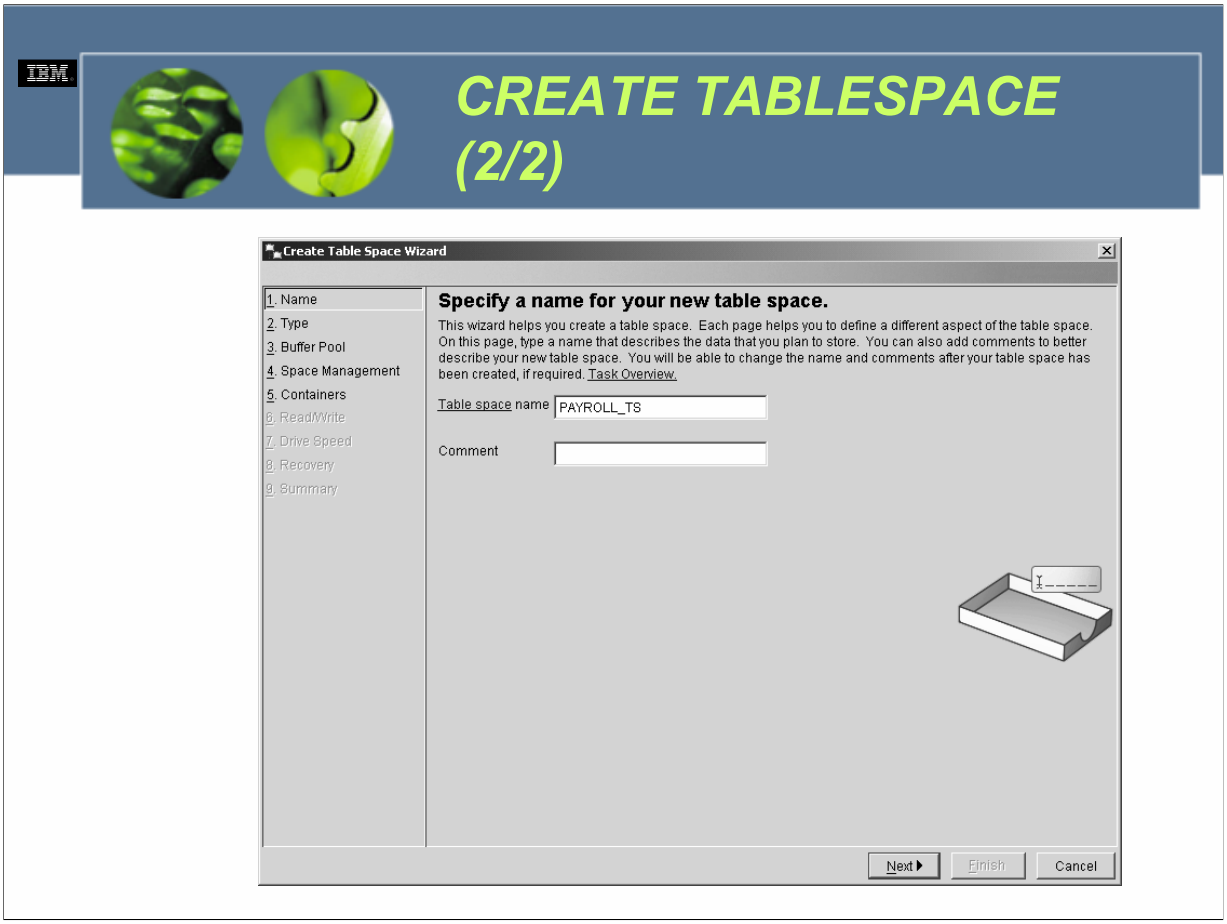
WCF03.0

TablespaceName определяет имя, которое должно быть назначено создаваемому табличному пространству. *PageSize* определяет размер, какой должна иметь каждая страница, используемая создаваемым табличным пространством. (Действительные значения включают 4096, 8192, 16384 или 32768 байтов – если указан суффикс K (Килобайты), этот параметр должен равняться 4, 8, 16 или 32). Если не указано иное, страницы, используемые табличным пространством, имеют размер 4 Кб. *Container* обозначает один или несколько контейнеров, которые должны использоваться для хранения данных, связанных с создаваемым табличным пространством. (Если используются несколько контейнеров, данные записываются в каждый из них на манер карусели, по одному экстенту за раз). *ContainerSize* определяет число страниц (размера *PageSize*), которые должны храниться в указанном контейнере табличного пространства. *ExtentPages* определяет число страниц данных, которые должны быть записаны в один контейнер табличного пространства перед использованием другого контейнера. *ExtentSize* определяет количество данных, которое должно быть записано в один контейнер табличного пространства до использования другого контейнера. Указанное для данного параметра значение рассматривается как общее число байтов, если также не указана буква K (для килобайтов), M (для мегабайтов) или G (для гигабайтов). (Если указано значение *ExtentSize*, оно преобразуется в значение *ExtentPages* с использованием предоставленного значения *PageSize*). *PrefetchPages* определяет число страниц данных, которые должны быть прочитаны из табличного пространства при выполнении предварительной выборки данных (предварительная выборка позволяет прочитывать данные, необходимые запросу, до того, как на них будет сделана ссылка, так что запрос тратит меньше времени в ожидании ввода/вывода). *PrefetchSize* определяет количество данных, которые должны быть прочитаны из табличного пространства при выполнении предварительной выборки данных. Указанное для данного параметра значение рассматривается, как общее число байтов, если не указана также буква K (для килобайтов), M (для мегабайтов) или G (для гигабайтов). (Если указано значение *PrefetchSize*, оно преобразуется в значение *PrefetchPages* с использованием предоставленного значения *PageSize*). *BufferPoolName* указывает имя буферного пула, который должен использоваться создаваемым табличным пространством. (Размер страницы указанного буферного пула должен совпадать с размером страницы создаваемого табличного пространства, иначе оператор CREATE TABLESPACE завершится неудачей).



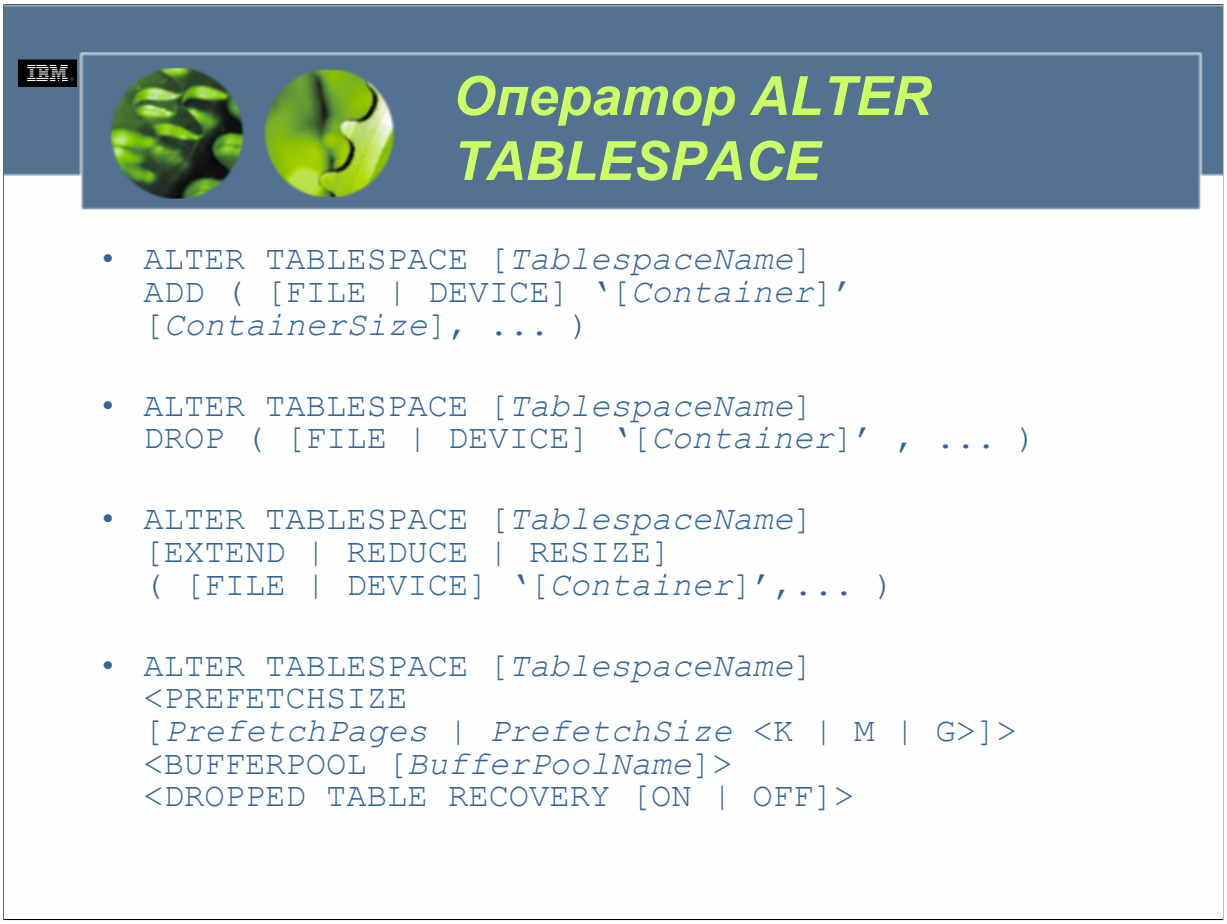
WCF03.0

Табличные пространства можно создавать также с использованием мастера Создания табличного пространства (Create Table Space Wizard), который можно вызвать, выбрав соответствующее действие из меню *Табличные пространства (Table Spaces)* центра управления.

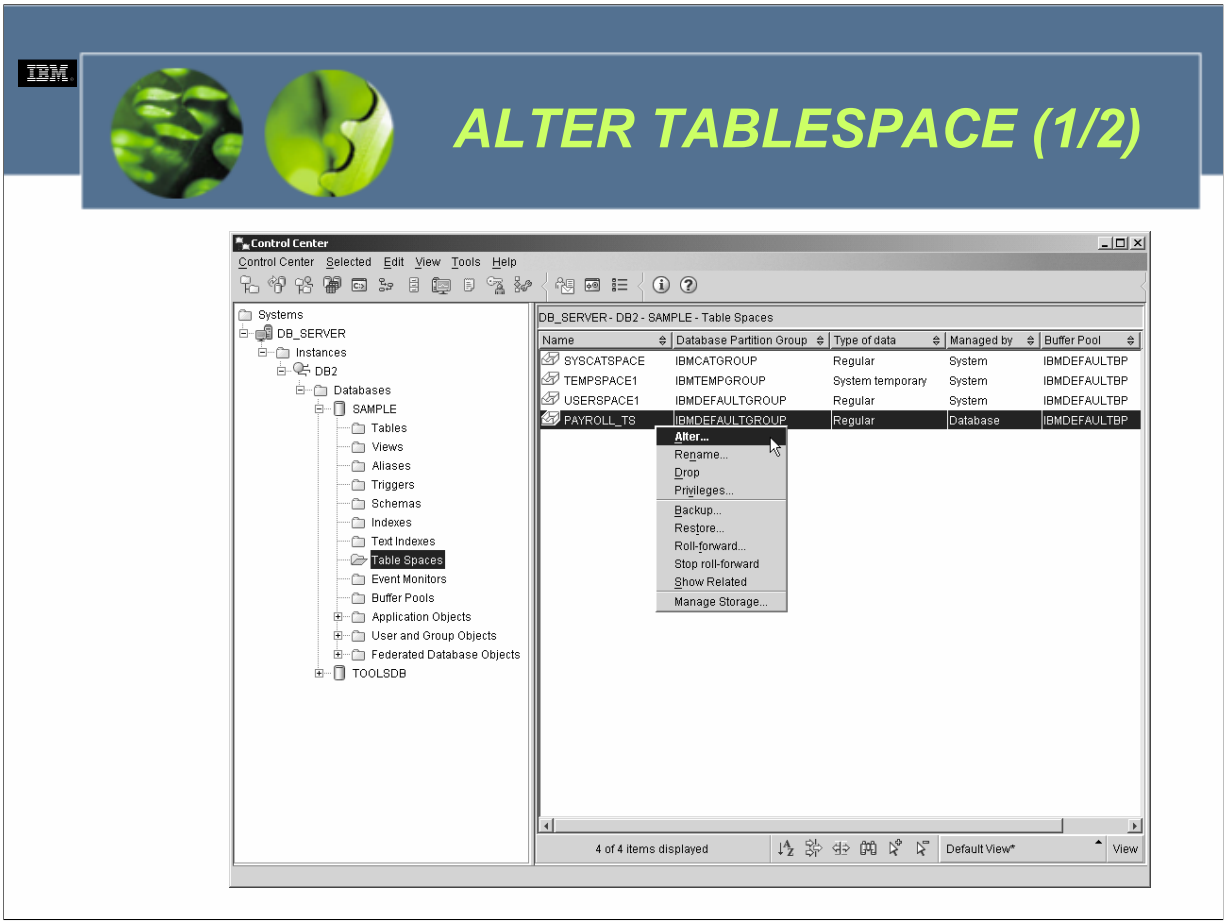


WCF03.0

Табличные пространства используются для контроля того, где данные хранятся физически, и для предоставления промежуточного уровня между объектами базы данных и каталогами, файлами или непосредственными устройствами (которые называются контейнерами), в которых данные находятся физически. В зависимости от того, как оно было определено, табличное пространство может быть управляемым системой (SMS) или управляемым базой данных (DMS). Для табличных пространств SMS каждый используемый контейнер должен быть каталогом, который находится в файловом пространстве операционной системы, а за управление хранением данных отвечает файловый менеджер операционной системы. Для табличных пространств DMS каждый используемый контейнер должен быть либо предварительно выделенным файлом фиксированного размера, либо непосредственным устройством, а за управление хранением данных отвечает менеджер баз данных DB2. По умолчанию в ходе процесса создания базы данных для базы данных создаются три табличных пространства (с именами SYSCATSPACE, USERSPACE1 и TEMPSPACE1).

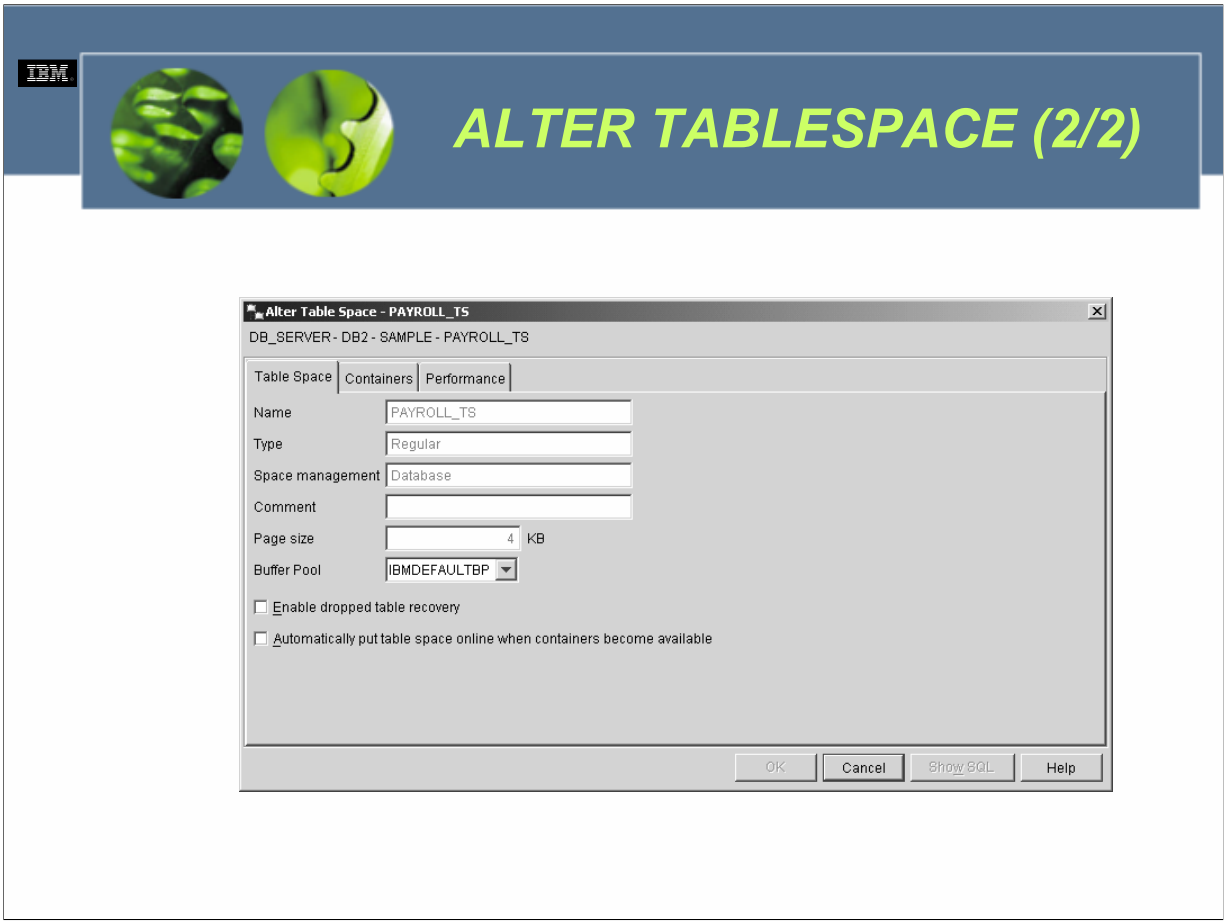


TablespaceName указывает имя, связанное с изменяемым табличным пространством.*Container*Определяет один или несколько контейнеров, которые должны использоваться для хранения данных, связанных в изменяемым табличным пространством. *ContainerSize* определяет число страниц, которые должны храниться в указанном контейнере табличного пространства. *PrefetchPages* определяет число страниц данных, которые должны прочитываться из табличного пространства, когда осуществляется предварительная выборка. *PrefetchSize* определяет объем данных, который должен быть прочитан из табличного пространства, когда осуществляется предварительная выборка. Указанное для этого параметра значение рассматривается, как общее число в байтах, если не указана также буква K (для килобайтов), M (для мегабайтов) или G (для гигабайтов). (Если указано значение *PrefetchSize*, оно преобразуется в значение *PrefetchPages* с использованием размера страницы изменяемого табличного пространства). *BufferPoolName* указывает имя буферного пула, который должен использоваться изменяемым табличным пространством. (Размер страницы указанного буферного пула должен совпадать с размером страницы, используемым изменяемым табличным пространством).



WCF03.0

Табличные пространства можно также изменить, используя диалог Изменение табличного пространства (Alter Table Space), который можно вызвать, выбрав соответствующее действие из меню *Табличные пространства (Table Spaces)* центра управления.



WCF03.0

Поскольку табличные пространства SMS полагаются на управление памятью физического пространства хранения операционной системой, их редко бывает нужно изменять после того, как они были успешно созданы. Табличные пространства DMS, с другой стороны, нужно внимательно проверять, чтобы убедиться, что предварительно выделенный(-е) файл(-ы) фиксированного размера или непосредственные физические устройства, которые они используют для пространства хранения , имеют достаточно свободной памяти для удовлетворения потребностей базы данных. Когда объем свободной памяти, доступной табличному пространству DMS, становится опасно маленьким (обычно менее 10 процентов), дополнительную свободную память можно добавить либо увеличив размер одного или нескольких его контейнеров, либо сделав доступным ему один или более дополнительных контейнеров. Размеры существующих контейнеров табличного пространства могут быть изменены, новые контейнеры могут быть сделаны доступными существующему табличному пространству и свойства существующих табличных пространств могут быть изменены посредством выполнения оператора SQL ALTER TABLESPACE.

WCF03.0

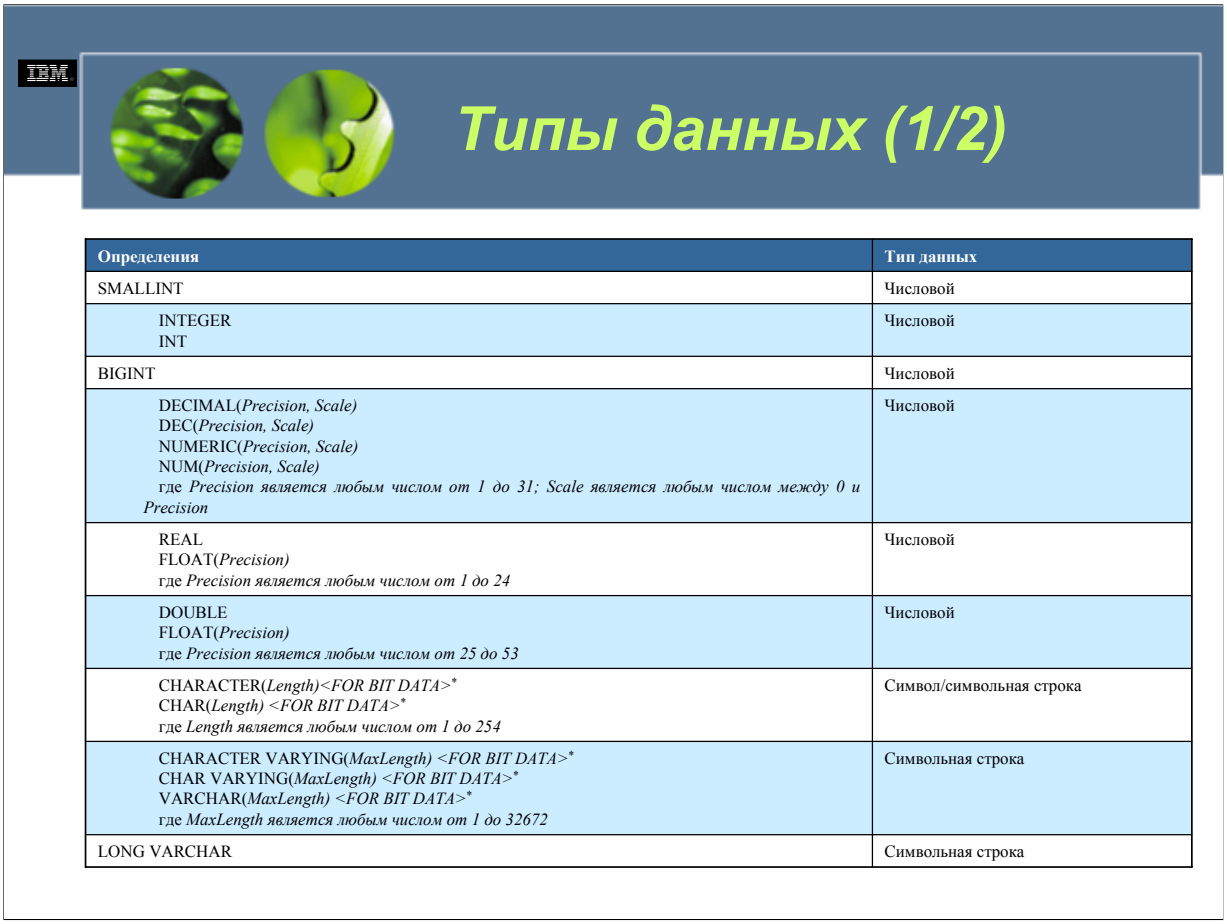
Таблица является логической структурой, используемой для представления данных как коллекции неупорядоченных строк с фиксированным числом столбцов. Каждый столбец содержит набор значений одного типа данных, а каждая строка содержит фактические данные таблицы. Поскольку таблицы являются базовыми объектами данных, использующимися для хранения информации, для одной базы данных их создается множество. Таблицы создаются путем выполнения оператора SQL CREATE TABLE. В своей простейшей форме синтаксис для данного оператора следующий:

CREATE TABLE [*TableName*]

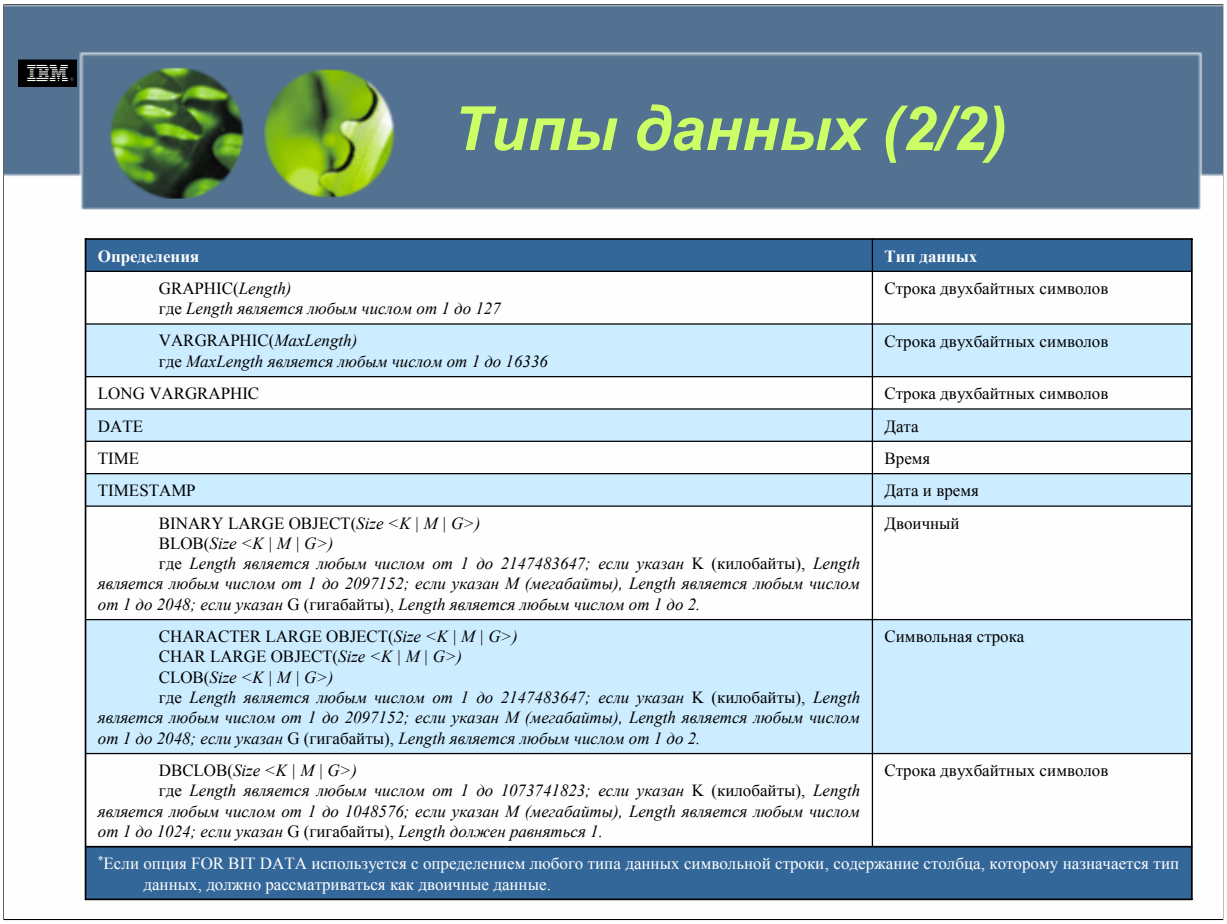
([ColumnName] [DataType] ,...)

где:

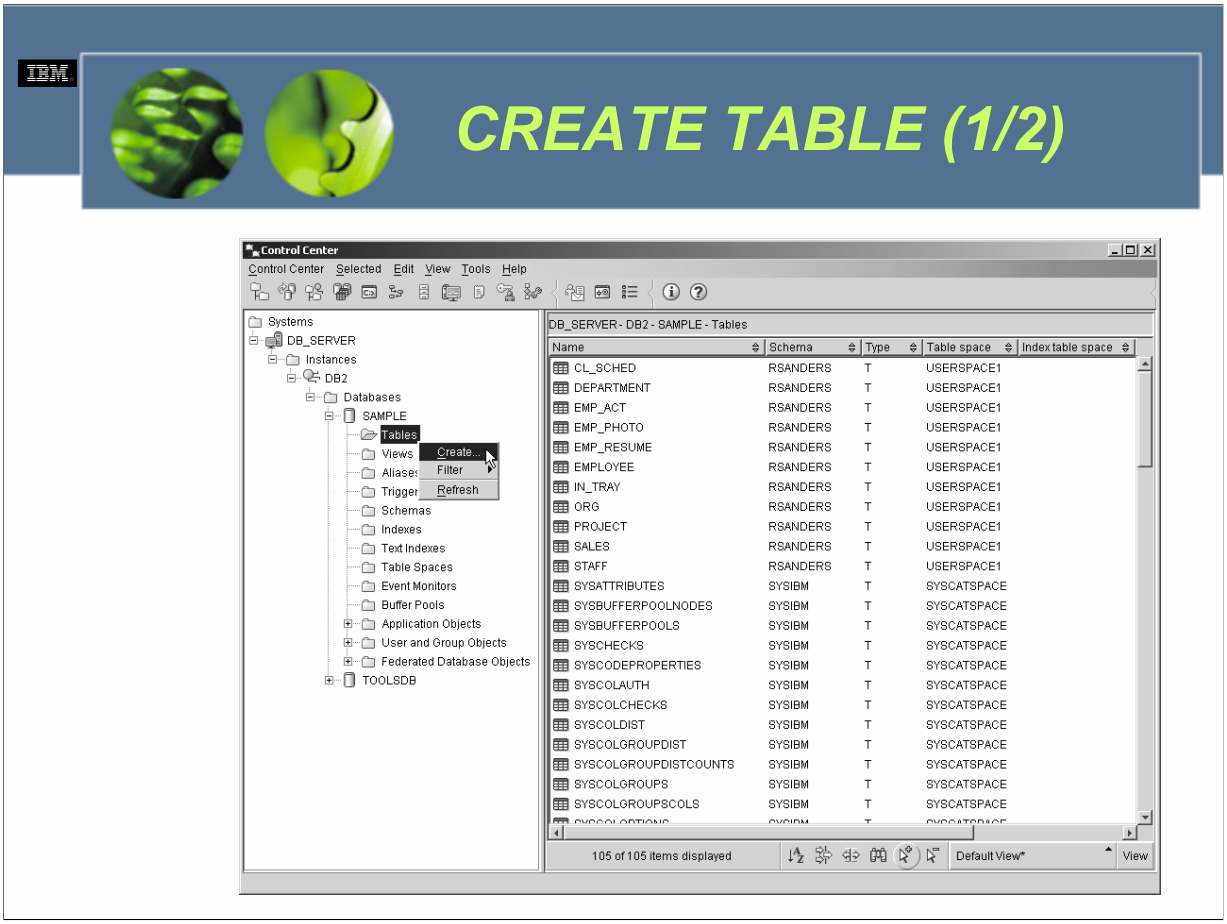
*TableName*Определяет имя, которое должно быть назначено создаваемой таблице. (Имя таблицы должно быть уникальным в пределах схемы, в которой должна быть определена таблица). *ColumnName*Определяет уникальное имя (в пределах определения таблицы) для назначения столбцу, который должен быть создан. *DataType*Указывает тип данных (встроенный или пользовательский), который должен быть назначен создаваемому столбцу; указанный тип данных определяет вид значений данных, которые могут храниться в столбце. (В таблице 5-1 содержится список действительных определений типов данных).



Определения	Тип данных
SMALLINT	Числовой
INTEGER INT	Числовой
BIGINT	Числовой
DECIMAL(*Precision, Scale)* DEC(*Precision, Scale)* NUMERIC(*Precision, Scale)* NUM(*Precision, Scale)* где *Precision является любым числом от 1 до 31; Scale является любым числом между 0 и Precision*	Числовой
REAL FLOAT(*Precision)* где *Precision является любым числом от 1 до 24*	Числовой
DOUBLE FLOAT(*Precision)* где *Precision является любым числом от 25 до 53*	Числовой
CHARACTER(*Length)<FOR BIT DATA>** CHAR(*Length) <FOR BIT DATA>** где *Length является любым числом от 1 до 254*	Символ/символьная строка
CHARACTER VARYING(*MaxLength) <FOR BIT DATA>** CHAR VARYING(*MaxLength) <FOR BIT DATA>** VARCHAR(*MaxLength) <FOR BIT DATA>** где *MaxLength является любым числом от 1 до 32672*	Символьная строка
LONG VARCHAR	Символьная строка



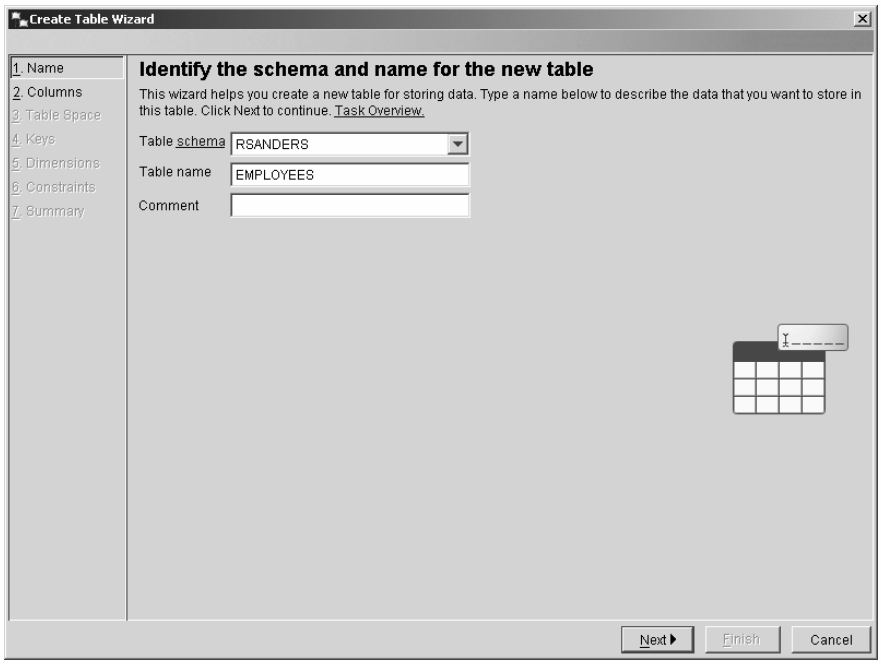
Определения	Тип данных
GRAPHIC(*Length*) где *Length является любым числом от 1 до 127*	Строка двухбайтных символов
VARGRAPHIC(*MaxLength*) где *MaxLength является любым числом от 1 до 16336*	Строка двухбайтных символов
LONG VARGRAPHIC	Строка двухбайтных символов
DATE	Дата
TIME	Время
TIMESTAMP	Дата и время
BINARY LARGE OBJECT(*Size <K | M | G>*) BLOB(*Size <K | M | G>*) где *Length является любым числом от 1 до 2147483647; если указан* K (килобайты), *Length является любым числом от 1 до 2097152; если указан M (мегабайты), Length является любым числом от 1 до 2048; если указан* G (гигабайты), *Length является любым числом от 1 до 2.*	Двоичный
CHARACTER LARGE OBJECT(*Size <K | M | G>*) CHAR LARGE OBJECT(*Size <K | M | G>*) CLOB(*Size <K | M | G>*) где *Length является любым числом от 1 до 2147483647; если указан* K (килобайты), *Length является любым числом от 1 до 2097152; если указан M (мегабайты), Length является любым числом от 1 до 2048; если указан* G (гигабайты), *Length является любым числом от 1 до 2.*	Символьная строка
DBCLOB(*Size <K | M | G>*) где *Length является любым числом от 1 до 1073741823; если указан* K (килобайты), *Length является любым числом от 1 до 1048576; если указан M (мегабайты), Length является любым числом от 1 до 1024; если указан* G (гигабайты), *Length должен равняться 1.*	Строка двухбайтных символов
*Если опция FOR BIT DATA используется с определением любого типа данных символьной строки, содержание столбца, которому назначается тип данных, должно рассматриваться как двоичные данные.	

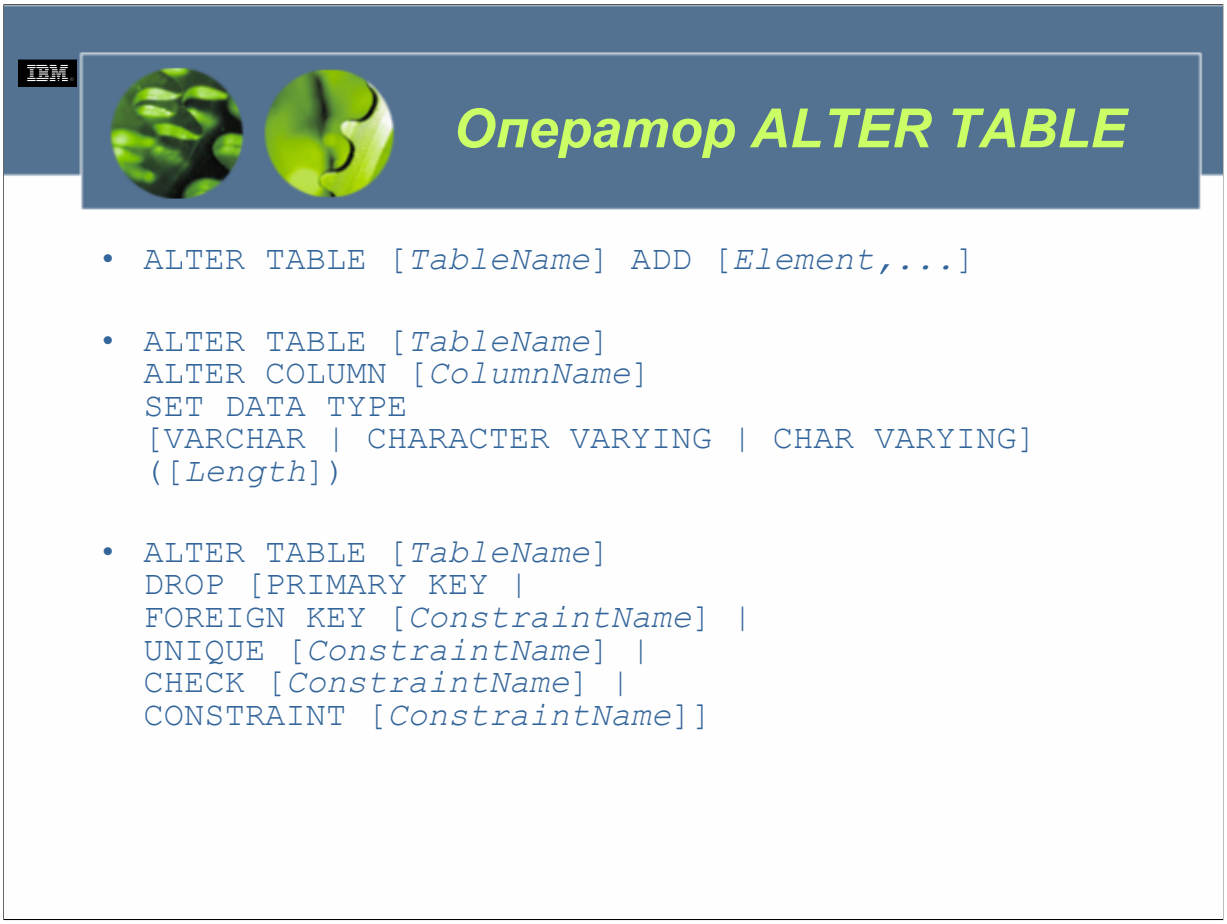


WCF03.0

Подобно многим другим доступным объектам базы данных, таблицы можно создавать с использованием графической утилиты, доступ к которой можно получить из центра управления. В данном случае это утилита мастера Создания таблицы (Create Table Wizard), ее можно запустить, выбрав соответствующее действие из меню *Таблицы (Tables)* центра управления.

CREATE TABLE (1/2)





WCF03.0

Можно изменить определенные свойства существующей таблицы, добавить дополнительные столбцы и ограничения (ограничения подробно освещаются в главе 6 «Работа с объектами DB2 UDB»), удалить существующие ограничения и увеличить длину значений типов данных с переменным числом символов, допустимую для определенного столбца, выполнив оператор SQL ALTER TABLE. Аналогично оператору CREATE TABLE, оператор ALTER TABLE может быть довольно сложным. Однако в своей простейшей форме синтаксис для оператора ALTER TABLE следующий:

ALTER TABLE [*TableName*] ADD [*Element,...*]

или

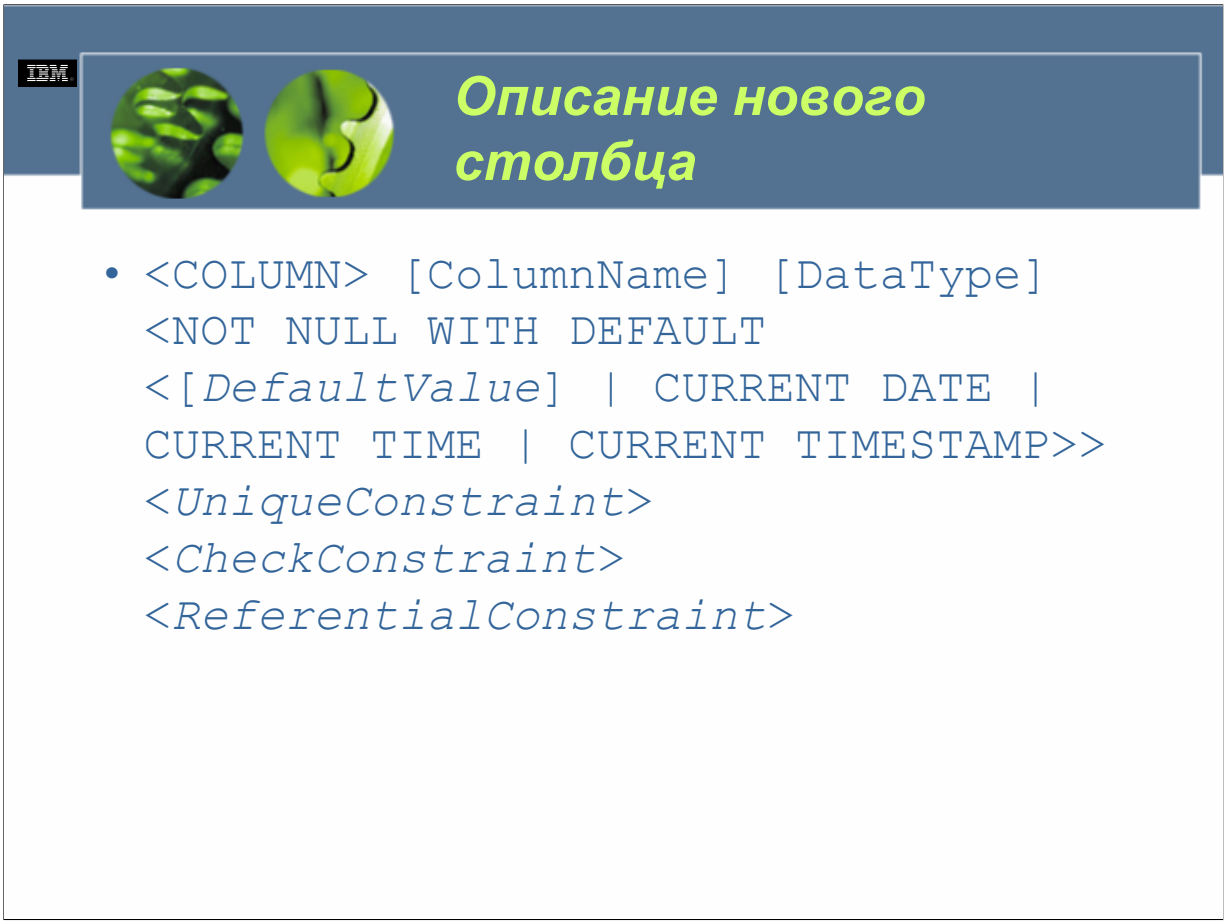
ALTER TABLE [*TableName*]
ALTER COLUMN [*ColumnName*]
SET DATA TYPE
[VARCHAR | CHARACTER VARYING | CHAR VARYING] ([*Length*])

или

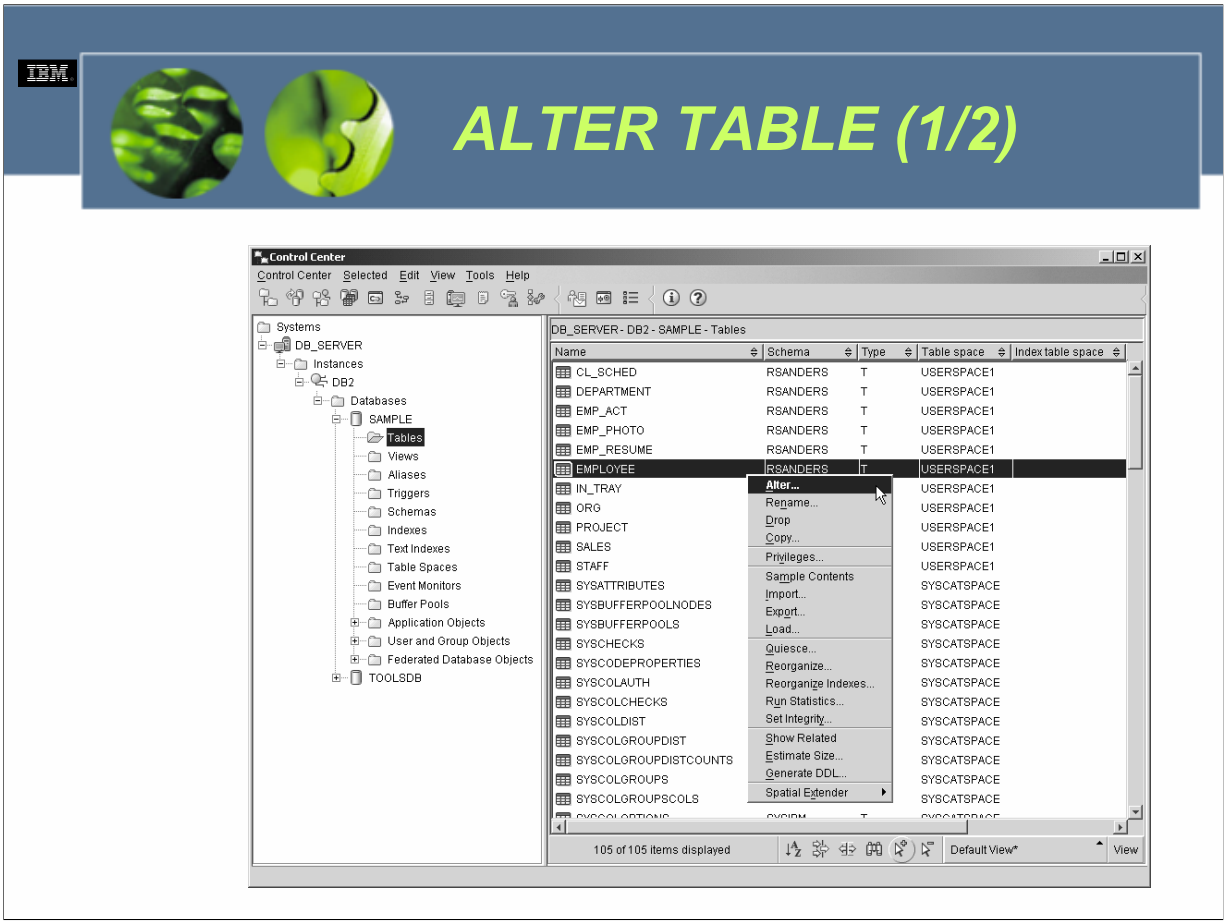
ALTER TABLE [*TableName*]
DROP [PRIMARY KEY |
FOREIGN KEY [*ConstraintName*] |
UNIQUE [*ConstraintName*] |
CHECK [*ConstraintName*] |
CONSTRAINT [*ConstraintName*]]

где:

TableName указывает имя, назначенное таблице, определение которой изменяется. *Element* указывает один или несколько столбцов, ограничения уникального/первичного ключа, реляционные ограничения и/или проверочные ограничения, которые должны быть добавлены к определению существующей таблицы. Синтаксис, используемый для определения каждого из этих элементов, меняется в зависимости от определяемого элемента. *ColumName* указывает в таблице существующий столбец с символами переменной длины, размер которого должен быть изменен. *Length* определяет новую максимальную длину, которую могут иметь значения данных для столбца с переменным числом символов. (Новое представленное значение должно быть больше, чем текущая длина столбца). *ConstraintName* определяет существующее уникальное ограничение, ограничение внешнего ключа или проверочное ограничение, которое должно быть удалено из определения таблицы.



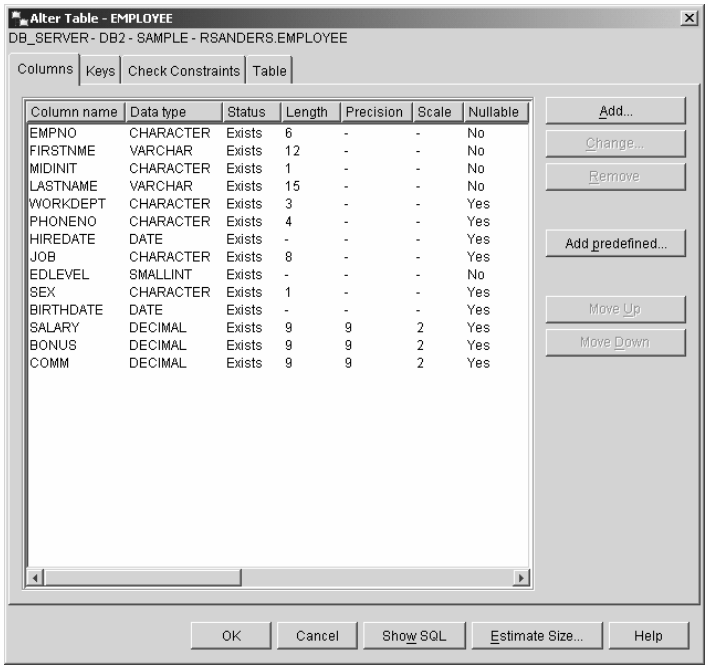
ColumnName определяет уникальное имя, которое должно быть назначено создаваемому столбцу.*DataType* обозначает тип данных (встроенный или пользовательский), который должен быть назначен создаваемому столбцу; указанный тип данных определяет вид значений данных, которые могут храниться в столбце. (В таблице 5-1 содержится список действительных определений типов данных). *DefaultValue* определяет значение, которое должно быть присвоено для столбца в случае, если для него не будет указано значение в ходе операции вставки или изменения значения столбца для таблицы. *UniqueConstraint* определяет ограничение уникального или первичного ключа, которое должно быть связано со столбцом. *CheckConstraint* определяет проверочное ограничение, которое должно быть связано со столбцом. *ReferentialConstraint* определяет реляционное ограничение, которое должно быть связано со столбцом. Синтаксис оператора ALTER TABLE, используемый для добавления ограничения уникального или первичного ключа, проверочного ограничения и/или реляционного ограничения в качестве части определения столбца тот же самый, что и синтаксис, использованный с более сложной формой оператора CREATE TABLE.

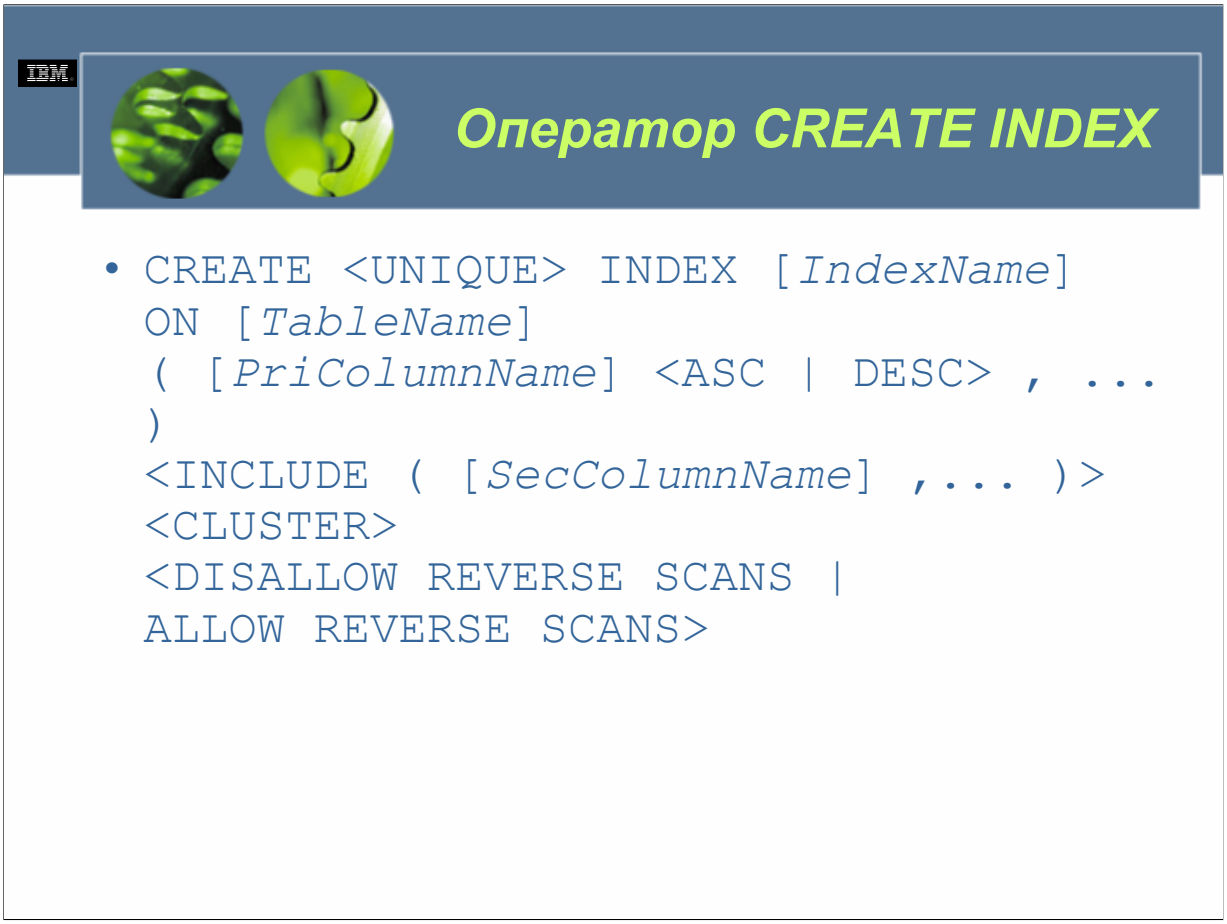


WCF03.0

Существующие таблицы можно также изменять, используя диалог Изменение таблицы (Alter Table), который можно вызвать, выбрав соответствующее действие из меню *Таблицы (Tables)* центра управления.

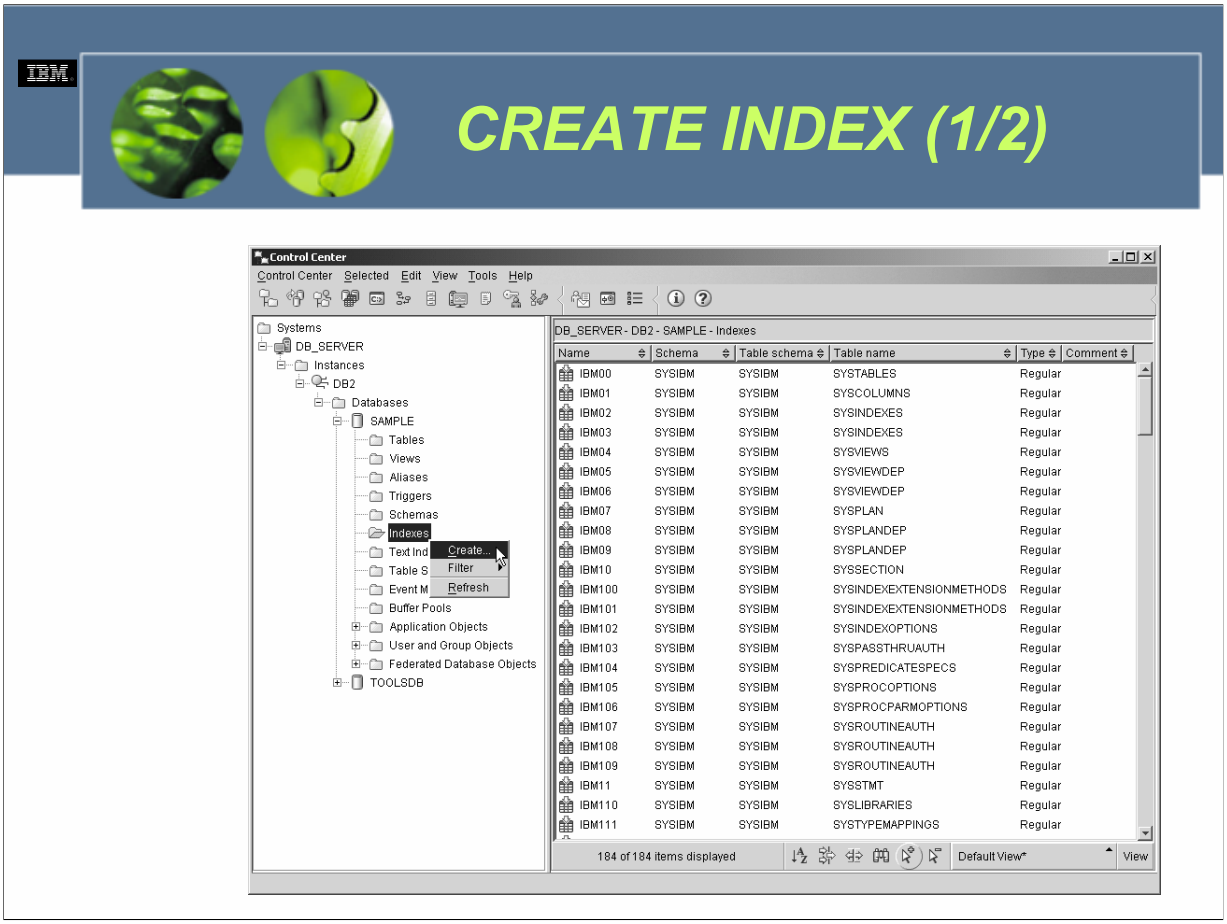
ALTER TABLE (2/2)





WCF03.0

IndexName определяет имя, которое должно быть назначено создаваемому индексу. *TableName* указывает имя, назначенное базовой таблице, с которой должен быть связан создаваемый индекс. *PriColumnName* указывает один или более первичных столбцов, которые должны быть частью ключа индекса. (Для обеспечения уникальности данных в связанной базовой таблице будут использоваться объединенные значения каждого указанного первичного столбца). *SecColumnName* указывает один или более вторичных столбцов, значения которых должны храниться со значениями указанных первичных столбцов, но которые не используются для обеспечения уникальности данных. Если при выполнении оператора CREATE INDEX указано условие UNIQUE, строки в таблице, связанные с создаваемым индексом, не должны содержать два или более одинаковых значения в наборах столбцов, составляющих ключ индекса. Если базовая таблица, для которой должен быть создан индекс, содержит данные, подобная уникальность проверяется, когда менеджер баз данных DB2 пытается создать указанный индекс; после создания индекса эта уникальность обеспечивается каждый раз при осуществлении операции вставки или изменения для таблицы. В обоих случаях, если нарушена уникальность ключа индекса, операция создания индекса, вставки или изменения потерпит неудачу, и будет сгенерирована ошибка. Важно иметь в виду, что когда используется условие UNIQUE, может быть ключ индекса, который содержит одно (и только одно) значение NULL.



WCF03.0

Индексы можно также создавать, используя диалог Создание индекса (Create Index), который можно вызвать, выбрав соответствующее действие из меню *Индексы (Indexes)* центра управления.

Если индекс создается для пустой таблицы, в этом индексе не будет сохраненных данных до тех пор, пока таблица, с которой связан индекс, не будет заполнена. С другой стороны, если индекс создается для таблицы, уже содержащей данные, для существующих данных будут сгенерированы элементы индекса и добавлены к индексу, как только он будет создан. Для таблицы может быть создано любое число индексов с использованием широкого разнообразия сочетаний столбцов. Однако за каждый индекс приходится платить как дополнительным расходом пространства, так и производительностью: поскольку каждый индекс дублирует свои ключевые данные, а для этого дублирования требуется дополнительное пространство памяти, а каждое изменение таблицы приводит к аналогичным изменениям всех индексов, определенных для таблицы, производительность при осуществлении операций вставки, изменения и удаления может снизиться. На самом деле, если создается большое число индексов для таблицы, которая часто изменяется, общая производительность снизится, а не возрастет, для всех операций, *за исключением* получения данных. Таблицы, которые используют для добывания данных (data mining), интеллектуальных ресурсов предприятия (business intelligence), долговременного хранения данных предприятия и других приложений, которые выполняют множество (часто очень сложных) запросов, в то же время редко изменяя данные, являются главными целями для множества индексов. С другой стороны, таблицы, которые используются в средах оперативной обработки транзакций (OLTP – On-Line Transactional Processing) или других средах, в которых требуется высокая пропускная способность данных, должны использовать индексы скупо.

WCF03.0

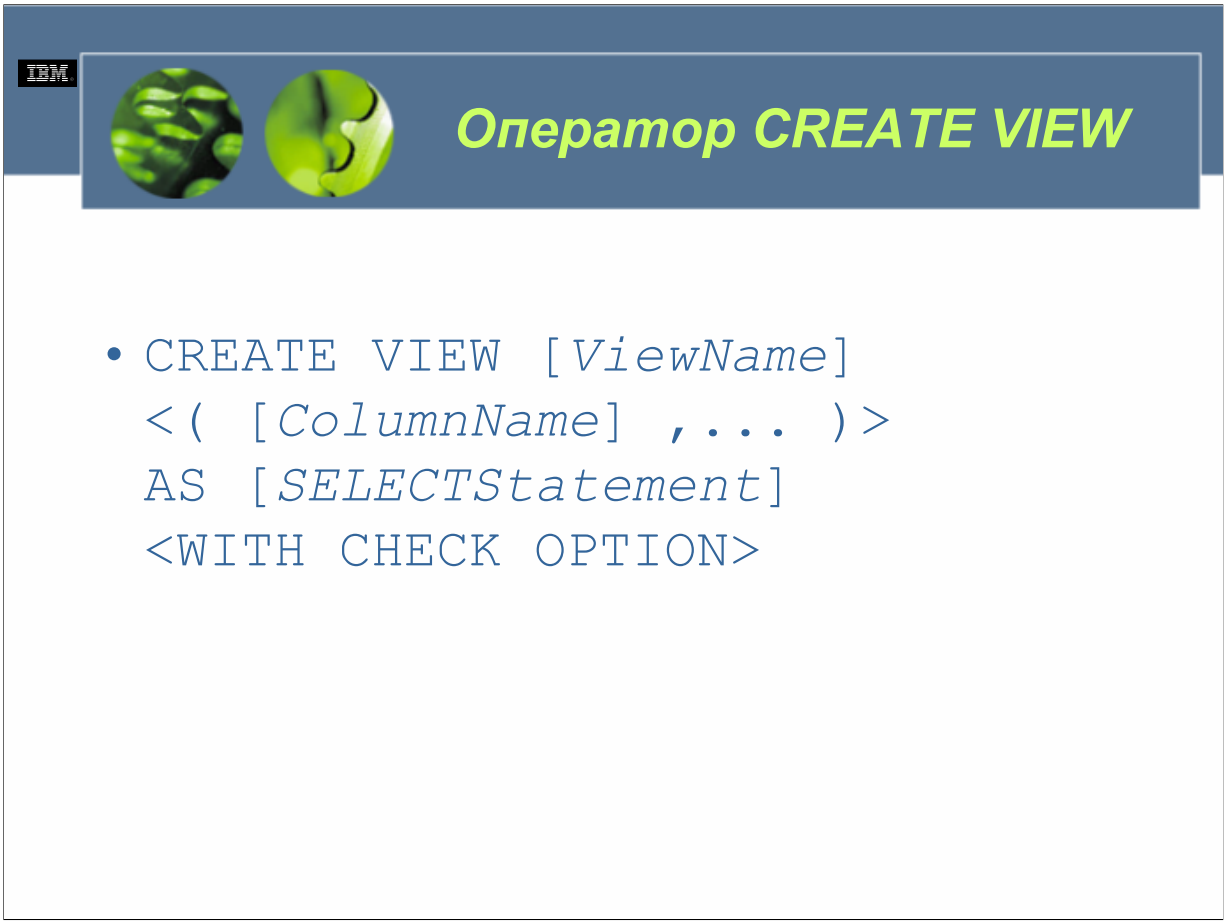
Индекс является объектом, содержащим упорядоченный набор указателей, которые ссылаются на записи, хранящиеся в базовой таблице. Индексы важны, потому что они:

Предоставляют быстрый, эффективный способ обнаружения определенных строк данных в очень больших таблицах. (В некоторых случаях всю информацию, необходимую для выполнения запроса, можно найти в самом индексе, в этом случае доступ к таблице с фактическими данными не требуется).

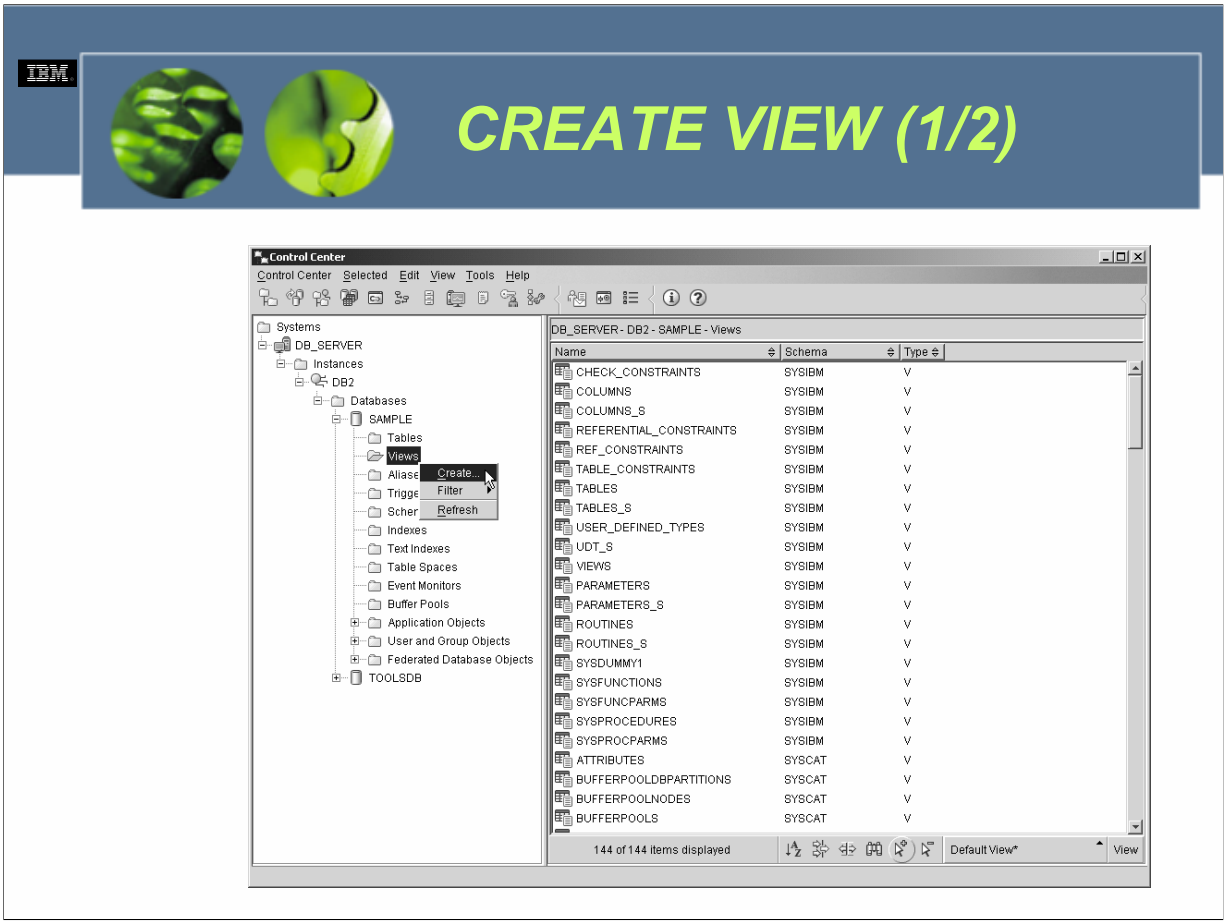
Обеспечивают логическое упорядочивание строк таблицы. (Когда используются индексы, значения одного или нескольких столбцов можно отсортировать в восходящем или нисходящем порядке; это свойство очень полезно при обработке запросов, содержащих условия ORDER BY или GROUP BY).

Могут обеспечить уникальность записей, хранящихся в таблице.

Могут потребовать от таблицы использование *кластеризованного (clustered)* хранения, которое вызывает физическое размещение строк таблицы в соответствии с упорядочиванием значений столбца их индекса. (Хотя все индексы обеспечивают логическое упорядочивание данных, лишь кластеризованный индекс обеспечивает физическую сортировку данных).



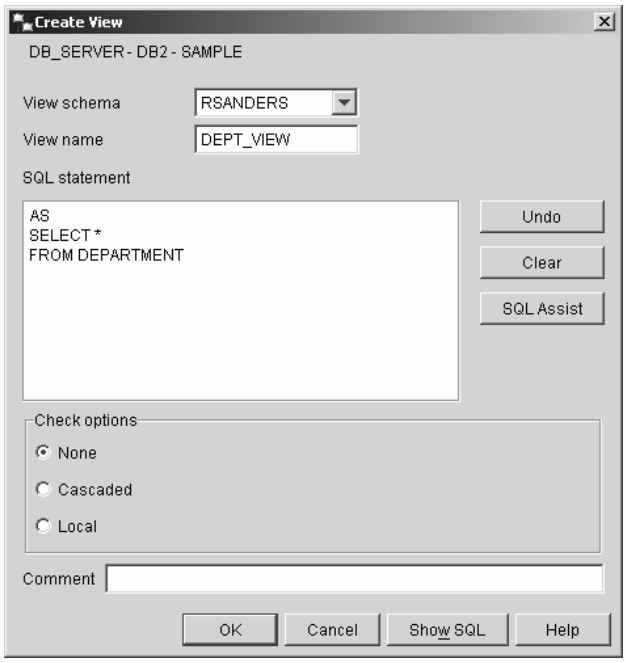
ViewName определяет имя, которое должно быть назначено создаваемой производной таблице.*ColumnName* обозначает имена одного или нескольких столбцов, которые должны быть включены в создаваемую производную таблицу. Если указан список имен столбцов, число предоставленных имен столбцов должно соответствовать числу столбцов, который будет возвращен оператором SELECT, использованным для создания производной таблицы. (Если список имен столбцов не предоставлен, столбцы производной таблицы унаследуют имена, которые назначены столбцам, возвращенным оператором SELECT, использованным для создания производной таблицы). *SELECTStatement* определяет оператор SQL SELECT, который при выполнении сгенерирует данные, которые можно увидеть с использованием производной таблицы, которая должна быть создана.

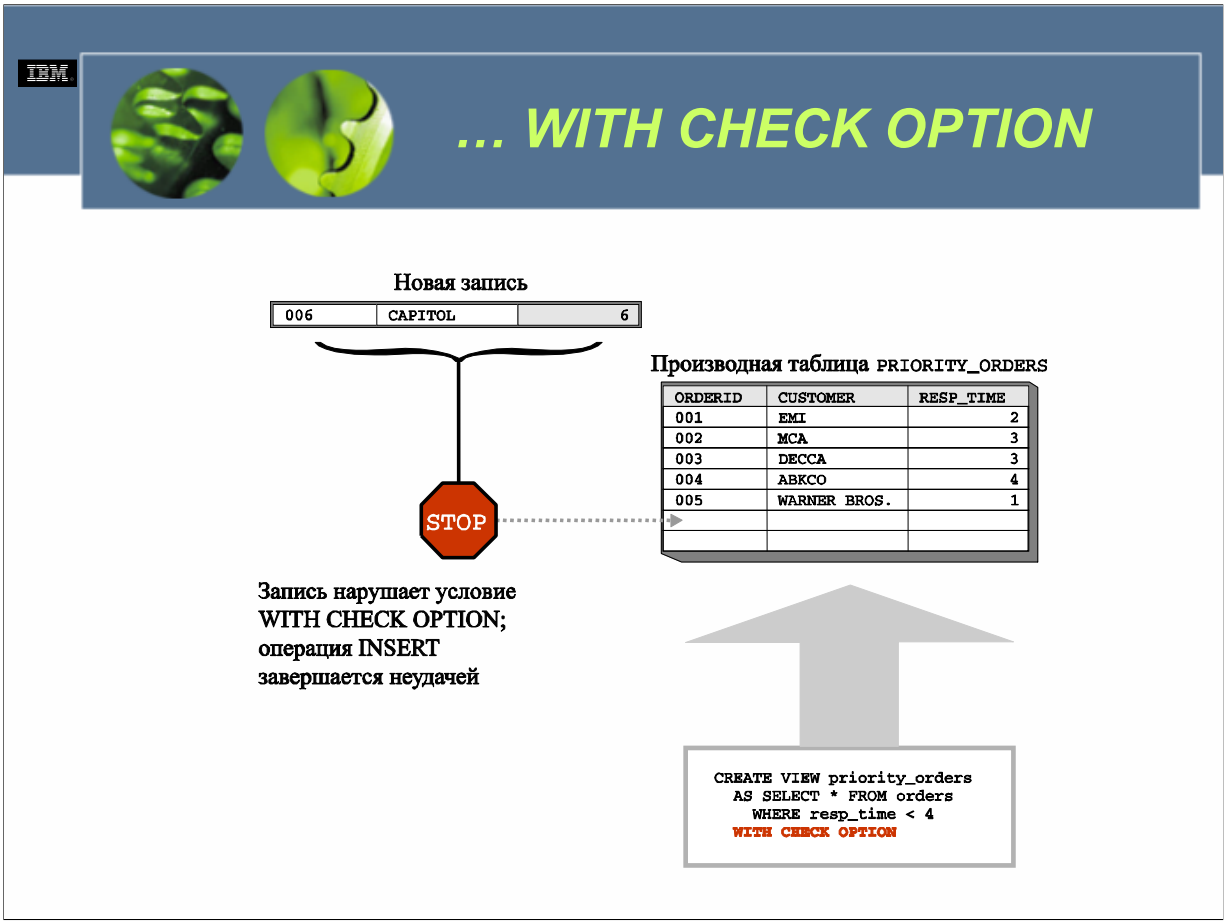


WCF03.0

Производные таблицы можно также создавать, используя диалог Создание производной таблицы (Create View), который можно вызвать, выбрав соответствующее действие из меню *Производные таблицы (Views)* центра управления.

CREATE VIEW (2/2)





WCF03.0

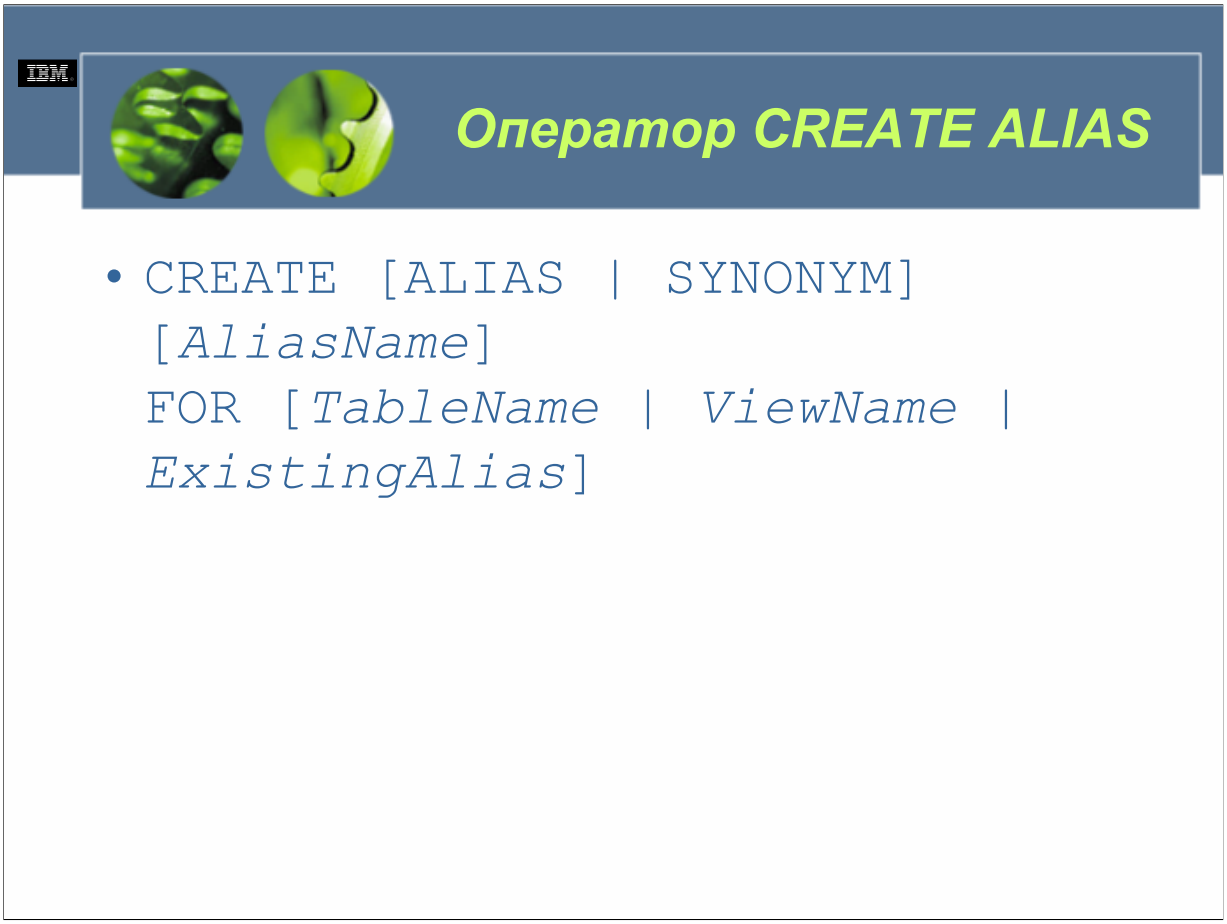
Если указано условие WITH CHECK OPTION оператора SQL CREATE VIEW, операции вставки и изменения , осуществляемые для созданной производной таблицы, проверяются на предмет того, что все строки, вставляемые или изменяемые в базовой таблице, на которую ссылается производная таблица, удовлетворяют определению производной таблицы (в противном случае операция вставки/изменения потерпит неудачу). Так что же это в точности означает? Предположим, была создана производная таблица с использованием следующего оператора CREATE VIEW:

CREATE VIEW PRIORITY_ORDERS

AS SELECT * FROM ORDERS WHERE RESPONSE_TIME < 4

WITH CHECK OPTION

Теперь представьте, что пользователь пытается ввести в эту производную таблицу запись, значение RESPONSE_TIME которой 6. Операция вставки потерпит неудачу, поскольку запись нарушает определение производной таблицы. Если бы производная таблица не была создана с условием WITH CHECK OPTION, операция вставки была бы успешной, даже если новая запись не была бы видна в производной таблице, которая была использована для ее добавления.



WCF03.0

Алиас является просто альтернативным именем для таблицы или производной таблицы, и что после создания на алиасы можно ссылаться тем же способом, как на таблицы или производные таблицы. Используя алиасы, можно конструировать операторы SQL таким образом, чтобы они были независимы от составных имен, определяющих базовые или производные таблицы, на которые они ссылаются. Каждый раз, когда в операторе SQL используется алиас, поведение оператора то же самое, как при использовании вместо алиаса целевого объекта (исходной таблицы или производной таблицы). Поэтому любое приложение, которое использует для доступа к данным алиас, легко можно заставить работать с множеством различных целевых объектов. Это происходит потому, что когда изменяется целевой объект алиаса, не нужно делать никаких изменений в приложении, использующем этот алиас.

Алиасы можно создавать, выполняя оператор SQL CREATE ALIAS. Базовый синтаксис для этого оператора следующий:

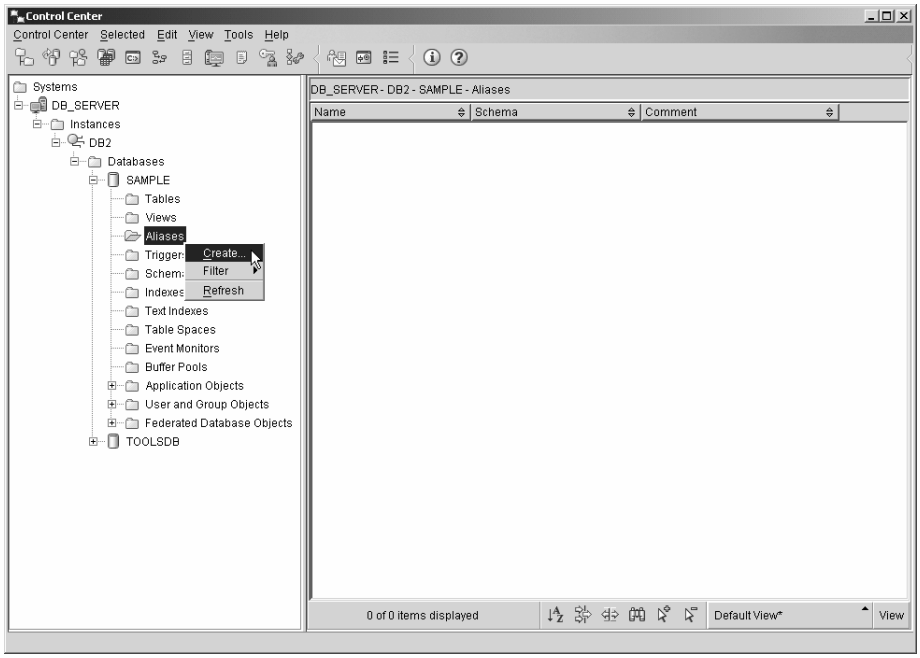
CREATE [ALIAS | SYNONYM] [*AliasName*]

FOR [*TableName* | *ViewName* | *ExistingAlias*]

где:

AliasName определяет имя, которое должно быть назначено создаваемому алиасу.*TableName* обозначает имя, назначенное таблице, на которую должен ссылаться создаваемый алиас.*ViewName* обозначает имя, назначенное производной таблице, на которую должен ссылаться создаваемый алиас. *ExistingAlias* обозначает имя, назначенное алиасу, на который должен ссылаться создаваемый алиас.

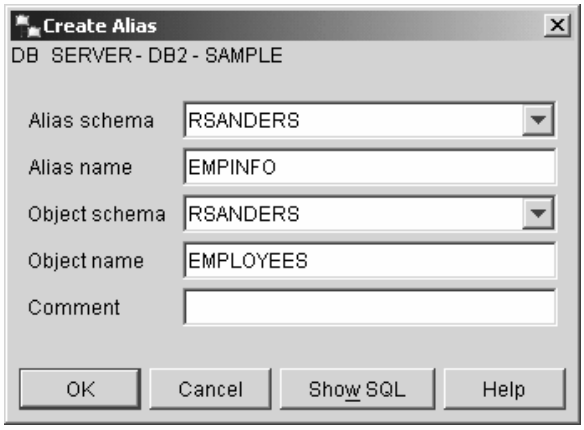
CREATE ALIAS (1/2)

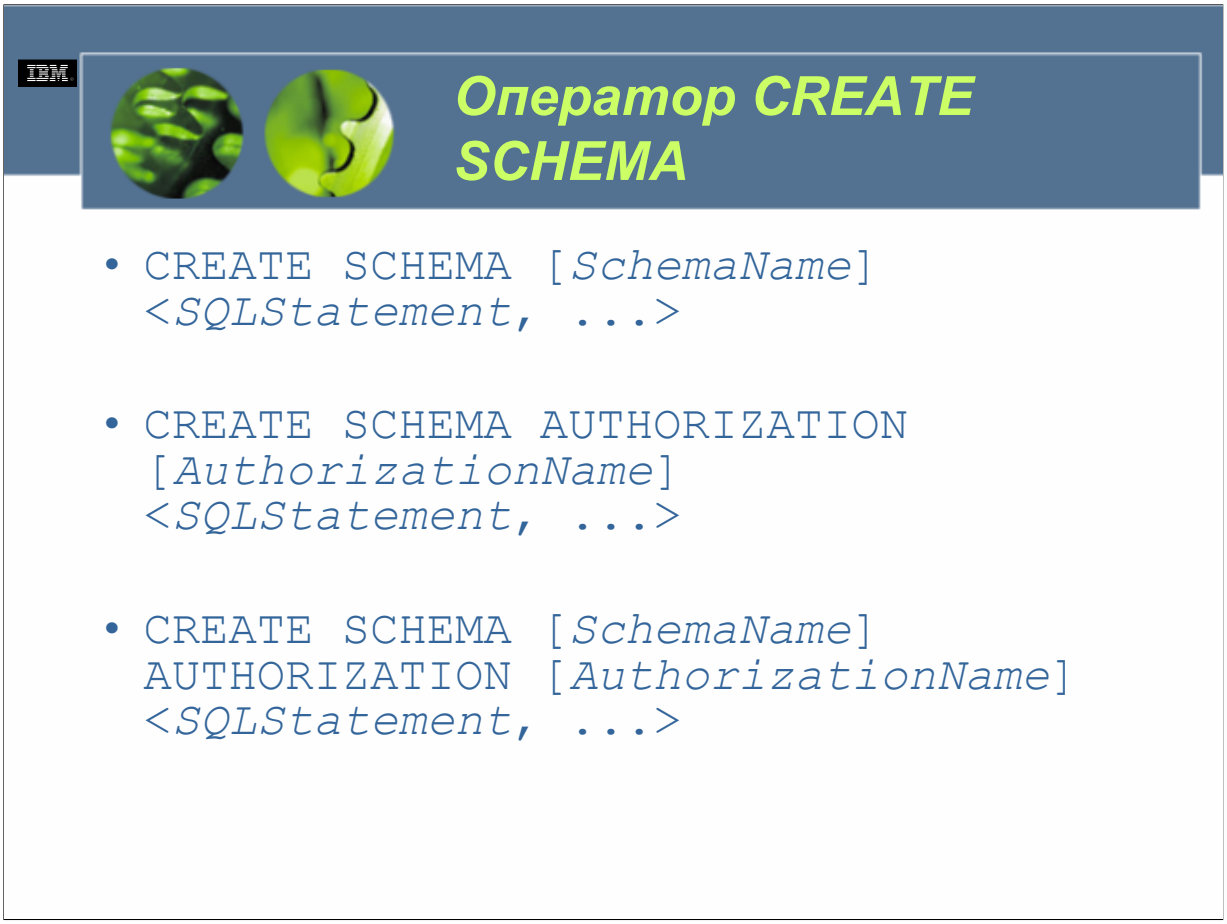


WCF03.0

Алиасы могут быть созданы также с использованием диалога Создание алиаса (Create Alias), который можно вызвать, выбрав соответствующее действие из меню *Алиасы (Aliases)* центра управления.

CREATE ALIAS (2/2)





WCF03.0

Схемы неявно создаются каждый раз, когда создается объект данных, которому было назначено составное имя, отличающееся от имен существующих схем, которые можно найти в базе данных – при условии, что пользователь, создающий объект, обладает полномочием IMPLICIT_SCHEMA. (Если в ходе процесса создания спецификатор не включен в качестве части имени, присваиваемого объекту, по умолчанию в качестве схемы используется ID авторизации пользователя, создающего объект). Схемы могут быть созданы явным образом путем выполнения оператора SQL CREATE SCHEMA. Базовый синтаксис для этого оператора следующий:

CREATE SCHEMA [*SchemaName*]

<*SQLStatement*, ...>

или

CREATE SCHEMA AUTHORIZATION [*AuthorizationName*]

<*SQLStatement*, ...>

или

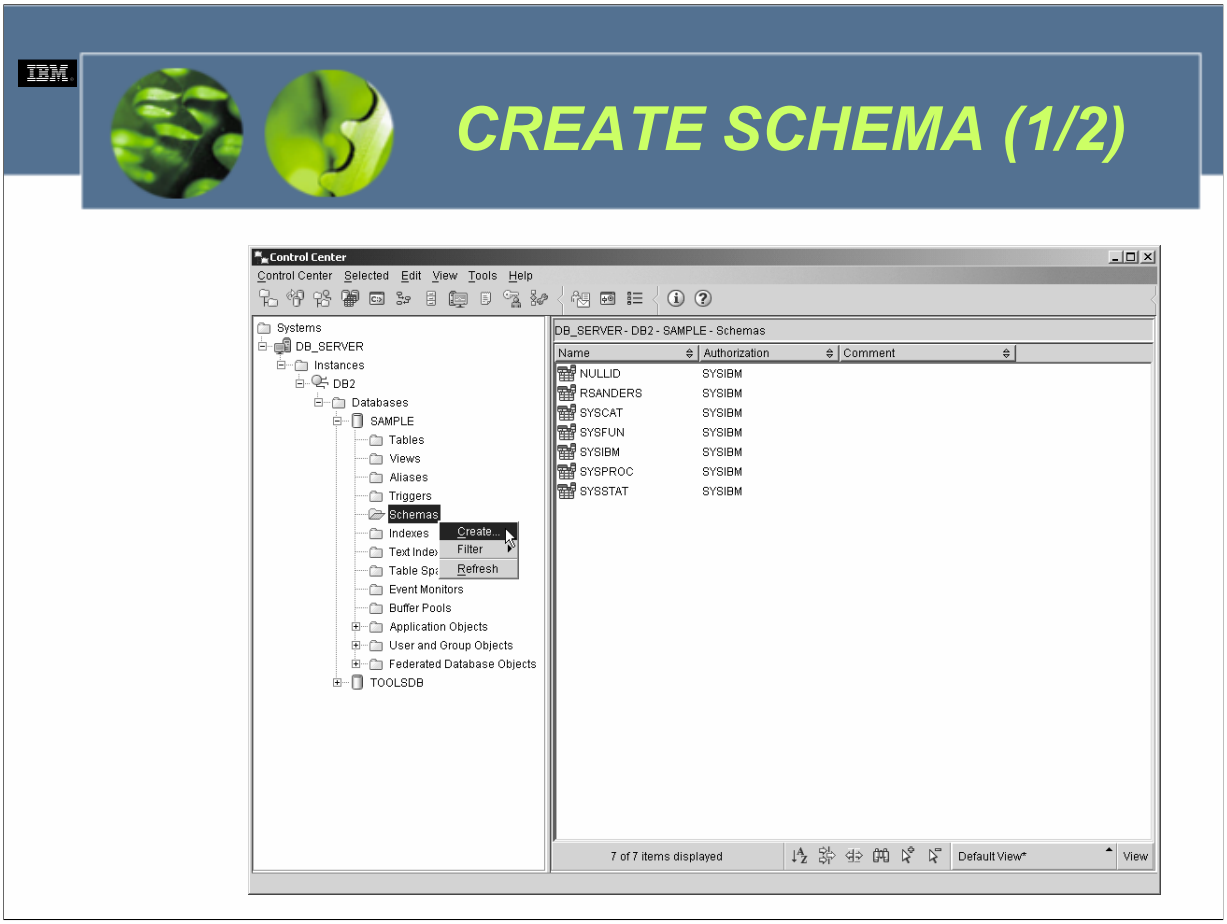
CREATE SCHEMA [*SchemaName*]

AUTHORIZATION [*AuthorizationName*]

<*SQLStatement*, ...>

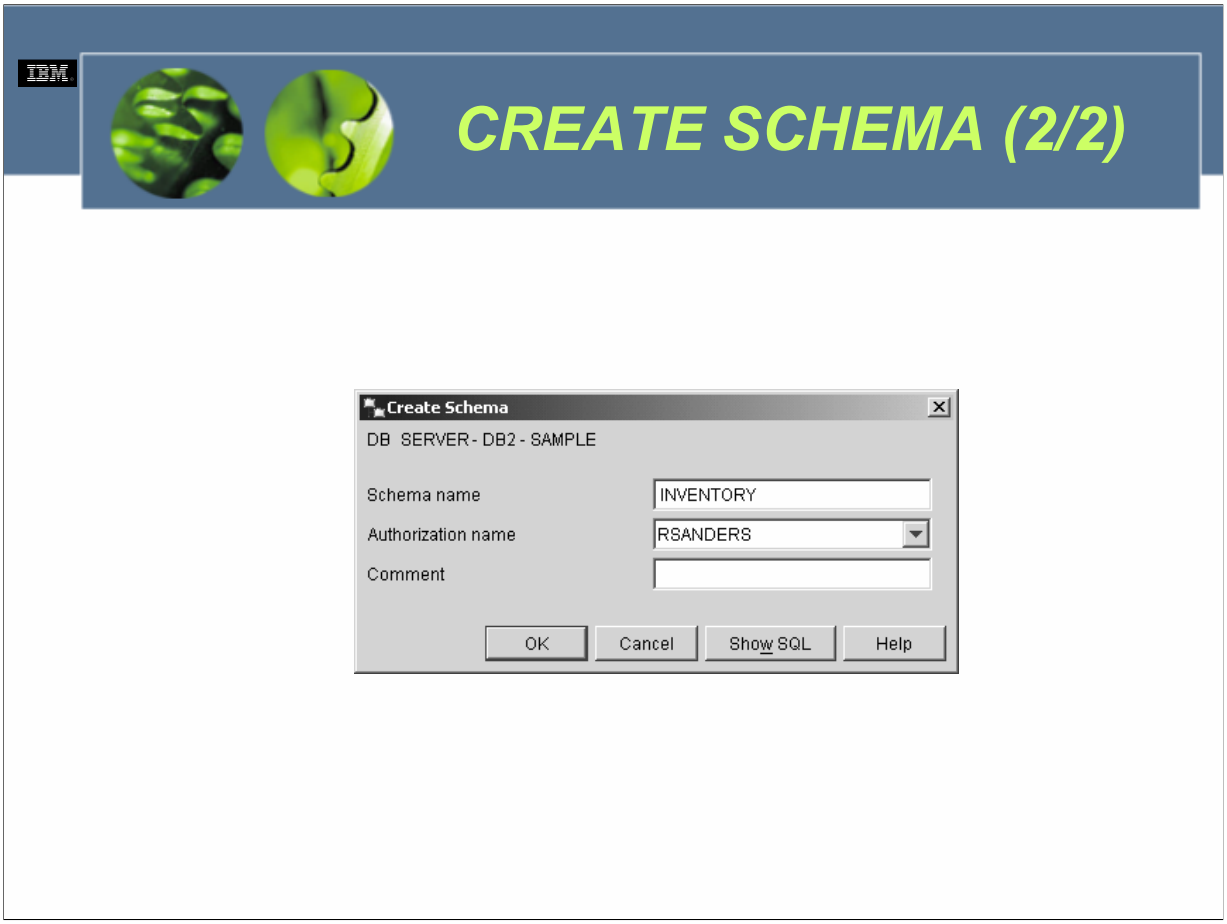
где:

SchemaName определяет имя, которое должно быть присвоено создаваемой схеме. *AuthorizationName* определяет пользователя, которому должно быть предоставлено владение создаваемой схемой.*SQLStatement* определяет один или несколько операторов SQL, которые должны быть выполнены вместе с оператором CREATE SCHEMA. (Действительны только следующие операторы SQL: CREATE TABLE, CREATE VIEW, CREATE INDEX, COMMENT ON и GRANT). Если указано имя схемы, но не предоставлено имя авторизации, владение новой схемой после ее создания передается ID авторизации пользователя, который исполнил оператор CREATE SCHEMA; если указано имя авторизации, но не указано имя схемы, новой схеме присваивается то же самое имя, которое использовалось в качестве имени авторизации.



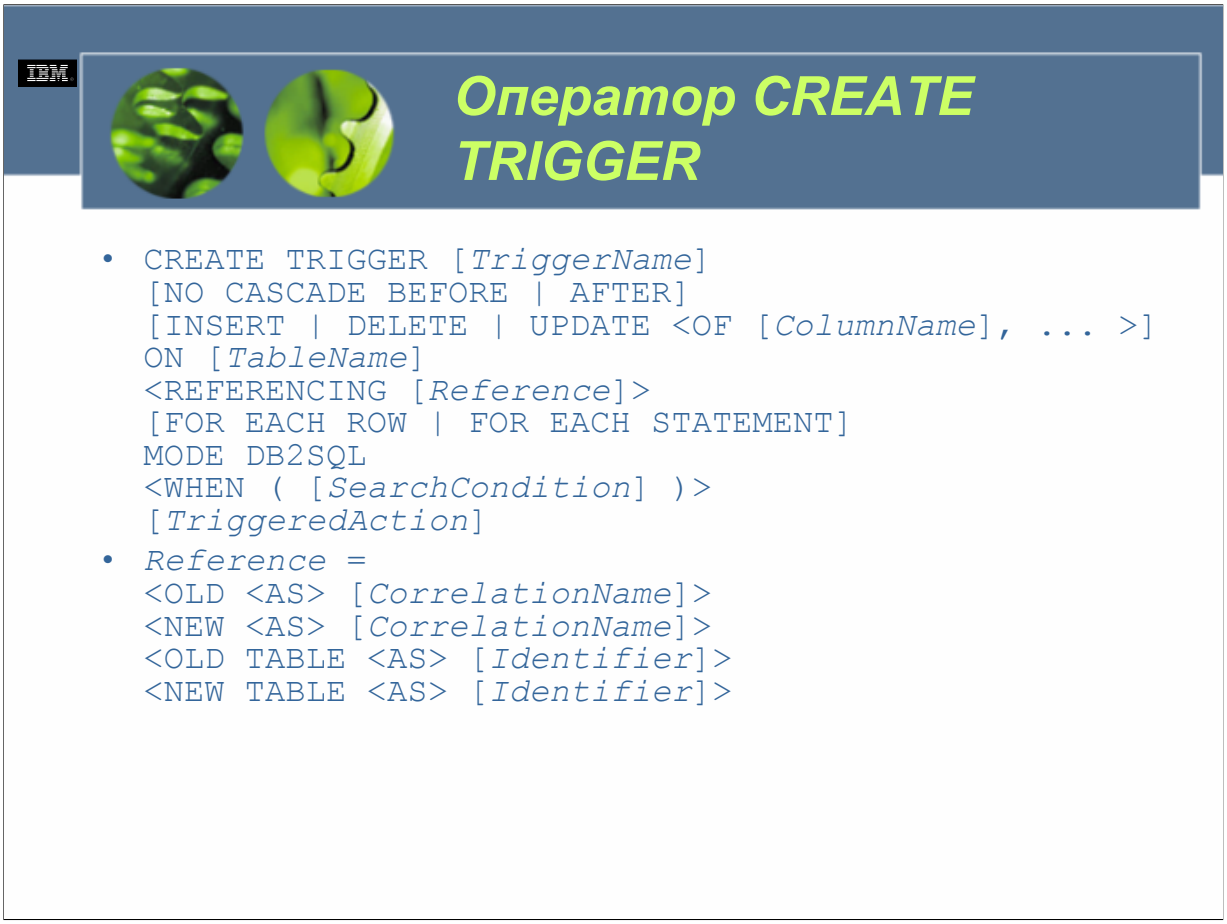
WCF03.0

Схемы можно также создавать с использованием диалога Создание схемы (Create Schema), который можно вызвать, выбрав соответствующее действие из меню *Схемы (Schemas)* центра управления.



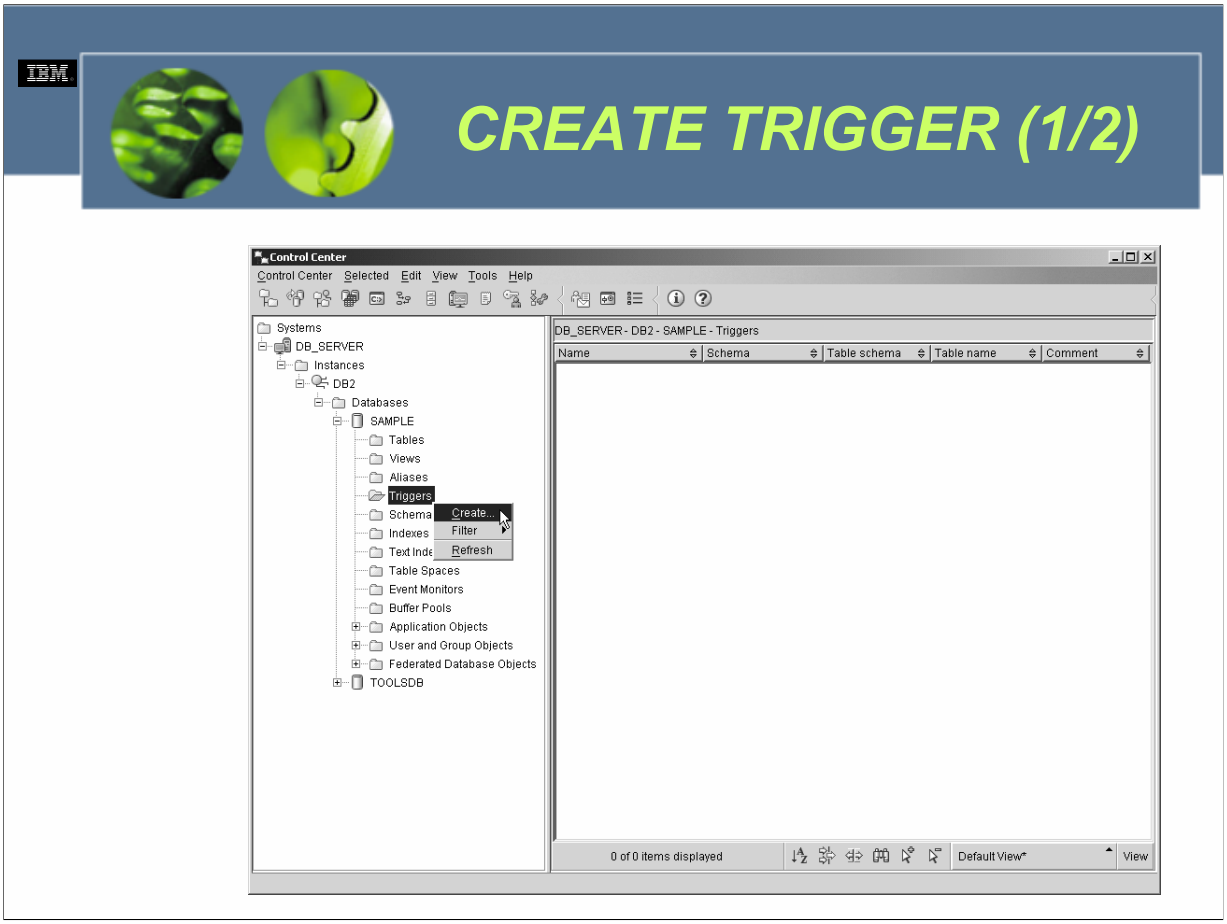
WCF03.0

Поскольку схемы можно создавать неявно путем создания объекта с именем новой схемы, вы, возможно, интересуетесь, почему кому-нибудь может понадобиться создавать схему явным образом, используя оператор CREATE SCHEMA или диалог Создание схемы. Основной причиной для явного создания схемы является обеспечение контроля доступа. У созданной явным образом схемы есть владелец, идентифицируемый либо по ID авторизации пользователя, выполнившего оператор CREATE SCHEMA, либо по ID авторизации, предоставленному для идентификации владельца при создании схемы. У владельца схемы есть полномочия создавать, изменять и уничтожать любой объект, хранящийся в схеме; уничтожать саму схему и предоставлять эти привилегии другим пользователям. С другой стороны, владельцем неявно созданных схем считается владелец “SYSIBM”. Любой пользователь может создать объект в неявно созданной схеме, а каждый объект в схеме контролируется пользователем, его создавшим. Более того, уничтожать неявно созданные схемы позволено лишь пользователям с полномочиями системного администратора (SYSADM) или администратора базы данных (DBADM). Таким образом, чтобы любой пользователь, кроме системного администратора или администратора базы данных, имел полный контроль над схемой, а также над всеми объектами базы данных, хранящимися в ней, схема должна быть создана явным образом.



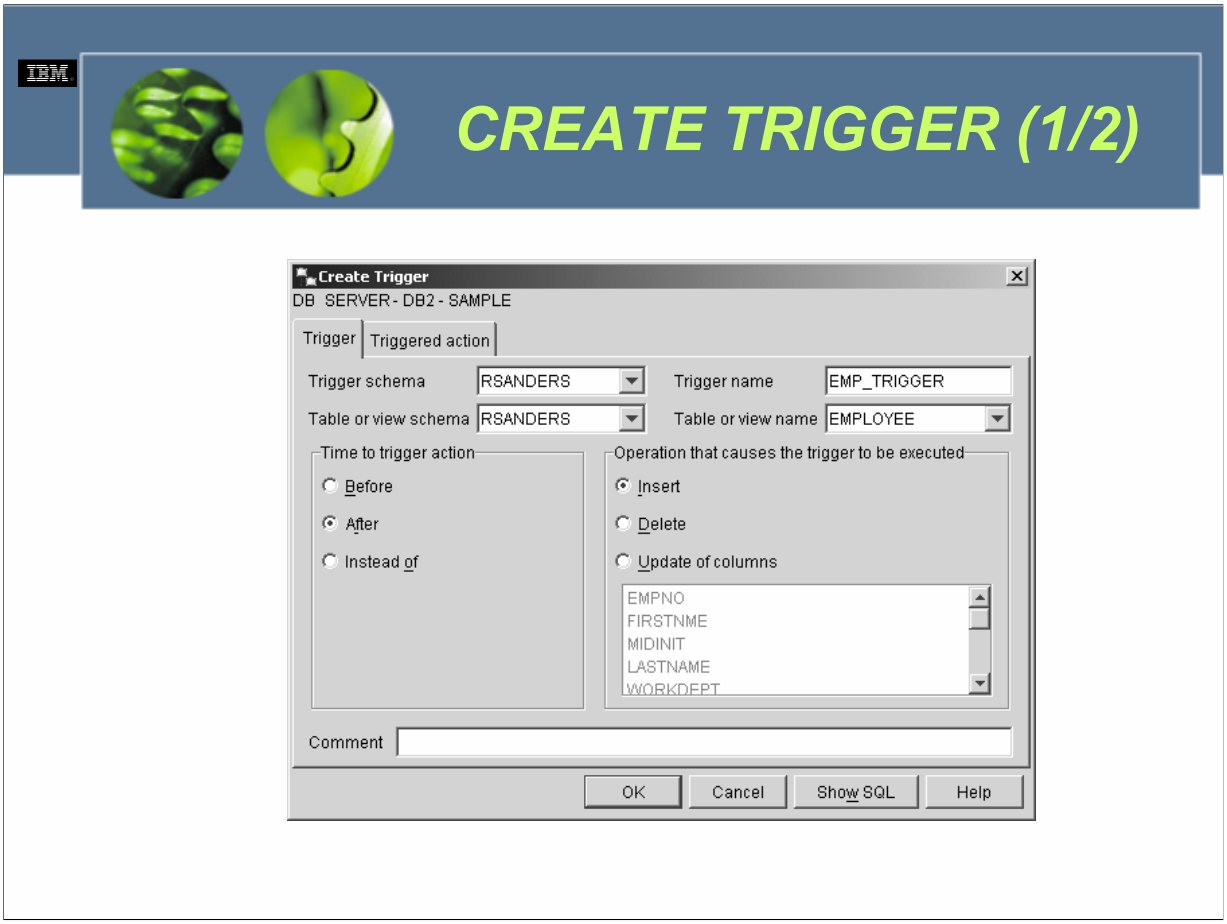
WCF03.0

TriggerName определяет имя, которое должно быть присвоено создаваемому триггеру.*ColumnName* определяет один или несколько столбцов в рабочей таблице триггера, значения которых должны быть изменены до выполнения действия триггера (*TriggeredAction*). *Reference* определяет одну или несколько переменных перехода и/или таблиц перехода, которые должны использоваться действием триггера (*TriggeredAction*). *CorrelationName* определяет имя, которое должно использоваться для идентификации определенной строки в рабочей таблице триггера, либо до того, как она будет изменена действием триггера (OLD <AS>), либо после того, как она будет изменена действием триггера (NEW <AS>). *Identifier* определяет имя, которое должно использоваться для идентификации временной таблицы, которая содержит набор строк из рабочей таблицы триггера, либо до того, как они были изменены действием триггера (OLD TABLE <AS>), либо после того, как они были изменены действием триггера (NEW TABLE <AS>). Каждый столбец, затронутый событием активации (операция вставки, изменения или удаления), может быть сделан доступным действию триггера путем специфицирования имени столбца соответствующим корреляционным именем или идентификатором таблицы. *SearchCondition* определяет условие поиска, которое дает в результате TRUE, FALSE или Unknown (не определено). Это условие используется для определения того, нужно ли выполнять действие триггера (*TriggeredAction*). *TriggeredAction* определяет действие, которое должно быть выполнено, когда триггер активируется. Действие триггера должно состоять из одного или нескольких операторов SQL; когда указано несколько операторов, перед первыми должны стоять ключевые слова BEGIN ATOMIC, за последним оператором должно следовать ключевое слово END, а каждый оператор между этими ключевыми словами должен завершаться точкой с запятой (;).



WCF03.0

Триггеры могут также создаваться с использованием диалога Создание триггера (Create Trigger), который можно вызвать, выбрав соответствующее действие из меню *Триггеры (Triggers)* центра управления.



WCF03.0

Прежде, чем может быть создан триггер, необходимо идентифицировать следующие компоненты:

•**Рабочая таблица**. Таблица, с которой взаимодействует триггер.

•**Событие триггера**. Операция SQL, которая заставляет триггер активироваться каждый раз при ее осуществлении для рабочей таблицы. Эта операция может быть операцией вставки, изменения или удаления.

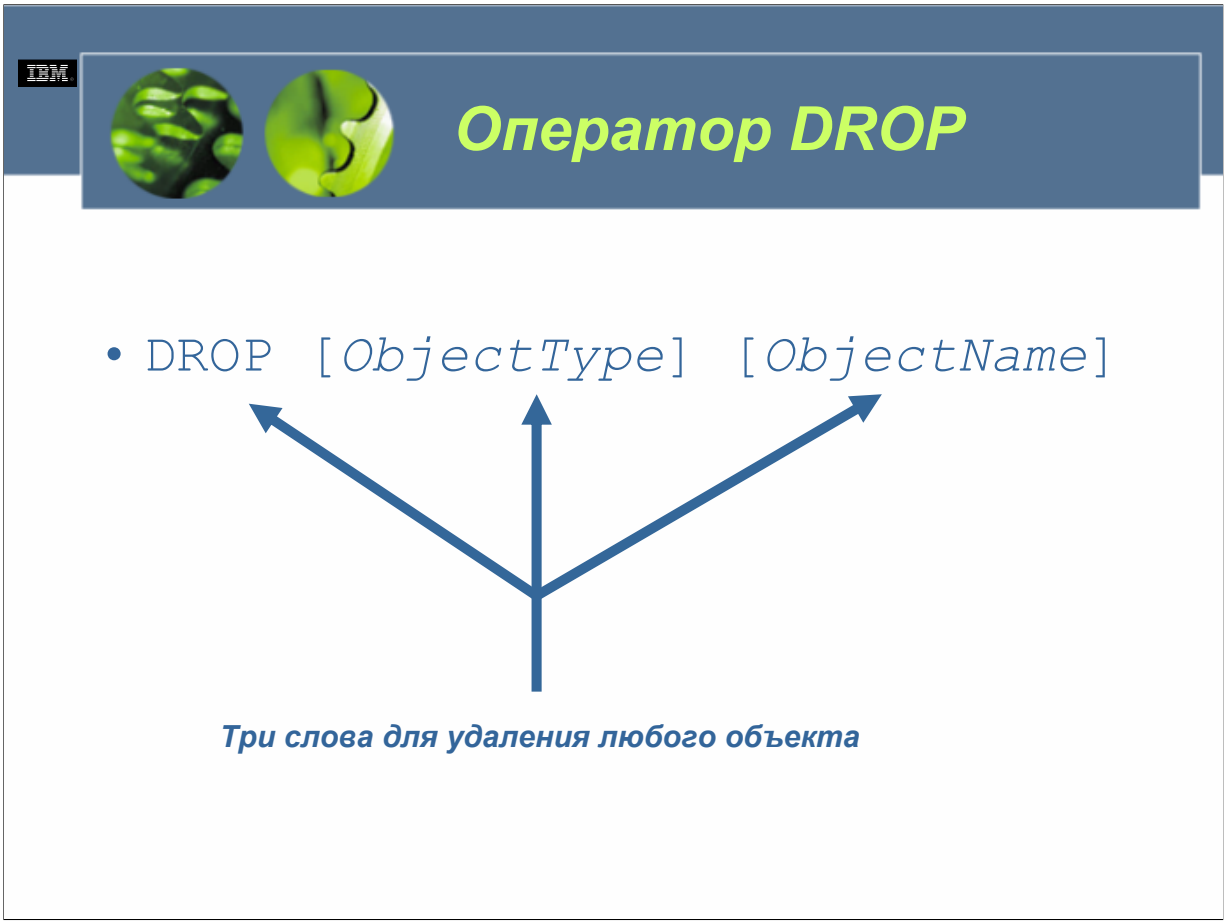
•**Время активации триггера**. Определяет, должен ли триггер активироваться до или после того, как произошло событие триггера. Триггер *before* будет активирован до возникновения события триггера; следовательно, будет можно увидеть новые значения данных до того, как они вставлены в рабочую таблицу. Триггер *after* будет активирован после возникновения события триггера; следовательно, можно видеть лишь те значения данных, которые уже были вставлены в рабочую таблицу. (Триггер before мог бы использоваться для перехвата и обработки нежелательных значений, тогда как триггер after мог бы использоваться для копирования значений введенных данных в другие таблицы или производные таблицы).

•**Набор изменяемых строк**. Строки рабочей таблицы, для которых была осуществлена вставка, изменение или удаление.

•**Кратность триггера**. Определяет, должны ли действия, которые будет осуществлять триггер, выполняться один раз для всей операции вставки, изменения или удаления, или они должны выполняться для каждой строки, затрагиваемой операцией вставки, изменения или удаления.

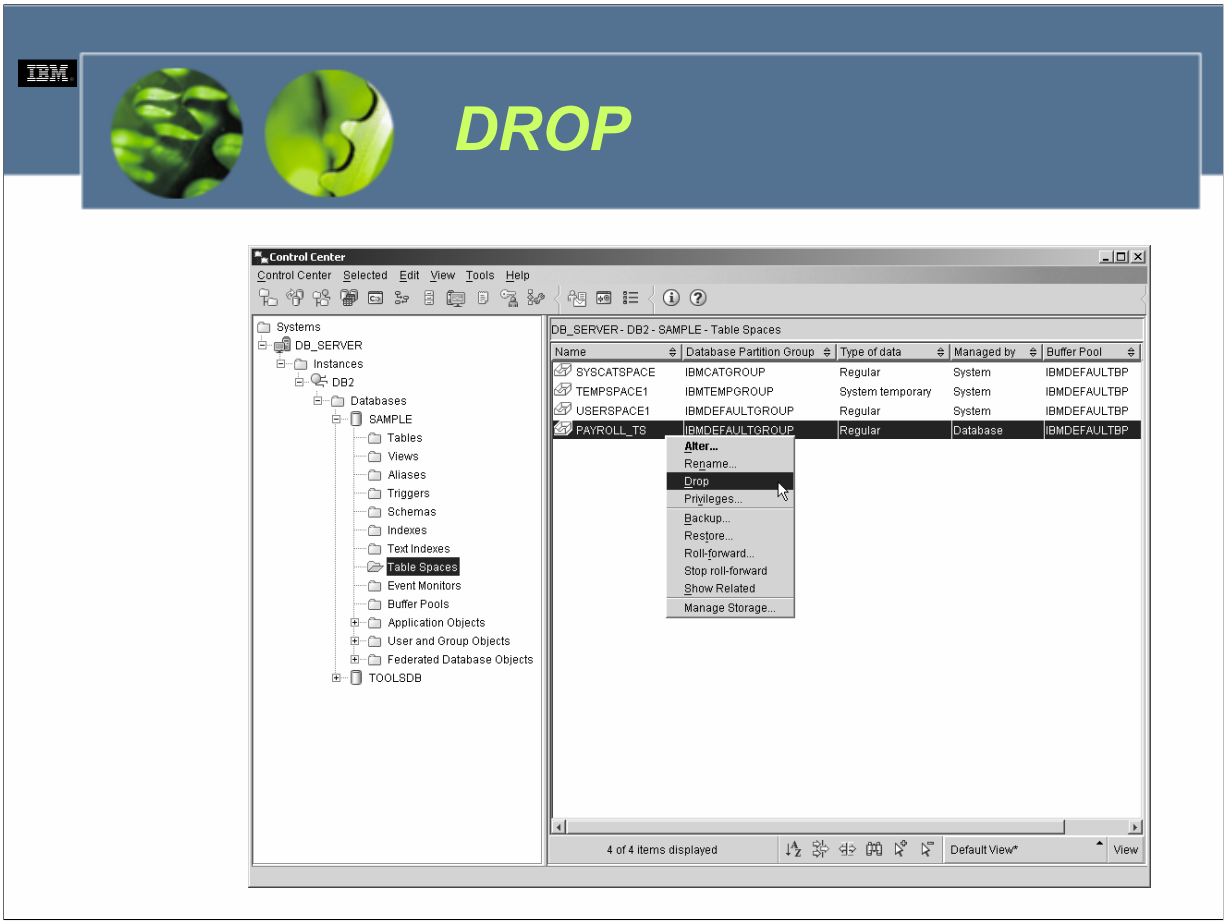
•**Действие триггера**. Необязательное условие поиска и набор операторов SQL, которые должны выполняться каждый раз, когда триггер активируется. (Если указано условие поиска, операторы SQL будут выполнены лишь в том случае, когда это условие истинно). Если триггер является триггером before, действие триггера может включать операторы, которые получают данные, устанавливают переменные перехода или сигнализируют о состояниях SQL. Если триггер является триггером after, действие триггера может включать операторы, которые получают данные, вставляют записи, изменяют записи, удаляют записи или сигнализируют о состояниях SQL.

Действия триггера могут ссылаться на значения в наборе изменяемых строк, используя так называемые *переменные перехода (transition variables)*. Переменные перехода используют имена столбцов в рабочей таблице, специфицируемые указанным именем, которое определяет, является ли ссылка ссылкой на оригинальное значение (до выполнения операции вставки, изменения или удаления) или ссылкой на новое значение (после выполнения операции вставки, изменения или удаления). Другим средством ссылки на значения в наборе изменяемых строк является использование *таблиц переходов (transition tables)*. Таблицы переходов также используют имена столбцов в рабочей таблице, но они позволяют рассматривать весь набор изменяемых строк в качестве таблицы. К сожалению, таблицы переходов могут использоваться лишь в триггерах after.



WCF03.0

ObjectType определяет тип удаляемого (уничтожаемого) объекта. (Действительные значения включают: BUFFERPOOL, TABLESPACE, TABLE, INDEX, VIEW, ALIAS, SCHEMA и TRIGGER). *ObjectName* указывает имя, присвоенное удаляемому объекту.



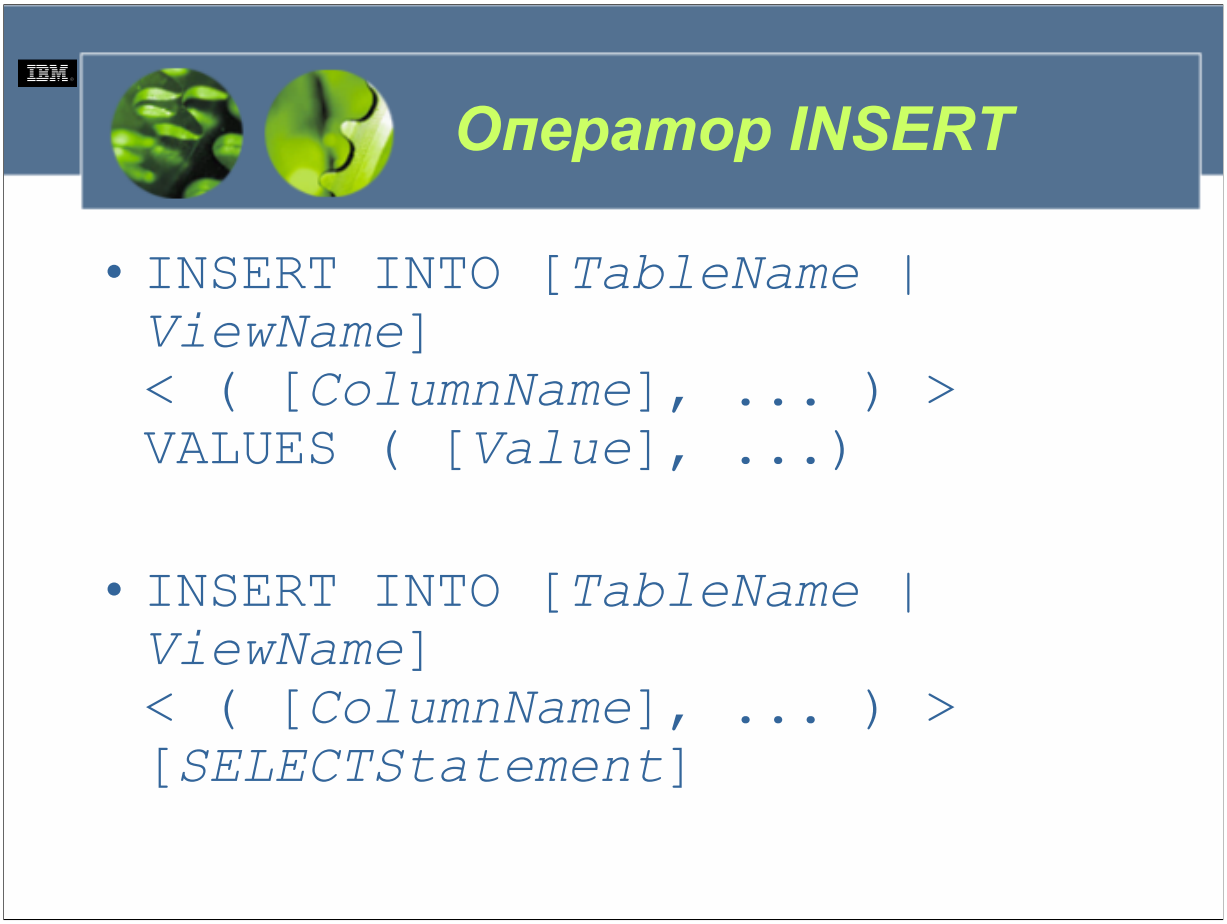
WCF03.0

Объекты базы данных можно также уничтожать из центра управления, выделив соответствующий объект и выбрав соответствующее действие из меню любого найденного объекта.

Важно иметь в виду, что когда объект уничтожается, его удаление может повлиять на другие объекты, которые зависят от его существования. В некоторых случаях, когда объект уничтожается, также удаляются все зависимые от него объекты (например, если уничтожается табличное пространство, содержащее одну или несколько таблиц, уничтожаются также все таблицы, размещенные в этом табличном пространстве, со всеми соответствующими данными). В других случаях объект не может быть уничтожен, если от его существования зависят другие объекты (например, схема может быть уничтожена лишь после того, как были уничтожены все объекты, которые были в этой схеме). Не стоит и говорить, что встроенные объекты, такие как таблицы и производные таблицы системных каталогов, не могут быть уничтожены.

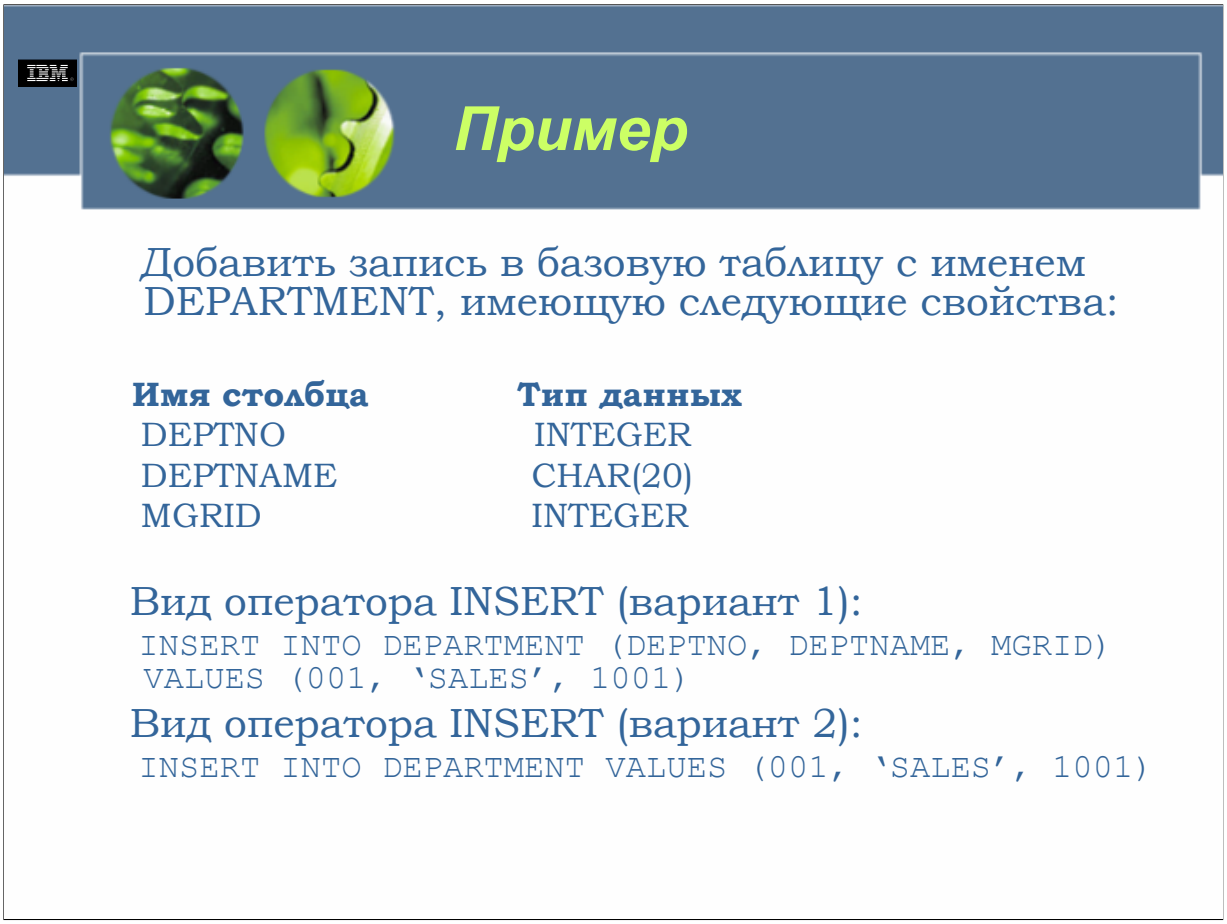
WCF03.0

Когда для определенной базы данных были созданы соответствующие объекты, они могут использоваться для хранения значений данных. И точно также, как есть набор операторов SQL, использующихся для определения и создания объектов, есть и набор операторов SQL, которые используются исключительно для хранения, преобразования, удаления и получения значений данных. Этот набор операторов называется операторами языка обработки данных (DML). Имеются четыре доступных оператора языка обработки данных: оператор INSERT, оператор UDPATE, оператор DELETE и оператор SELECT.



WCF03.0

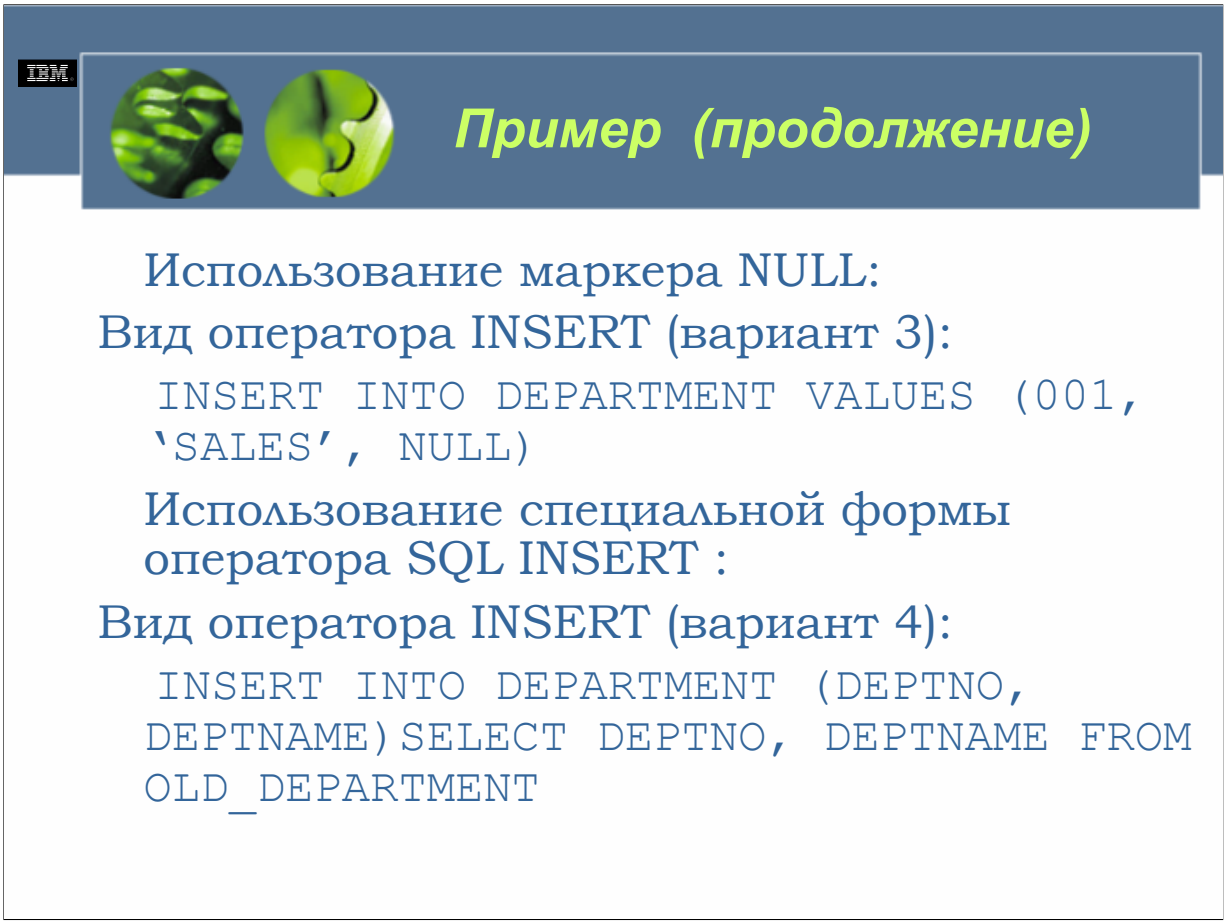
TableName определяет имя, присвоенное таблице, в которую должны быть добавлены данные. *ViewName* определяет имя, присвоенное обновляемой производной таблице, в которую должны быть добавлены данные. *ColumnName* определяет имя одного или нескольких столбцов, которым должны быть назначены значения данных, добавляемых к таблице/производной таблице. Каждое представленное имя должно идентифицировать существующий столбец в указанной таблице или обновляемой производной таблице.*Value* определяет одно или несколько значений данных, которые должны быть добавлены в указанные столбцы таблицы или обновляемой производной таблицы. *SELECTStatement* определяет оператор SQL SELECT, который при выполнении создаст значения данных, которые должны быть добавлены в указанные столбцы таблицы или обновляемой производной таблицы (путем получения данных из других таблиц и/или производных таблиц).



WCF03.0

Число значений, представленных в условии VALUES, должно равняться числу имен столбцов, представленных в соответствующем списке (вариант 1). Более того, предоставленные значения будут назначены указанным столбцам в том же порядке, в котором они представлены – другими словами, первое представленное значение будет присвоено первому столбцу, указанному в списке названий столбцов, второе представленное значение будет присвоено второму указанному столбцу и т.д. Каждое представленное значение должно также быть совместимым с типом данных столбца, которому присваивается значение.

Если значения предоставлены (в условии VALUES) для каждого столбца таблицы, список названий столбцов можно опустить. В этом случае первое представленное значение будет присвоено первому найденному столбцу таблицы, второе представленное значение будет присвоено второму найденному столбцу и т.д. Таким образом, строку данных, которая была добавлена в таблицу DEPARTMENT в предыдущем примере, можно также добавить, выполнив оператор INSERT (вариант 2).

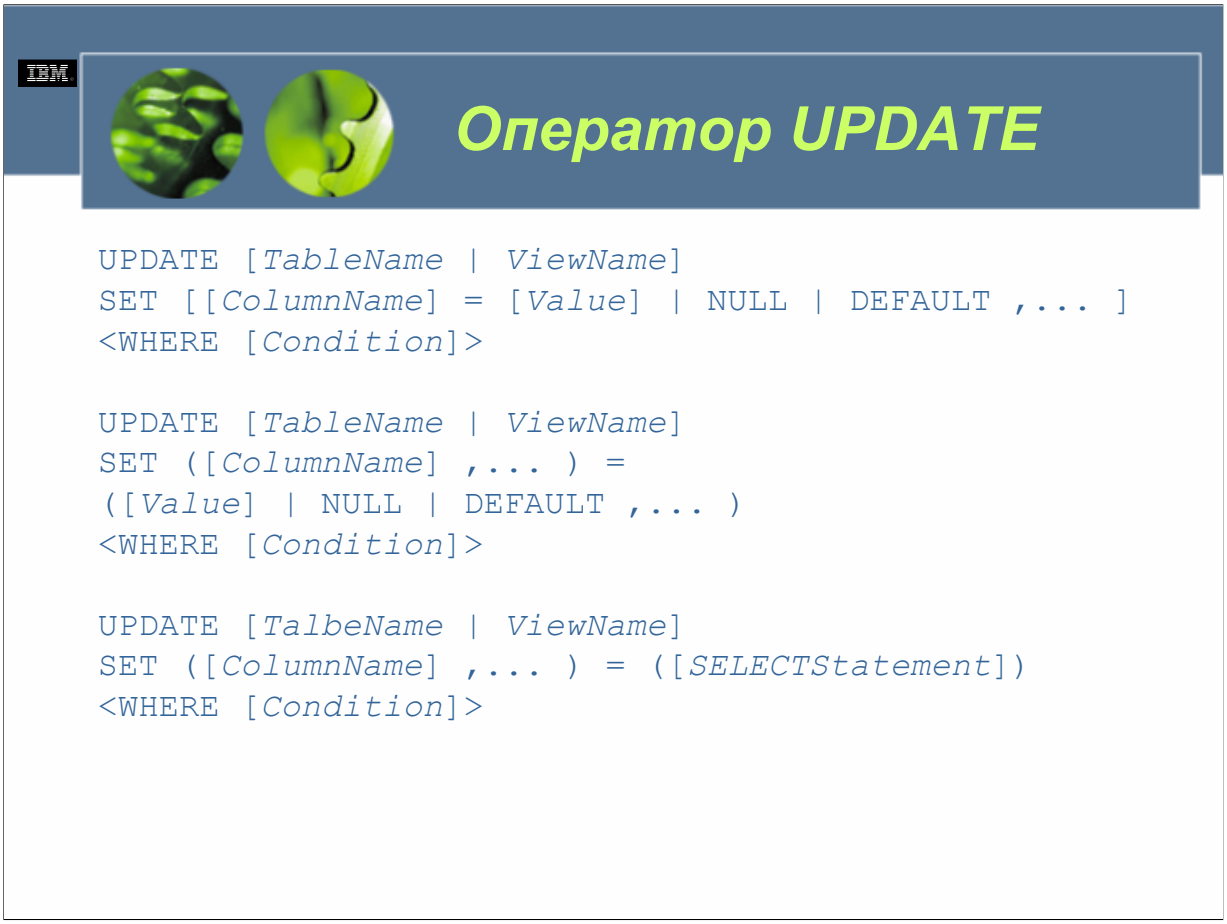


WCF03.0

Вместе с литеральными константами для назначения значений, которые должны присваиваться столбцам базовой таблицы, могут использоваться два специальных маркера. Первым из них является маркер DEFAULT, который используется для присваивания системного или пользовательского значения по умолчанию для столбца, определенного с ограничением WITH DEFAULT. Вторым является маркер NULL, который используется для присваивания значения NULL любому столбцу, который не был определен с ограничением NOT NULL. Чтобы добавить запись, содержащую значение NULL для столбца MGRID таблицы DEPARTMENT, следует использовать оператор INSERT (вариант 3).

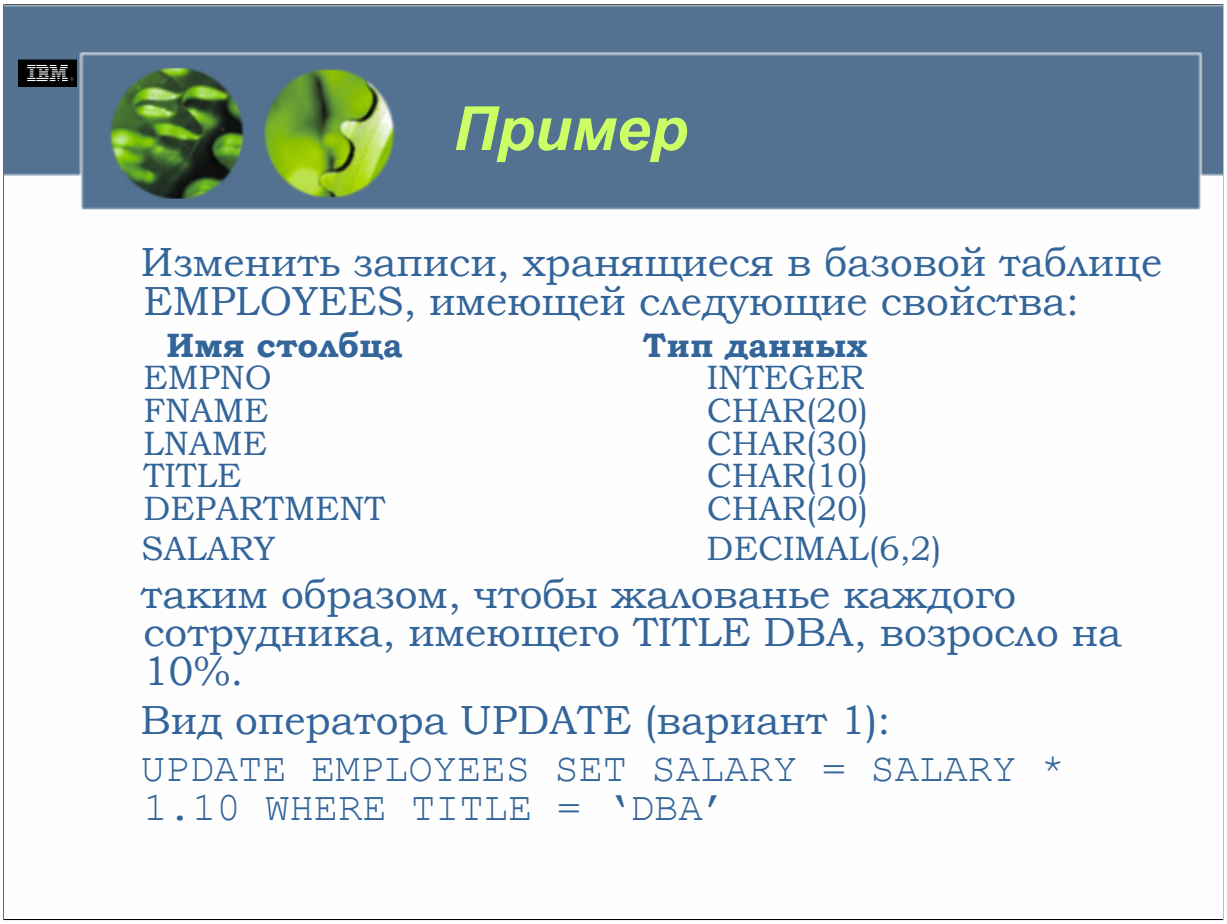
Используя специальную форму оператора SQL INSERT, результаты запроса также могут использоваться для предоставления значений для одного или нескольких столбцов в базовой таблице. В этой форме оператора INSERT на месте условия VALUES представлен оператор SELECT (известный как *подвыборка (subselect)*), и результаты оператора SELECT присваиваются соответствующим столбцам. (Эта форма оператора INSERT создает тип действия «вырезки и вставки», когда значения, полученные из одной базовой или производной таблицы, вставляются в другую). Число значений, возвращаемых подвыборкой, должно совпадать с числом столбцов, представленных в списке названий столбцов (или с числом столбцов в таблице, если список названий столбцов не предоставлен), а порядок присваивания тот же самый, который использовался при представлении значений литеральных констант в условии VALUES. В этом случае оператор INSERT записывается в варианте 4.

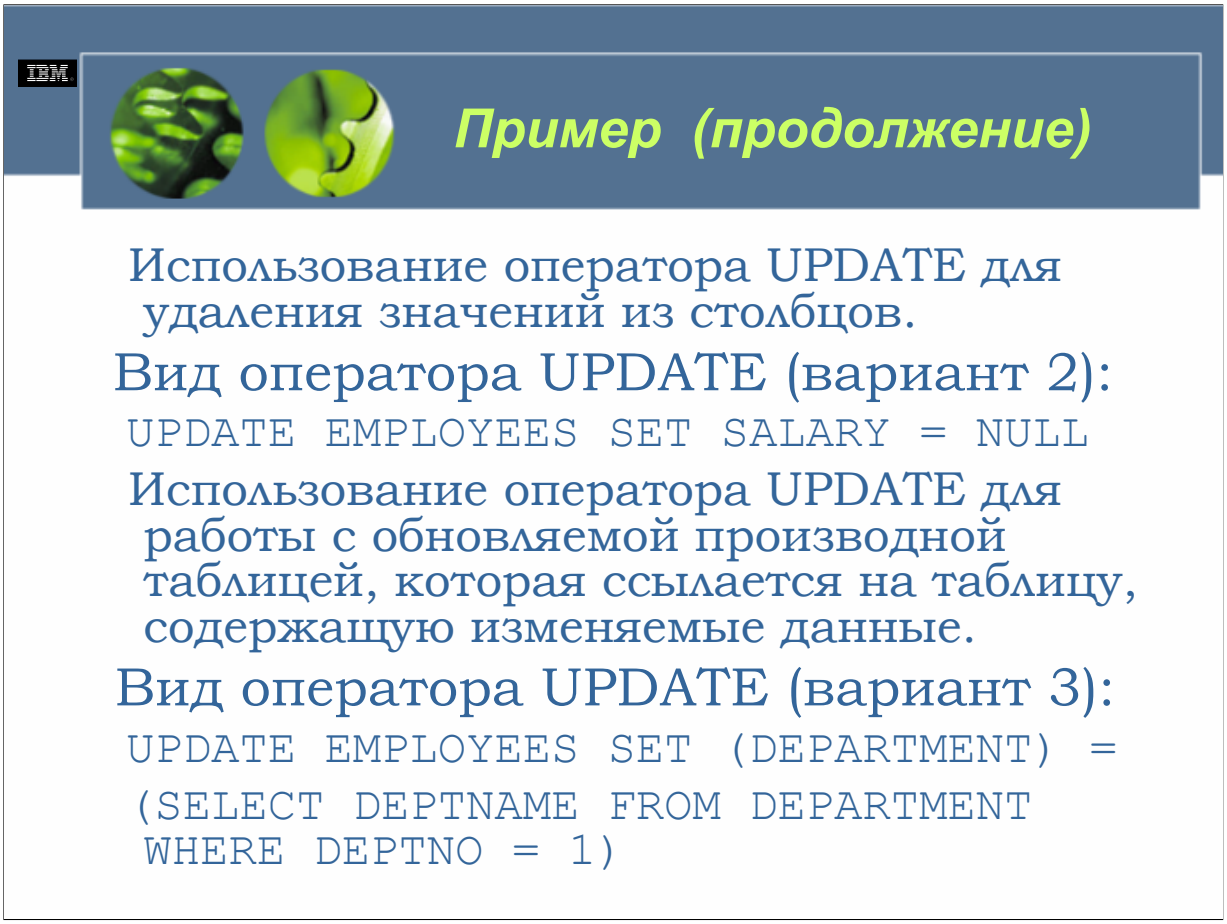
Оператор INSERT, использованный в варианте 4, не предоставлял значений для каждого столбца в таблице DEPARTMENT. Если необходимо вводить в таблицу не полные, а частичные записи, то такие операции можно осуществить, перечислив в списке названий столбцов лишь те столбцы, для которых есть данные, и предоставив соответствующие значения, использовав либо условие VALUES, либо подвыборку. Однако для того, чтобы такой оператор INSERT выполнялся корректно, все столбцы в таблице, в которую вставляется запись, которые не появляются в предоставленном списке названий столбцов, должны либо принимать пустые (null) значения, либо должны иметь определенное ограничение значения по умолчанию. В противном случае оператор INSERT завершится неудачей.



WCF03.0

TableName определяет имя, присвоенное таблице, которая содержит изменяемые данные. *ViewName* определяет имя, присвоенное обновляемой производной таблице, содержащей модифицируемые данные.*ColumnName* определяет имя одного или нескольких столбцов, которые содержат изменяемые данные. Каждое предоставленное имя должно обозначать существующий столбец в указанной таблице или обновляемой производной таблице. *Value* определяет одно или несколько значений данных, которые должны использоваться для замещения существующих значений в указанных столбцах. *SELECTStatement* определяет оператор SQL SELECT, который при выполнении создаст значения данных, которые должны использоваться для замещения существующих данных в указанных столбцах (путем получения данных из других таблиц и/или производных таблиц). *Condition* определяет критерий поиска, который должен использоваться для обнаружения одной или нескольких определенных строк, значения данных которых должны быть изменены. (Это условие кодируется аналогично условию WHERE, которое может использоваться с оператором SQL SELECT; мы рассмотрим условие WHERE и его предикаты позже). Если условие не указано, операция изменения будет осуществлена для каждой строки, найденной в указанной таблице или обновляемой производной таблице.



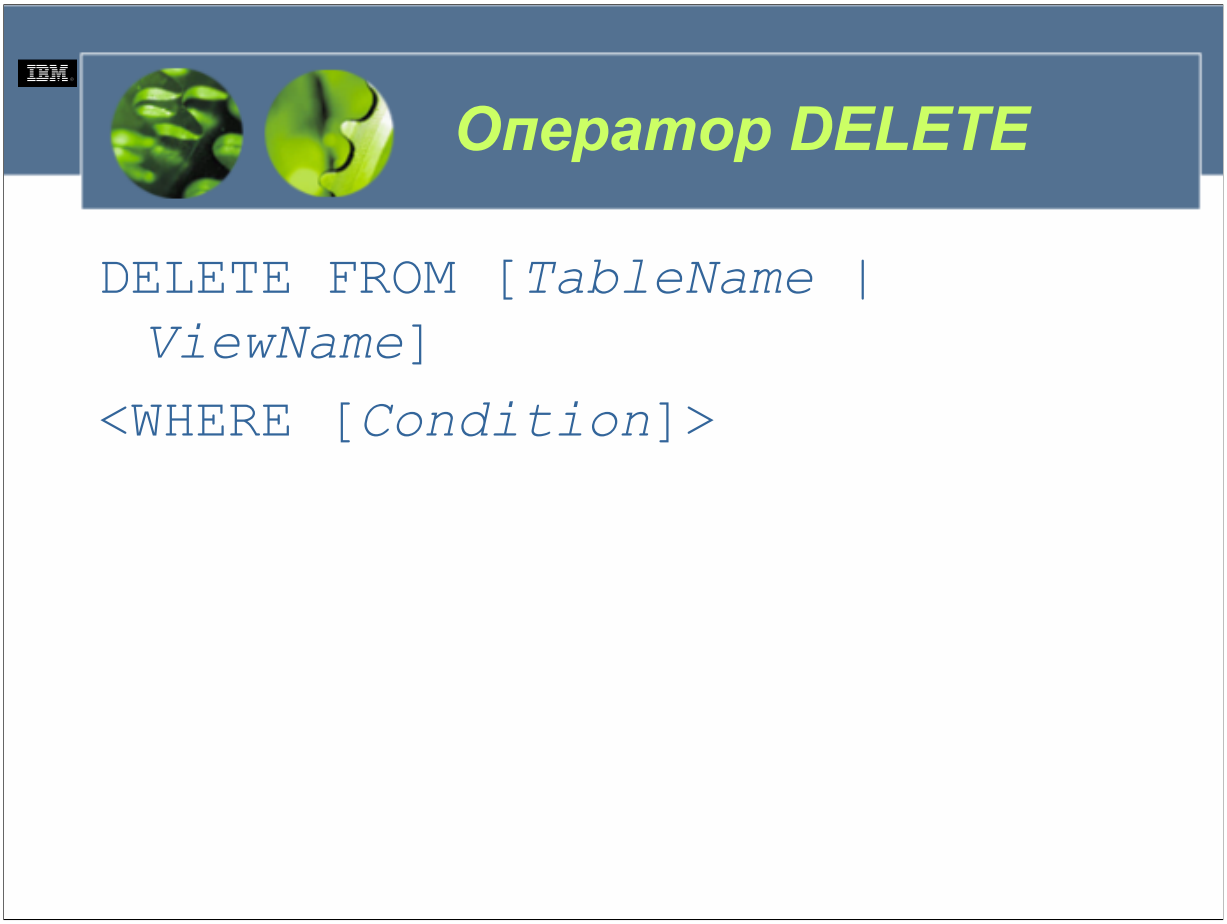


WCF03.0

Аналогично оператору INSERT, оператор UPDATE может также работать непосредственно с таблицей, содержащей изменяемые значения, или он может работать с обновляемой производной таблицей, которая ссылается на таблицу, содержащую изменяемые данные. Сходным образом результаты запроса или подзапроса могут использоваться для предоставления значений для одного или нескольких столбцов, указанных в предоставленном списке имен столбцов. (Эта форма оператора UPDATE создает тип действия «вырезания и вставки», когда значения, полученные из одной базовой или производной таблицы, используются для изменения значений, хранящихся в другой). Число значений, возвращаемых подвыборкой, должно соответствовать числу столбцов, представленных в указанном списке названий столбцов. Таким образом, можно изменить значение, присвоенное столбцам DEPARTMENT каждой записи в таблице EMPLOYEES с использованием результатов запроса, выполнив оператор UPDATE (вариант 2)

Операции изменения можно проводить одним из двух способов: путем осуществления *поискового изменения (searched update)* или путем осуществления *позиционного изменения (positioned update)*. Для осуществления позиционного изменения сначала должен быть создан, открыт и установлен на изменяемую строку указатель (cursor). Затем оператор UPDATE, который должен использоваться для изменения одного или нескольких значений данных, должен содержать условие WHERE CURRENT OF [*CursorName*] (*CursorName* обозначает используемый указатель). Из-за своей дополнительной сложности позиционные изменения обычно осуществляются приложениями встроенного SQL.

Необходимо предусматривать соответствующее условие WHERE каждый раз при использовании оператора UPDATE. Отказ сделать это может вызвать осуществление операции изменения для каждой строки, находящейся в таблице или обновляемой производной таблице.



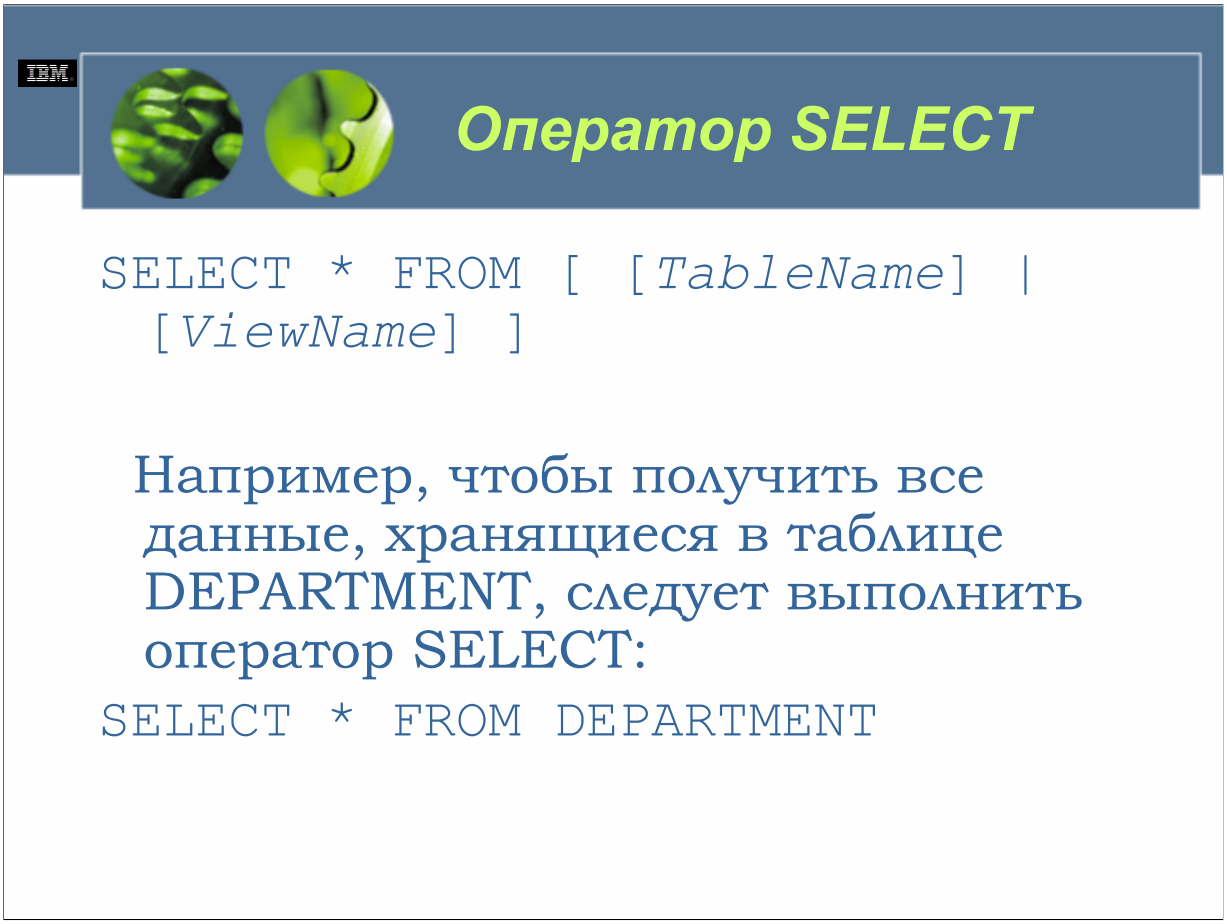
WCF03.0

TableName определяет имя, назначенное таблице, из которой должны быть удалены данные. *ViewName* определяет имя, назначенное обновляемой производной таблице, из которой должны быть удалены данные. *Condition* определяет критерий поиска, который должен использоваться для обнаружения одной или нескольких определенных строк, которые должны быть удалены. (Это условие кодируется аналогично условию WHERE, использующемуся с оператором SQL SELECT). Если условие не указано, операция удаления будет осуществлена для каждой строки, находящейся в указанной таблице или обновляемой производной таблице.

WCF03.0

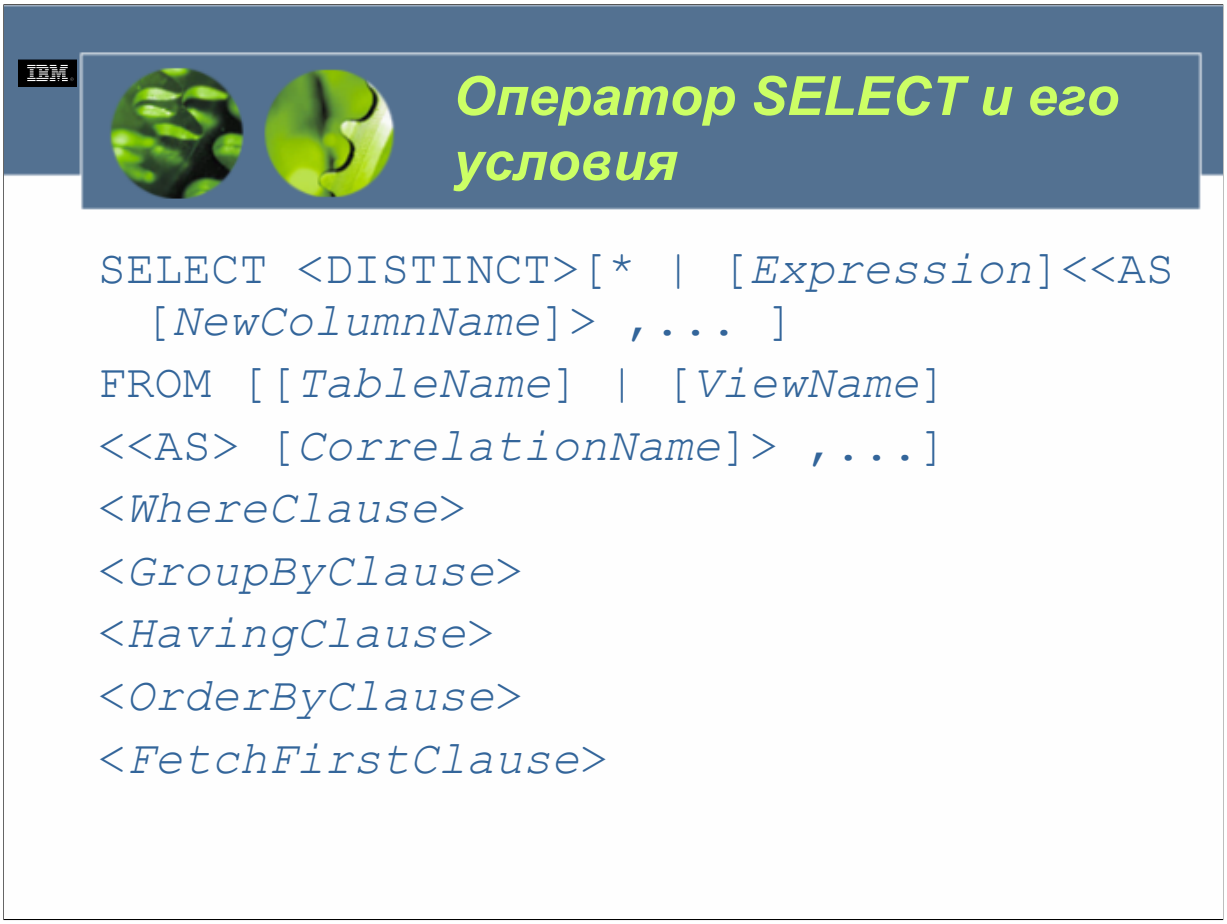
Аналогично операциям изменения, операции удаления могут проводиться одним из двух способов: в виде операций поискового *удаления (searched delete)* и в виде операций *позиционного удаления (positioned delete)*. Для осуществления позиционного удаления сначала должен быть создан, открыт и установлен на удаляемую строку указатель (cursor). Затем оператор DELETE, использующийся для удаления строки, должен содержать условие WHERE CURRENT OF [*CursorName*] (*CursorName* обозначает использующийся указатель). Из-за своей дополнительной сложности операции позиционного удаления обычно осуществляются приложениями встроенного SQL.

Важно помнить, что пропуск условия WHERE в операторе SQL DELETE вызывает применение операции удаления ко всем строкам в указанной таблице или производной таблице. Необходимо каждый раз предоставлять вместе с оператором DELETE условие WHERE, если только вы не хотите явным образом стереть все данные, хранящиеся в таблице.



WCF03.0

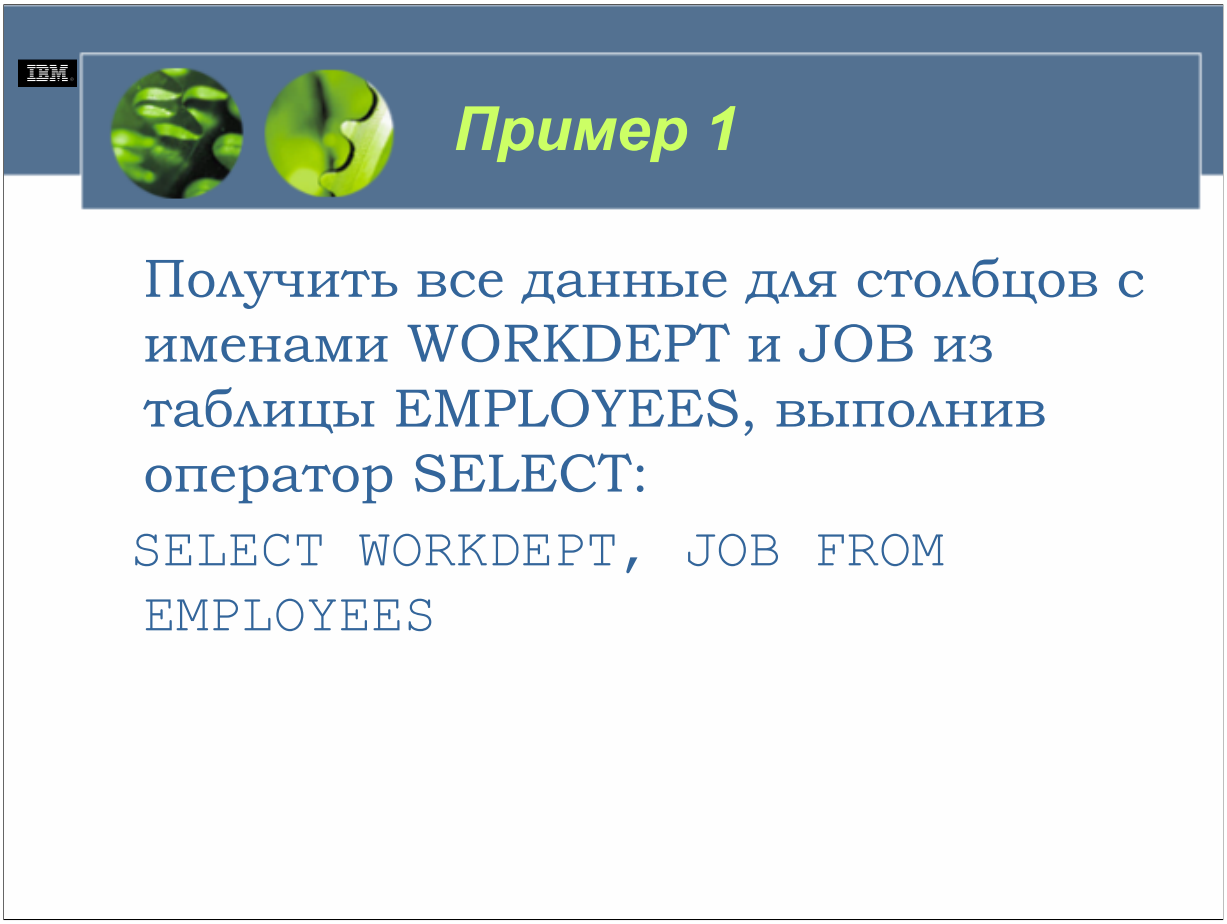
TableName определяет имя, назначенное таблице, из которой должны быть получены данные. *ViewName* определяет имя, назначенное производной таблице, из которой должны быть получены данные.



WCF03.0

Expression определяет один или несколько столбцов, для которых должны быть возвращены значения при выполнении оператора SELECT. Указанное для этой опции значение может быть любым действительным элементом языка SQL; однако обычно используются имена столбцов соответствующих таблиц или производных таблиц. *NewColumnName* определяет новое имя столбца, которое должно использоваться вместо имени столбца соответствующей таблицы или производной таблицы, указанного в наборе результатов данных, возвращаемых оператором SELECT. *TableName* указывает имена одной или нескольких таблиц, из которых должны быть получены данные. *ViewName* указывает имена одной или нескольких производных таблиц, из которых должны быть получены данные. *CorrelationName* определяет сокращенное имя, которое может использоваться при ссылке на таблицу или производную таблицу, с которой связано это сокращенное имя в любом из условий оператора SELECT. *WhereClause* определяет условие WHERE, которое должно использоваться с оператором SELECT. *GroupByClause* определяет условие GROUP BY, которое должно использоваться с оператором SELECT. *HavingClause* определяет условие HAVING, которое должно использоваться с оператором SELECT. *OrderByClause* определяет условие ORDER BY, которое должно использоваться с оператором SELECT. *FetchFirstClause* определяет условие FETCH FIRST, которое должно использоваться с оператором SELECT.

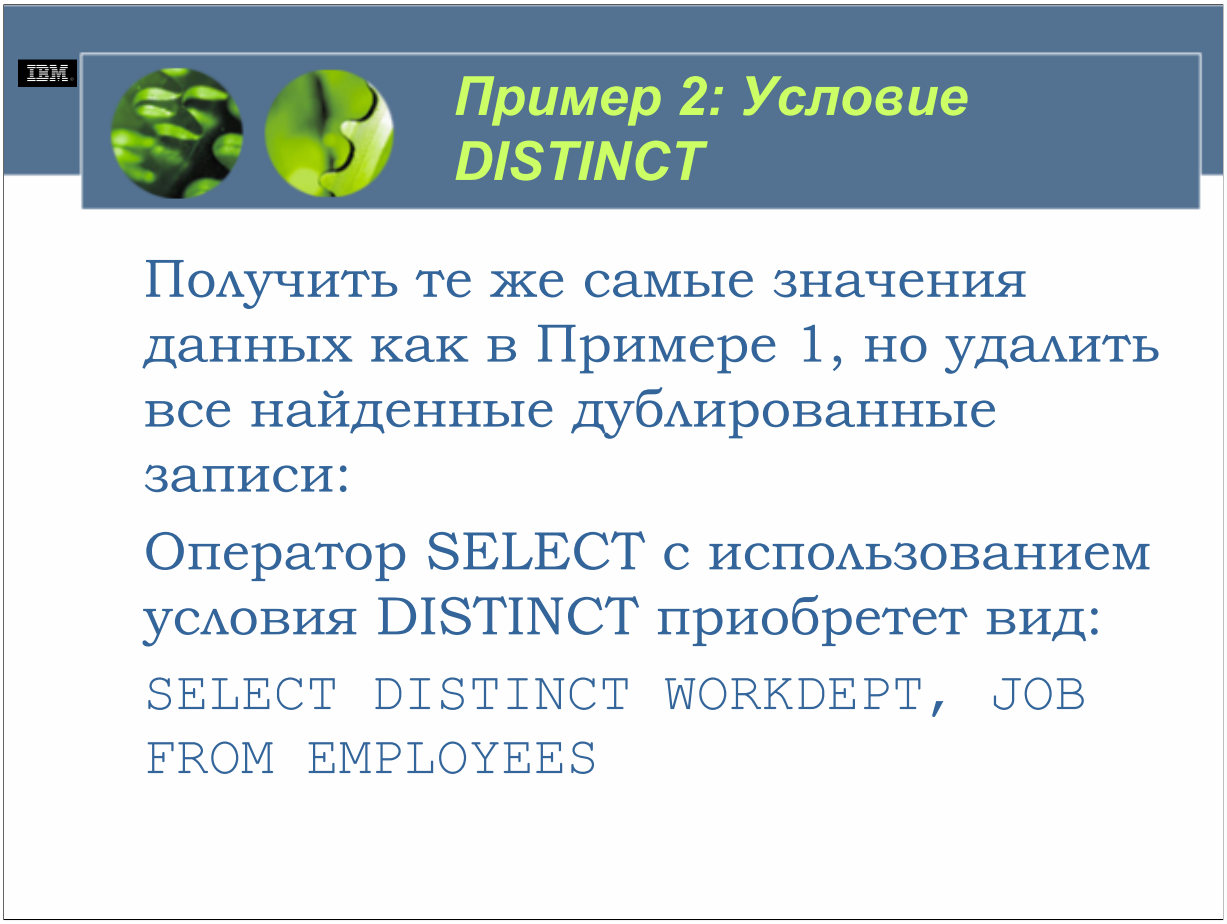
Если с оператором SELECT указано условие DISTINCT, из возвращаемого финального набора результатов данных удаляются дублированные строки. (Две строки считаются дублирующими друг друга лишь в том случае, если каждое значение в первой строке идентично соответствующему значению во второй строке. С целью определения того, являются ли две строки идентичными, пустые значения считаются равными). Однако, если используется условие DISTINCT, произведенный результирующий набор данных не должен содержать столбцов, содержащих данные типа LONG VARCHAR, LONG VARGRAPHIC, DATALINK, BLOB, CLOB или DBCLOB.



WCF03.0

Примерный набор результатов данных:

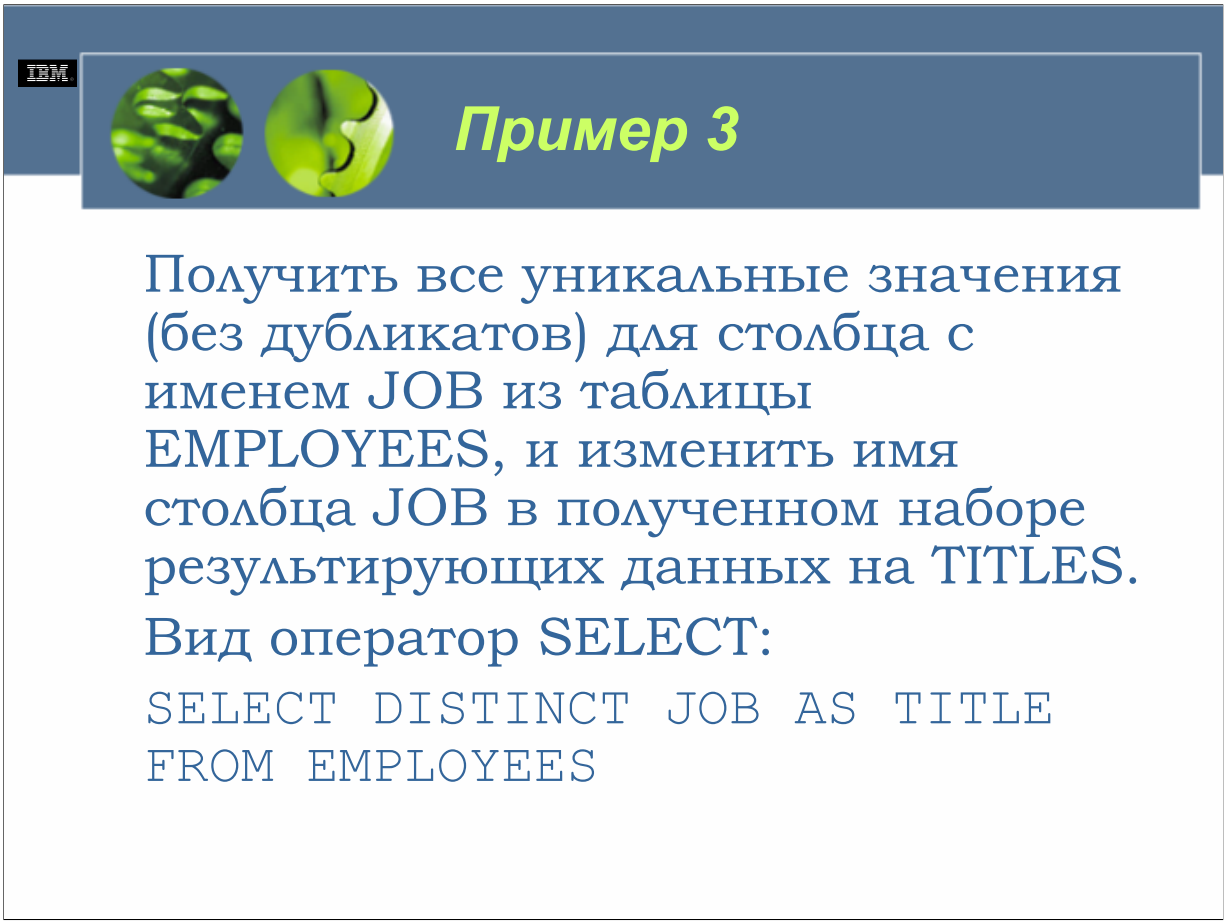
```
WORKDEPT JOB
-------- --------
A00 PRES                D11 DESIGNER            D21 CLERK
B01 MANAGER             D 11 DESIGNERE          11 OPERATOR
C01 MANAGER             D11 DESIGNER            E11 OPERATOR
E01 MANAGER             D11 DESIGNER            E11 OPERATOR
D11 MANAGER             D11 DESIGNERE           11 OPERATOR
D21 MANAGER             D11 DESIGNER            E21 FIELDREP
E11 MANAGER             D11 DESIGNERE           21 FIELDREP
E21 MANAGER             D11 DESIGNERE           21 FIELDREP
A00 SALESREP            D21 CLERK
A00 CLERK               D21 CLERK
C01 ANALYST             D21 CLERK
C01 ANALYST             D21 CLERK

32 record(s) selected.
```



WCF03.0

Примерный набор результатов данных:

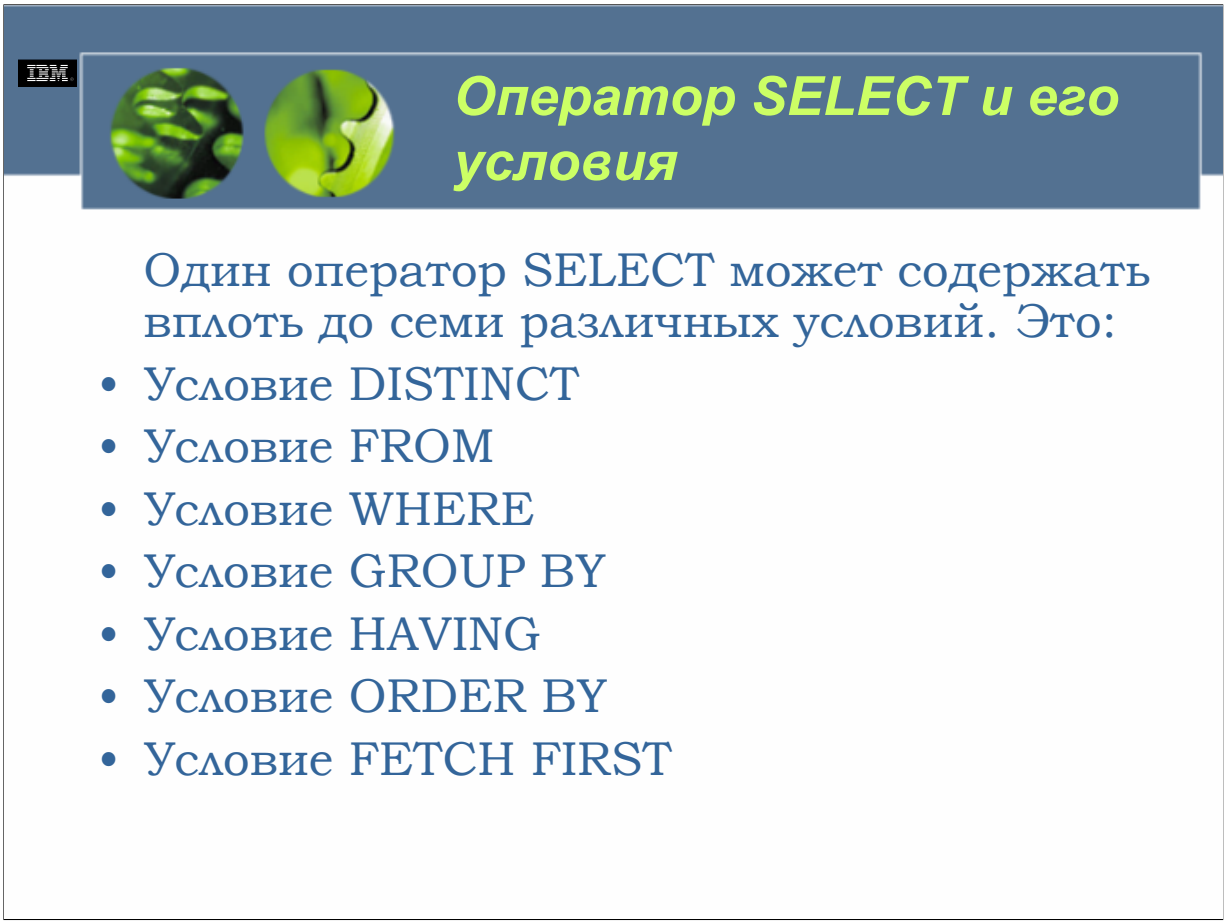
```
WORKDEPT JOB
-------- --------
C01 ANALYST             D21 MANAGER
A00 CLERK               E01 MANAGER
D21 CLERK               E11 MANAGER
D11 DESIGNER            E21 MANAGER
E21 FIELDREP            E11 OPERATOR
B01 MANAGER             A00 PRES
C01 MANAGER             A00 SALESREP
D11 MANAGER
15 record(s) selected.
```



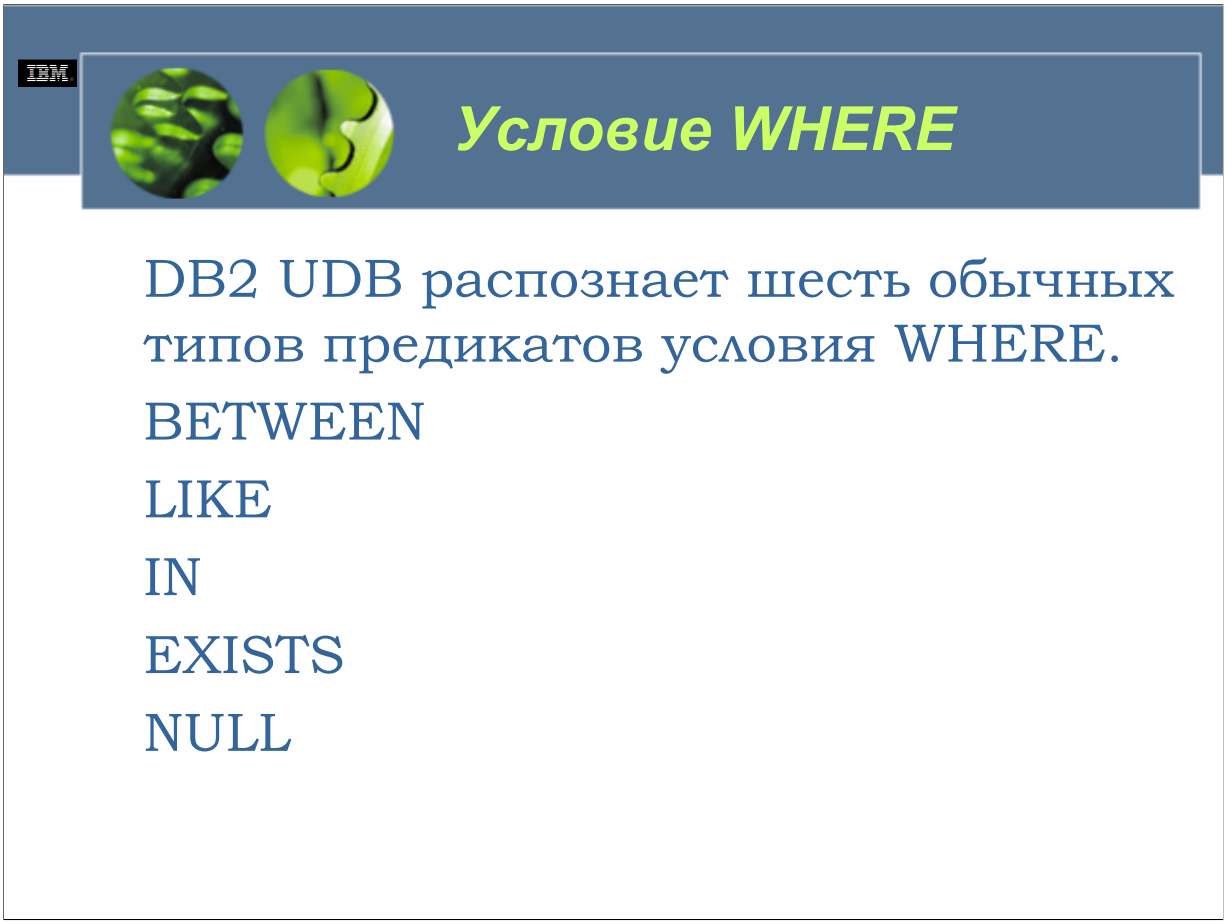
WCF03.0

Примерный набор результатов данных:

```
TITLE
--------
ANALYST
CLERK
DESIGNER
FIELDREP
MANAGER
OPERATOR
PRES
SALESREP
  8 record(s) selected.
```



Эти условия обрабатываются в приведенном порядке.

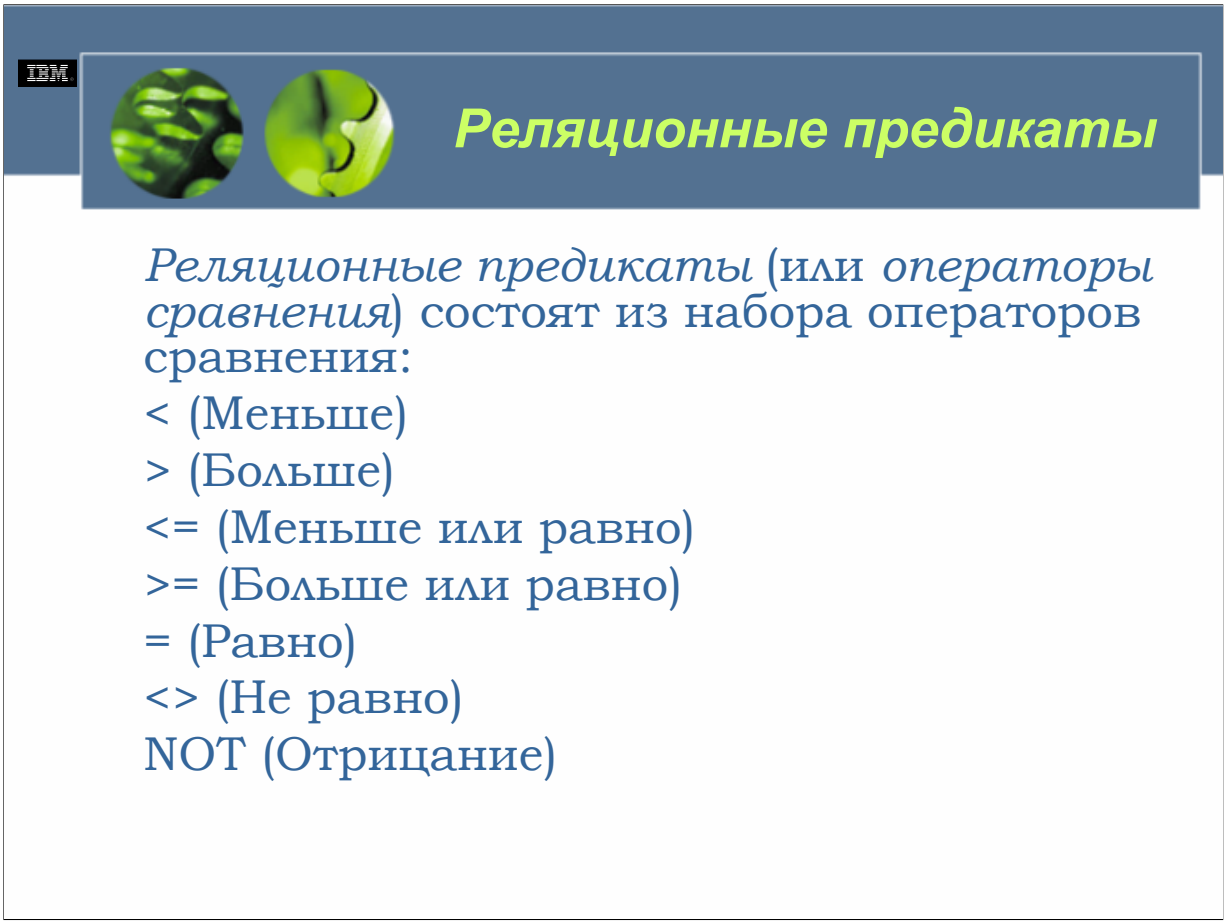


WCF03.0

Условие WHERE используется для того, чтобы сообщить менеджеру баз данных DB2, как выбрать строки, которые должны быть возвращены в наборе результирующих данных, выдаваемых в ответ на запрос. Когда условие WHERE указано, за ним следует *условие поиска*, которое в сущности представляет собой простую проверку, применение которой к строке данных дает значение TRUE, FALSE или Unknown. Если результат проверки дает TRUE, строка должна быть возвращена в создаваемом наборе результирующих данных; если результат проверки дает FALSE или Unknown, строка пропускается.

Условие поиска WHERE составлено из одного или нескольких предикатов, которые используются для сравнения содержания столбца с константным значением, содержания столбца с содержанием другого столбца из той же самой таблицы, или содержания столбца в одной таблице с содержанием столбца в другой таблице (называя лишь некоторые).

Каждый из этих предикатов может использоваться отдельно, или один или несколько из них могут комбинироваться с использованием скобок или булевых операторов, таких как AND, OR и NOT.

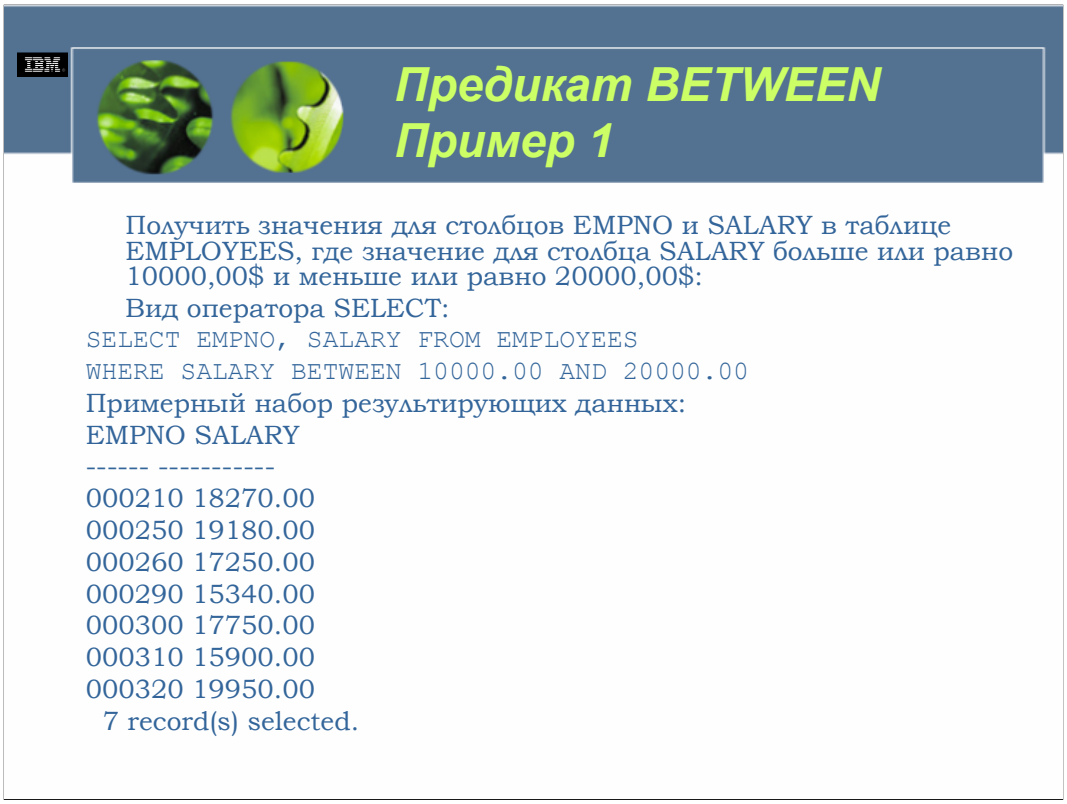


WCF03.0

Операторы сравнения используются для определения отношения сравнения между содержанием столбца и константным значением, содержанием двух столбцов из одной и той же таблицы или содержания столбца в одной таблице с содержанием столбца из другой таблицы. Обычно реляционные предикаты используются для включения или исключения определенных строк их набора результирующих данных, созданных в ответ на запрос.

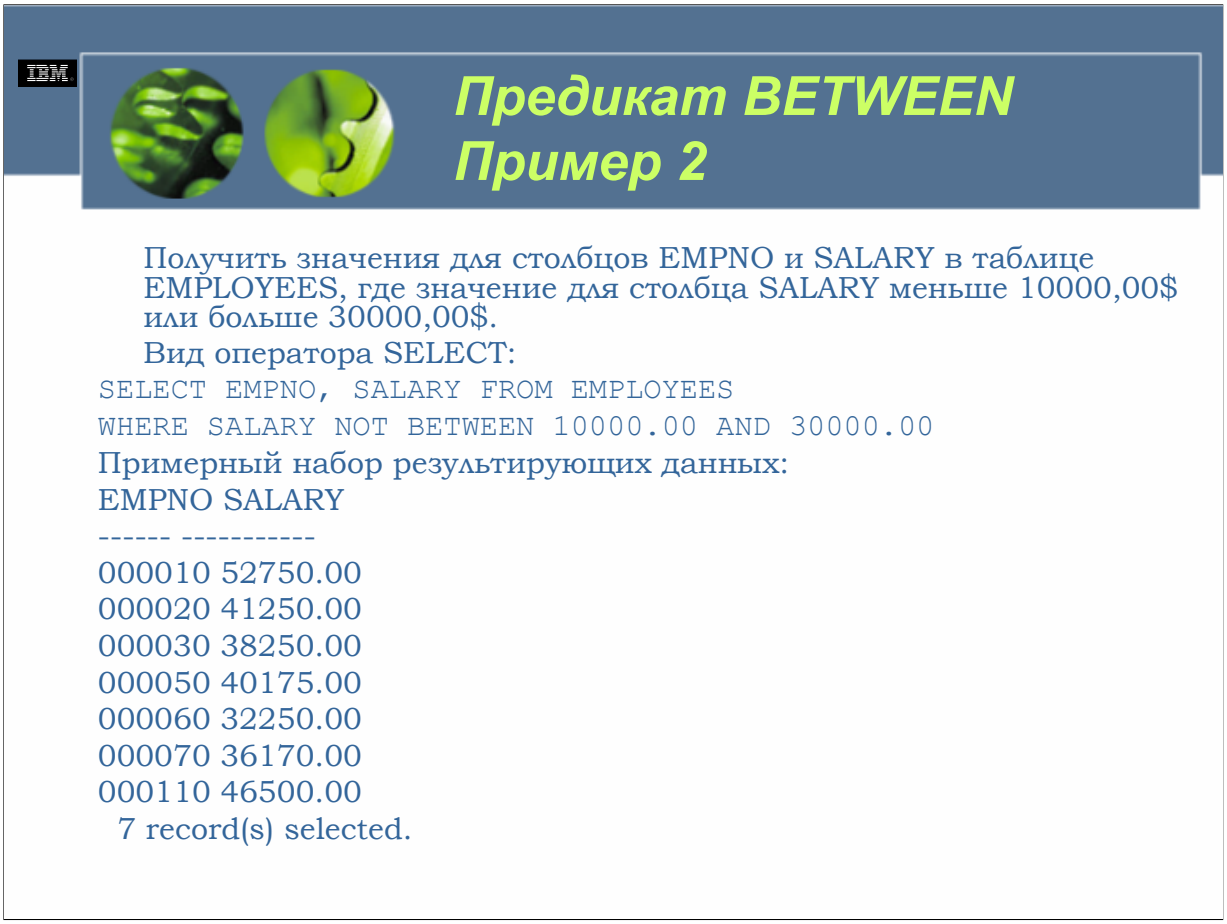
WCF03.0

Типы данных всех элементов, включенных в реляционный предикат, должны быть совместимы, иначе сравнение завершится неудачей. При необходимости для удовлетворения этому условию можно в сочетании с реляционным предикатом использовать скалярные функции (для осуществления необходимых преобразований).



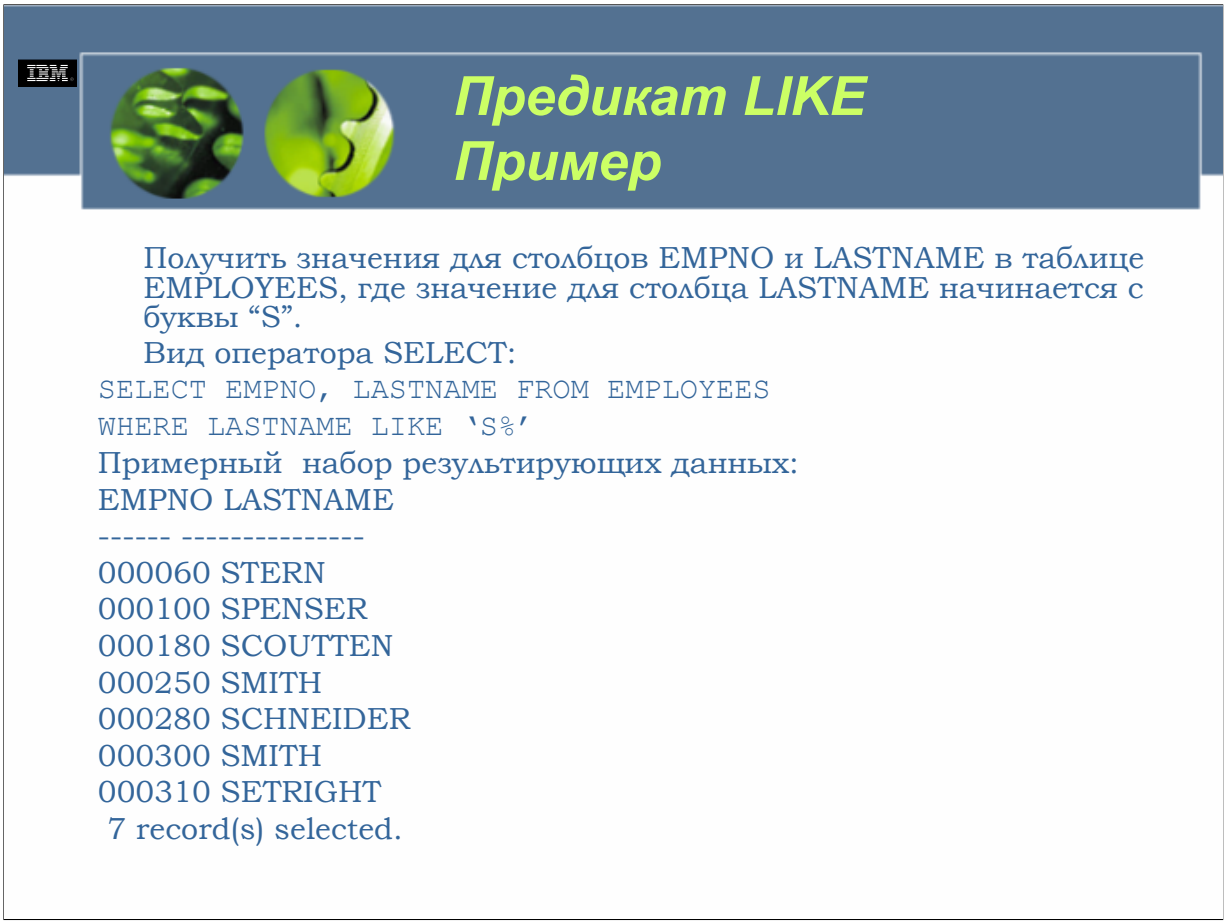
WCF03.0

Предикат BETWEEN используется для определения отношения сравнения, в котором содержание столбца проверяется на предмет того, попадает ли оно в диапазон значений. Как и в случае с реляционными предикатами, предикат BETWEEN используется для включения или исключения определенных строк из набора результирующих данных, созданных в ответ на запрос.



WCF03.0

Если в сочетании с предикатом BETWEEN (или с любым другим предикатом) используется оператор NOT (отрицания), значение предиката меняется на противоположное. В случае предиката BETWEEN проверяется содержимое столбца, и в обработанном наборе результирующих данных возвращаются лишь те значения, которые не попадают в указанный диапазон значений.



WCF03.0

Предикат LIKE используется для определения отношения сравнения, в котором символьное значение проверяется на предмет того, не содержит ли оно определенный шаблон символов. Указанный шаблон символов может состоять из обычных алфавитно-цифровых символов и/или специальных метасимволов, которые распознает DB2 UDB и которые интерпретируются следующим образом:

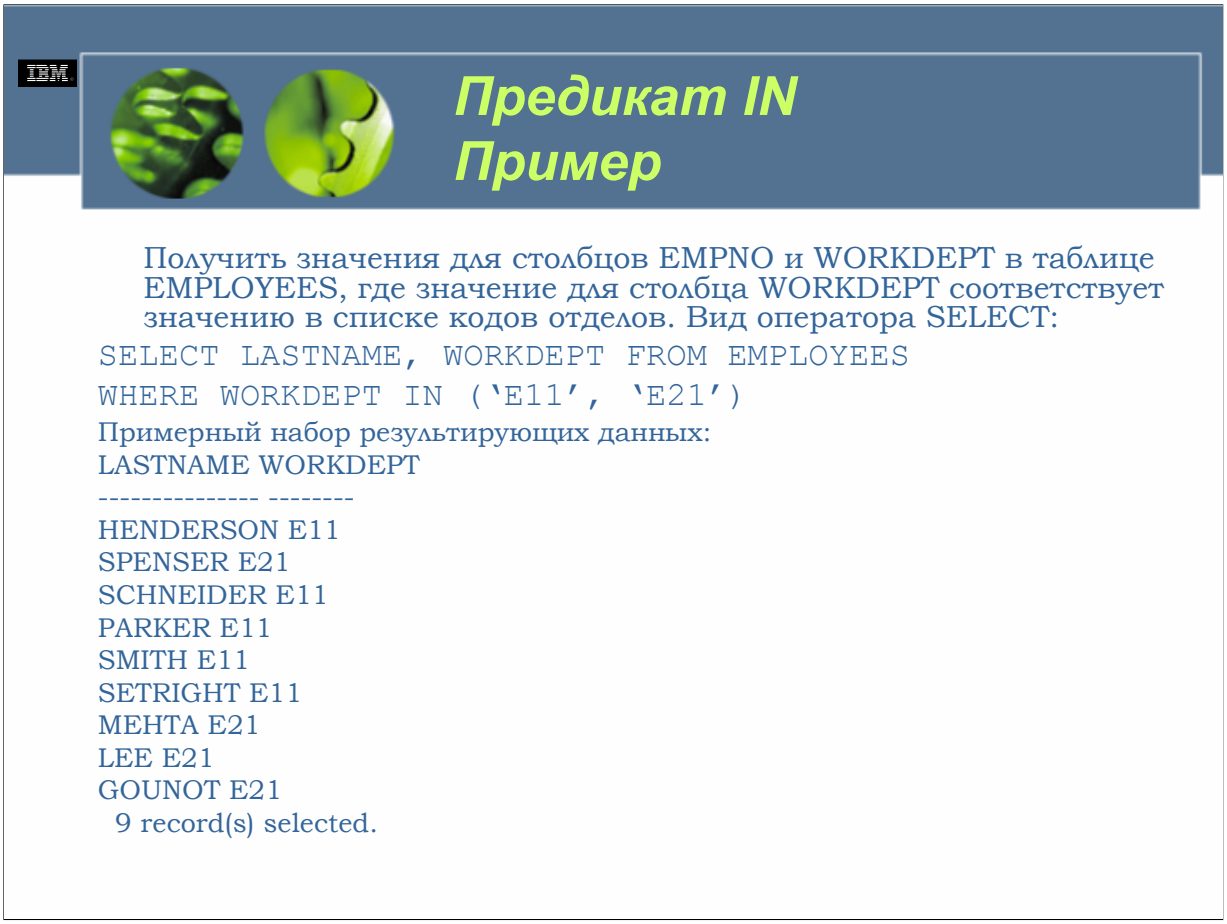
Символ подчеркивания (_) рассматривается как подстановочный символ, который заменяет любой отдельный алфавитно-цифровой символ.

Символ процента (%) рассматривается как подстановочный символ, который заменяет любую последовательность алфавитно-цифровых символов.

При использовании подстановочных символов нужно позаботиться о том, чтобы они были помещены в соответствующем месте указанной строки шаблона . Обратите внимание, что в данном примере возвращаются лишь записи для сотрудников, фамилии которых начинаются с буквы “S”. Если бы указанная строка шаблона символов была бы “%S%”, были бы возвращены записи для сотрудников, фамилии которых содержат букву “S” (в любом месте фамилии).

Необходимо соблюдать осторожность с использованием заглавных и строчных букв в строках шаблона; если проверяемые данные хранятся с учетом регистра, регистр символов, используемых в строке шаблона, должен соответствовать регистру символов сохраненных данных в столбце, для которого осуществляется поиск, иначе соответствующие строки не будут найдены.

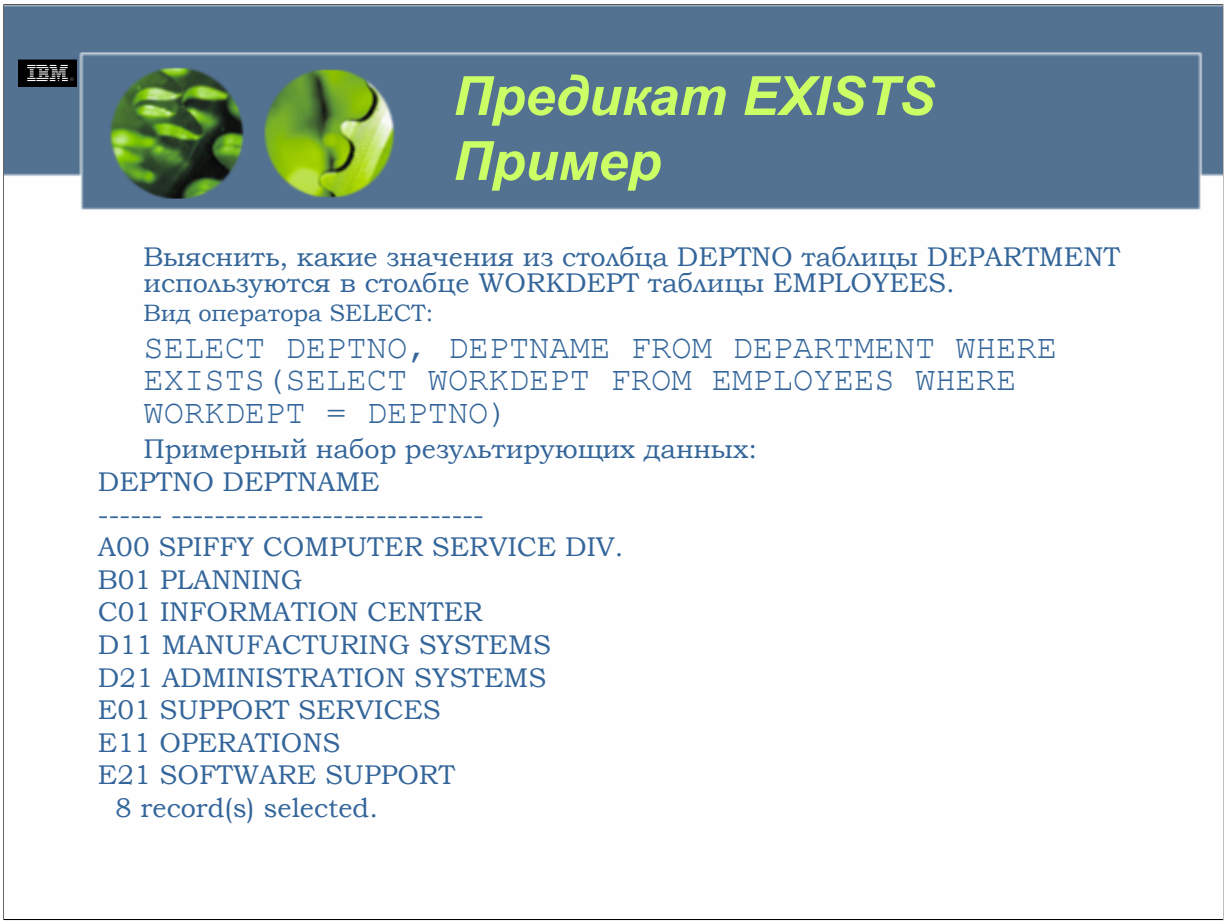
Хотя предикат LIKE предоставляет сравнительно простой способ поиска значений данных, его следует использовать осторожно; издержки, связанные с обработкой предиката LIKE, очень велики и могут поглощать чрезвычайно много ресурсов.



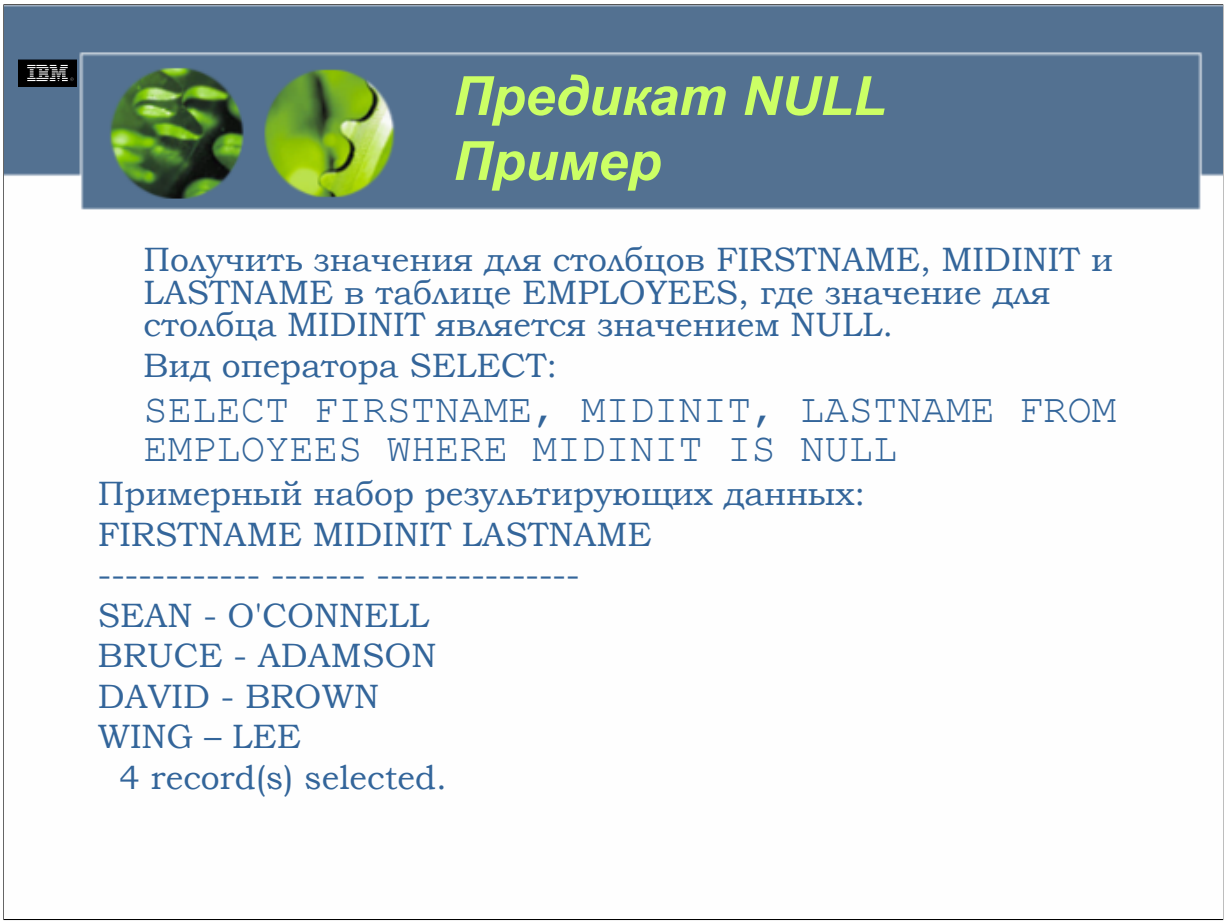
WCF03.0

Предикат IN используется для определения отношения сравнения, в котором значение проверяется на предмет того, совпадает ли оно с одним из конечного набора значений. Этот конечный набор значений может состоять из одного или нескольких литеральных значений, которые закодированы непосредственно в операторе SELECT, или он может быть составлен из непустых значений набора результирующих значений, созданных вторым оператором SELECT (известном по-другому как *подзапрос*).

Подзапросы обычно появляются в условии поиска условия WHERE или условия HAVING (подзапросы могут также использоваться с операциями вставки, изменения или удаления). Подзапрос может включать свои собственные условия поиска, которые, в свою очередь, могут включать свои собственные подзапросы. При обработке таких «вложенных» подзапросов менеджер баз данных DB2 сначала выполняет самый внутренний запрос и использует результаты для выполнения следующего внешнего запроса, и т.д. пока не будут обработаны все вложенные запросы.

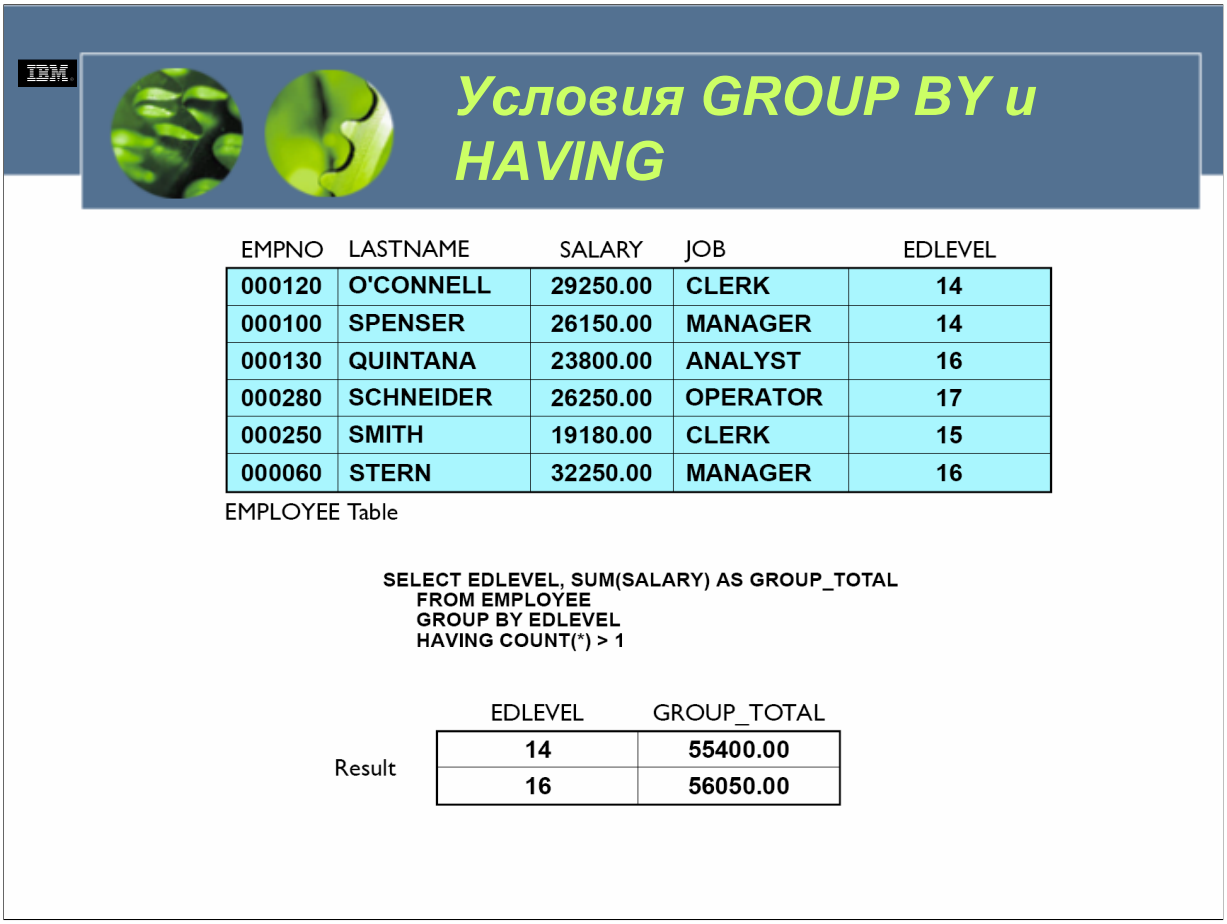


Предикат EXISTS используется для определения того, существует в данной таблице определенное значение или нет. За предикатом EXISTS всегда следует подзапрос, и он возвращает либо TRUE, либо FALSE для указания того, найдено ли определенное значение в наборе результирующих данных, созданных подзапросом. В большинстве ситуаций предикаты EXISTS объединяются с помощью AND с другими предикатами для определения конечного выбора строк.



WCF03.0

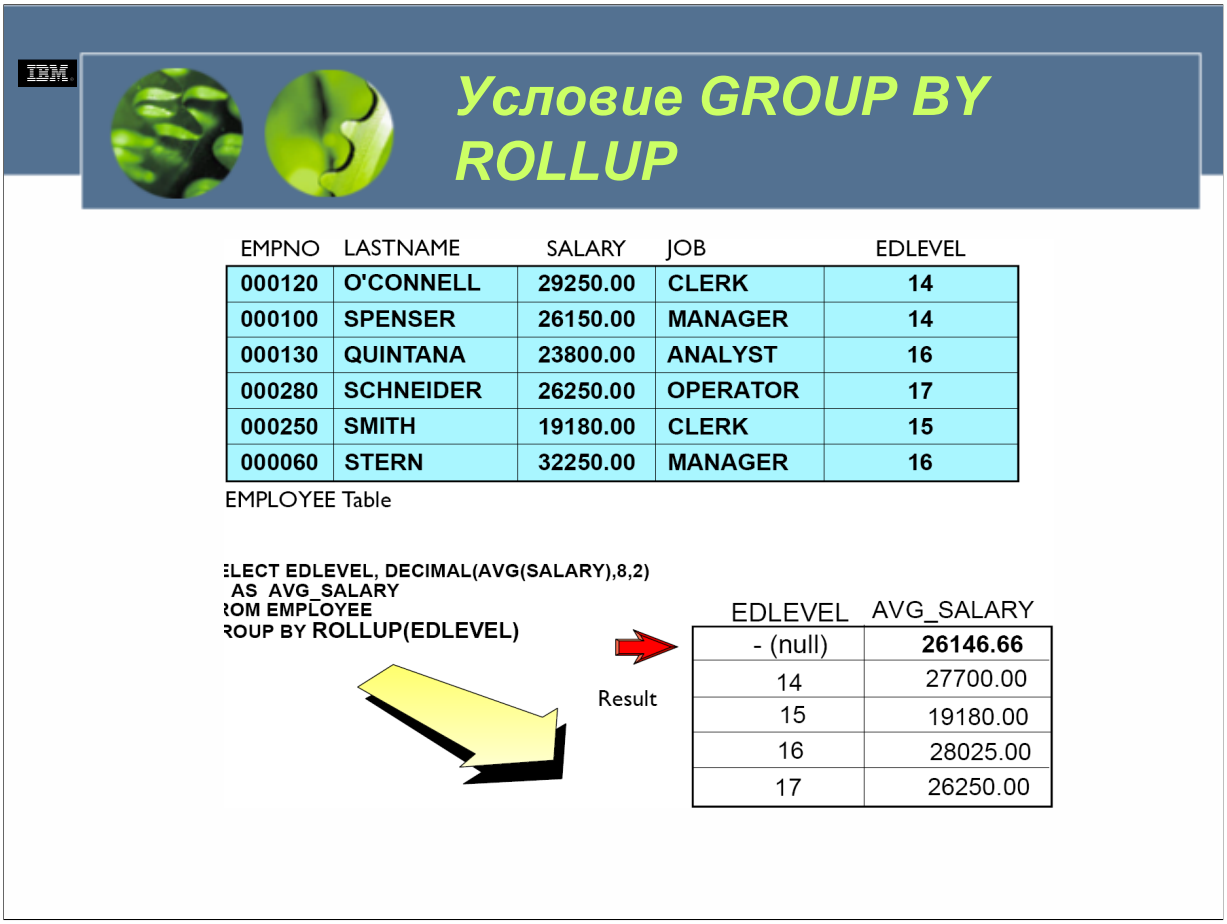
Предикат NULL используется для определения того, является ли определенное значение значением NULL или нет. При использовании предиката NULL важно иметь в виду, что NULL, ноль (0) и пропуск («») не являются одним и тем же значением. NULL является особым маркером, который используется для представления отсутствующей информации, тогда как ноль и пропуск (пустая строка) являются действительными значениями, которые могут храниться в столбце для указания определенного значения (или его отсутствия). Более того, некоторые столбцы принимают значения NULL, тогда как другие не принимают, в зависимости от своего определения. Поэтому перед написанием операторов SQL, которые проверяют наличие значений NULL, убедитесь, что значение NULL поддерживается указываемыми столбцами.



WCF03.0

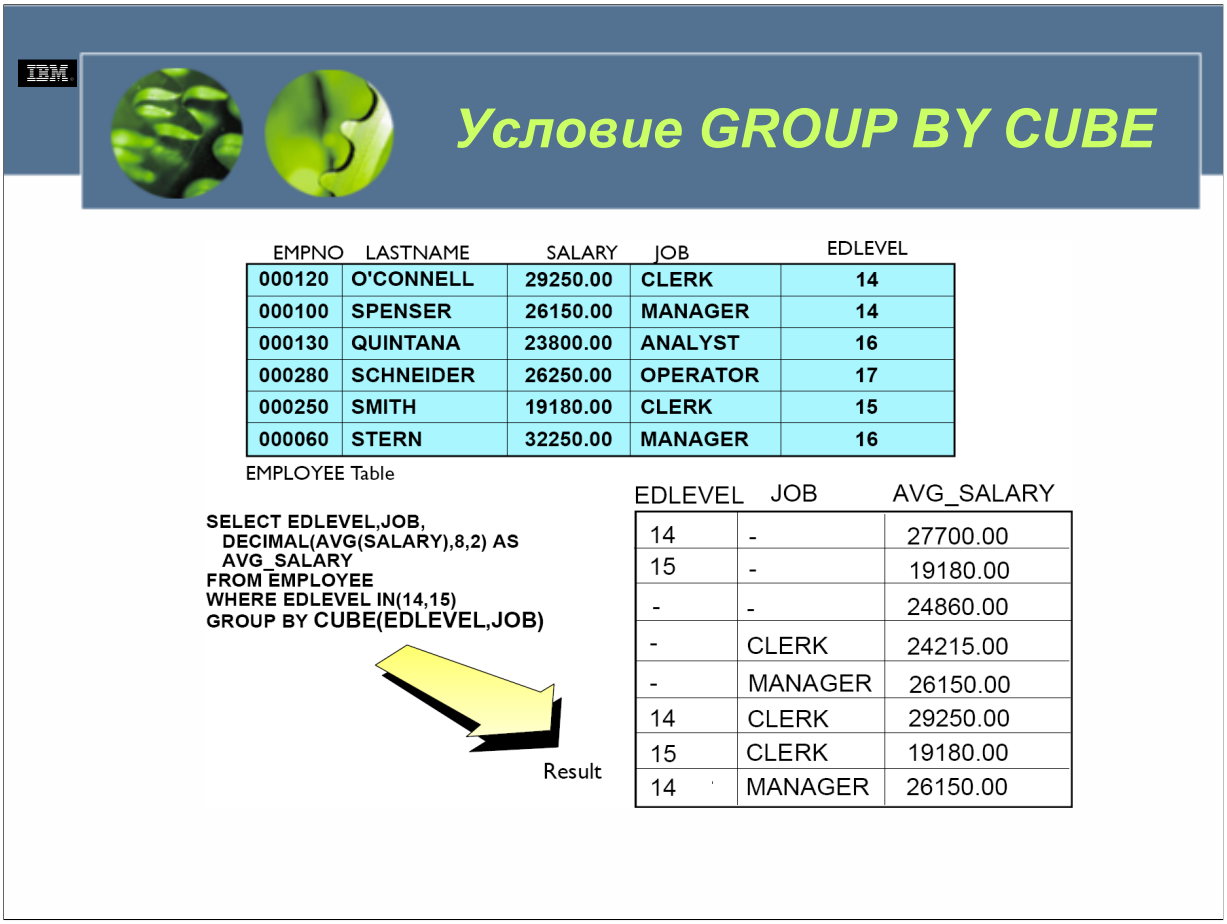
Условие GROUP BY используется для сообщения менеджеру баз данных DB2 о том, как организовать строки данных, возвращаемых в наборе результирующих данных, созданных в ответ на запрос. В своей простейшей форме за условием GROUP BY следует выражение группировки, обычно представляющее собой названия одного или нескольких столбцов (соответствующих названиям столбцов в наборе результирующих данных, который должен быть упорядочен условием GROUP BY). Условие GROUP BY используется также для указания того, какие столбцы должны группироваться совместно для предоставления ввода функциям сводки, таким как SUM() и AVG().

Условие HAVING используется для применения дальнейших критериев выбора к столбцам, на которые осуществлена ссылка в условии GROUP BY. Это условие ведет себя как условие WHERE за тем исключением, что оно ссылается на данные, которые уже были сгруппированы условием GROUP BY (условие HAVING используется, чтобы сообщить менеджеру баз данных DB2, как выбрать строки, которые должны быть возвращены в виде набора результирующих данных, которые уже были сгруппированы). И аналогично условию WHERE, за условием HAVING следует условие поиска, которое действует в качестве простого теста, при применении которого к строке данных будет получено значение TRUE, FALSE или Unknown. Если результат проверки равен TRUE, строка должна быть возвращена в создаваемом наборе результирующих данных; если результат проверки равен FALSE или Unknown, строка пропускается. Кроме того, условие поиска конструкции HAVING может состоять из тех же самых предикатов, которые распознаются условием WHERE.



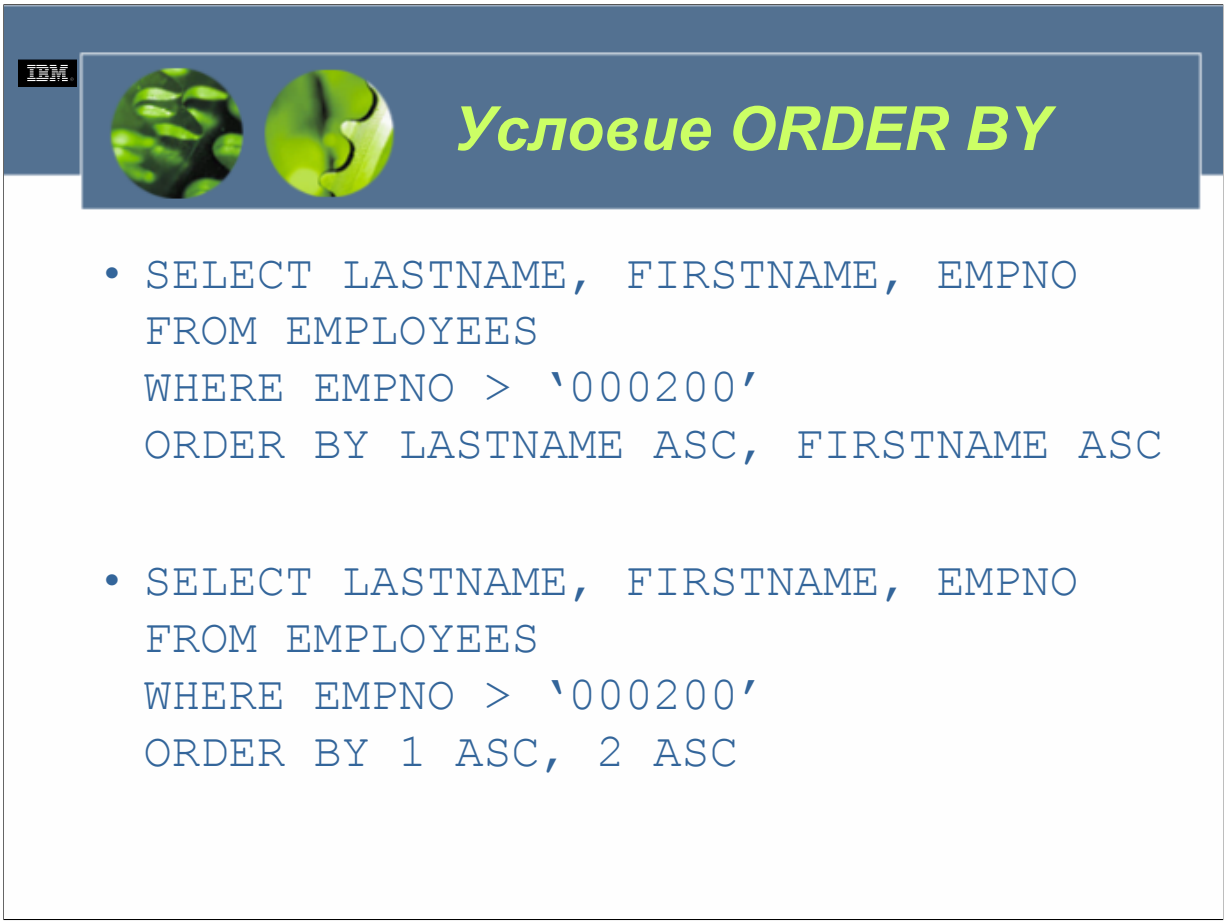
WCF03.0

Условие GROUP BY ROLLUP используется для анализа набора данных в одном (иерархическом) измерении, но с более чем одним уровнем детализации. Например, вы могли бы сгруппировать данные по последовательно возрастающим организационным блокам, таким как команда, отдел и подразделение, или по последовательно возрастающим географическим блокам, таким как город, округ, штат или область, страна и континент. В данном примере в условии GROUP BY ROLLUP указано лишь одно выражение (называемое *группирующим выражением (grouping expression)*; в данном случае группирующим выражением является EDLEVEL). Однако в одном условии GROUP BY ROLLUP может быть определено одно или несколько группирующих выражений (например, GROUP BY ROLLUP (EDLEVEL,JOB)). Когда указано множество группирующих выражений, менеджер баз данных DB2 группирует данные по всем используемым группирующим выражениям, затем по всем используемым выражениям, кроме последнего, и т.д. Затем он делает одну заключительную группировку, которая состоит из всего содержания указанной таблицы. Кроме того, при указании нескольких группирующих выражений важно убедиться, что они перечислены в соответствующем порядке – если один вид группы логически содержится внутри другой (например, отделы внутри подразделения), тогда эта группа должна быть перечислена после группы, в которой она содержится (т.е. GROUP BY ROLLUP (DIVISION,DEPARTMENTS)), но никогда не до нее.



WCF03.0

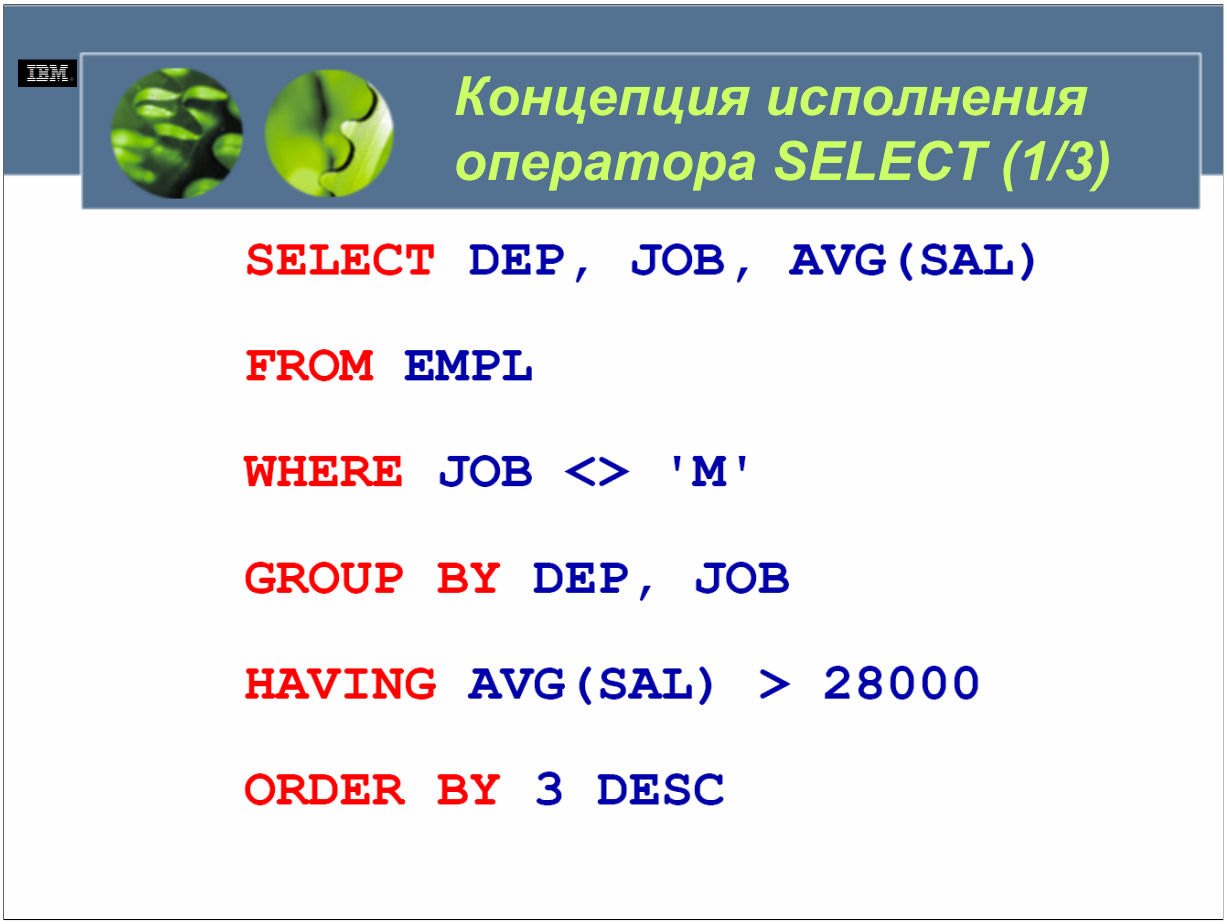
Условие GROUP BY CUBE используется для анализа набора данных путем организации его в группы в нескольких измерениях.



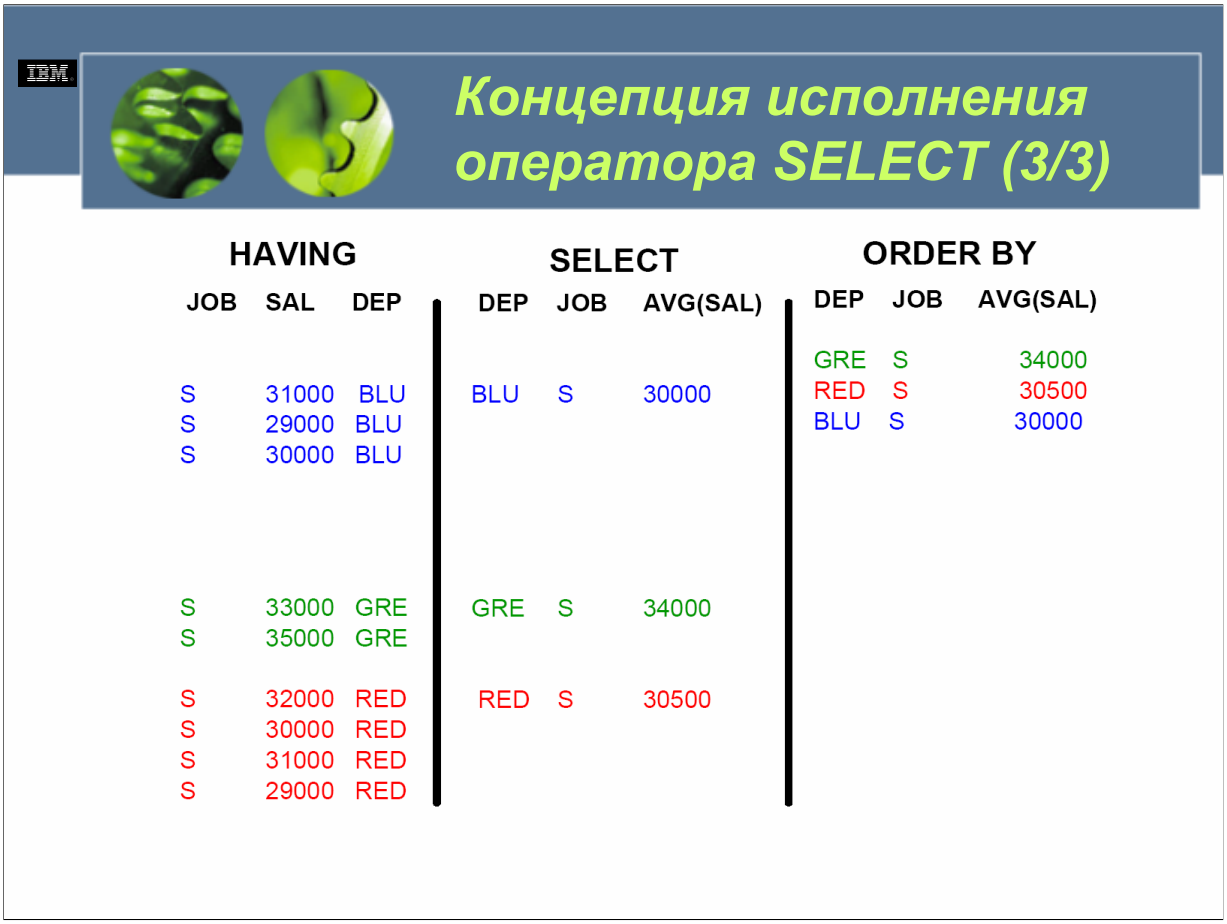
WCF03.0

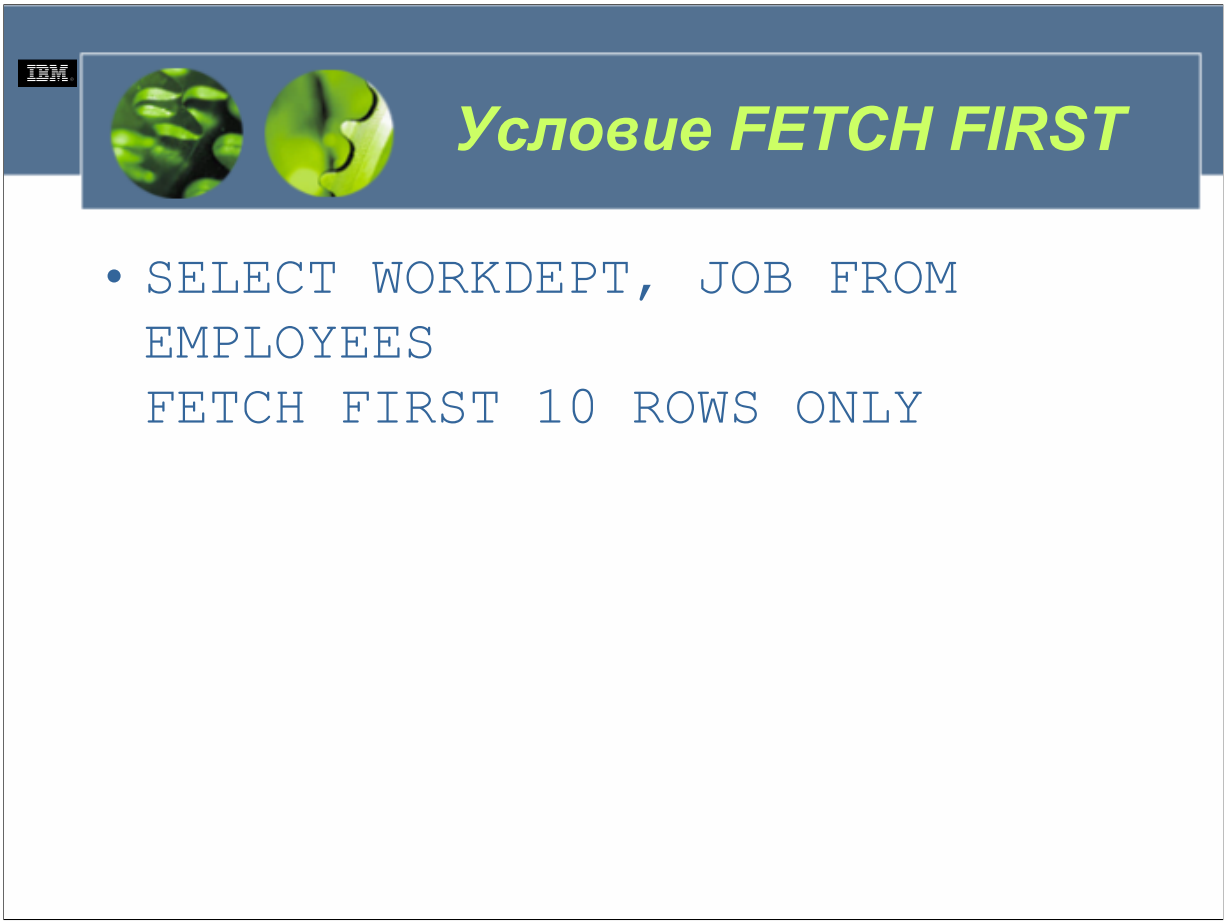
Условие ORDER BY используется для того, чтобы сообщить менеджеру баз данных DB2, как сортировать и упорядочивать строки, которые должны быть возвращены в созданном в ответ на запрос наборе результирующих данных. Когда условие ORDER BY указано, за ним следуют имена столбцов, данные которых должны быть отсортированы. Для сортировки можно использовать несколько столбцов, а каждый использованный столбец может быть отсортирован либо в восходящем, либо в нисходящем порядке. Если за именем столбца следует ключевое слово ASC, используется восходящий порядок, а если за именем столбца следует ключевое слово DESC, используется нисходящий порядок. Более того, когда условием ORDER BY идентифицируется более одного столбца, соответствующий набор результирующих данных сортируется по первому указанному столбцу (первичная сортировка), затем отсортированные данные сортируются снова по следующему указанному столбцу и т.д. до тех пор, пока данные не будут отсортированы по каждому из указанных столбцов.

Использование условия ORDER BY просто, если набор результирующих данных полностью составлен из именованных столбцов. Но что случится, если созданный набор результирующих данных нужно упорядочить по столбцу сводки или столбцу результата, который нельзя указать по имени? Поскольку такие ситуации могут существовать, на месте имени столбца с условием ORDER BY может использоваться целое значение, соответствующее номеру определенного столбца. Когда используются целые значения, первый столбец слева в созданном наборе результирующих данных считается столбцом 1, следующий столбец 2 и т.д.



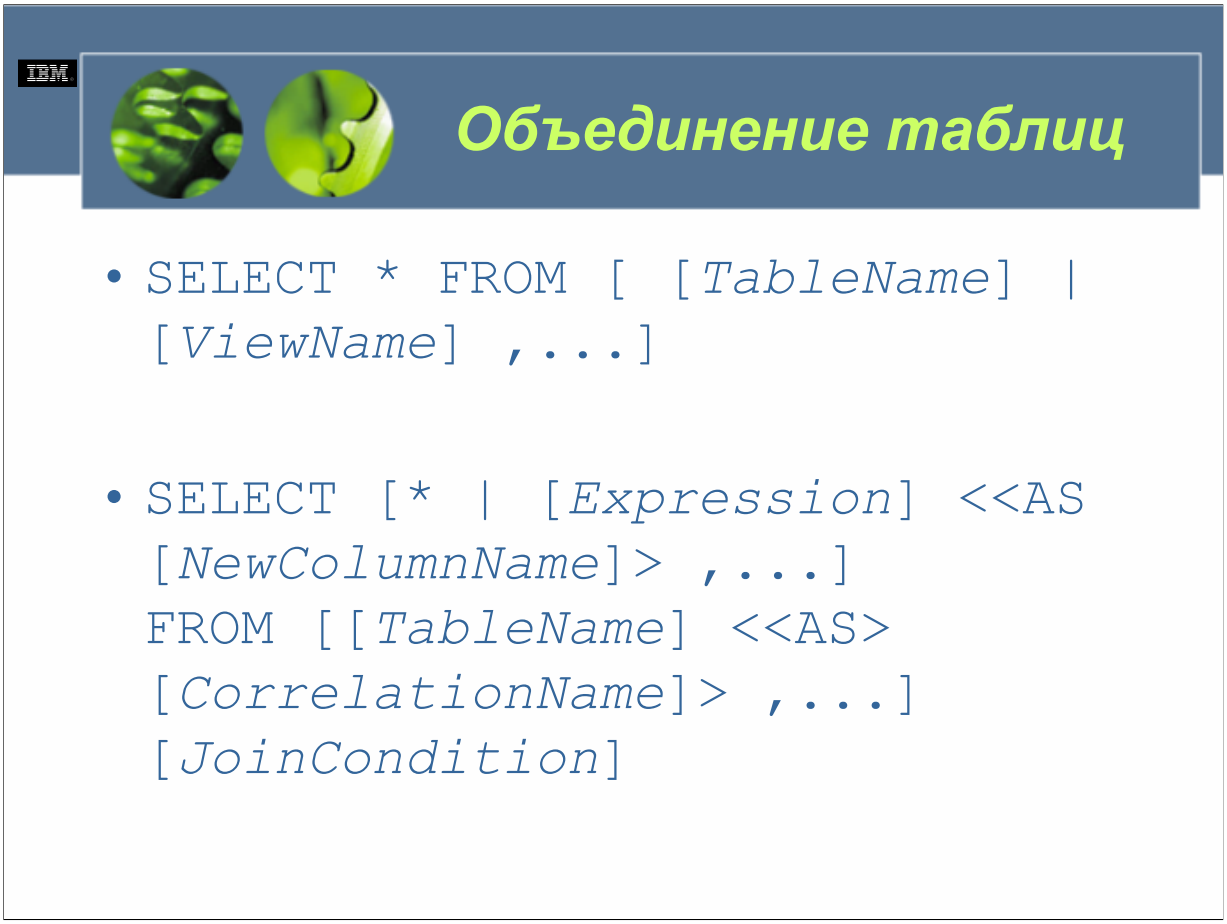
WCF03.0





WCF03.0

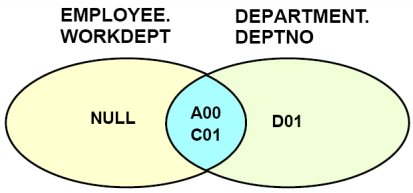
Условие FETCH FIRST используется для ограничения числа строк, возвращаемых в созданном в ответ на запрос наборе результирующих данных. Когда используется условие FETCH FIRST, за ним следует положительное целое значение и слова ROWS ONLY. Это сообщает менеджеру баз данных DB2 о том, что пользователь/приложение, выполняющие запрос, не хотят получать более *n* строк, независимо от того, сколько строк могло бы быть в наборе результирующих данных, которые были бы созданы, если условие FETCH FIRST не было бы указано.



WCF03.0

Expression определяет один или несколько столбцов, значения которых должны быть возвращены при выполнении оператора SELECT. Указанное для этой опции значение может быть любым действительным элементом языка SQL; однако обычно используется имена столбцов соответствующих таблиц или производных таблиц. *NewColumnName* определяет новое имя столбца, которое должно использоваться в наборе результирующих данных, возвращенных оператором SELECT, вместо указанного имени столбца соответствующей таблицы или производной таблицы. *TableName* определяет имена одной или нескольких таблиц, из которых должны быть получены данные. *CorrelationName* определяет сокращенное имя, которое может использоваться при ссылке на имя таблицы, указанное в параметре *TableName*. *JoinCondition* определяет условие, которое должно использоваться для объединения каждой из указанных таблиц. Обычно это условие WHERE, в котором значения столбца в одной таблице сравниваются со значениями аналогичного столбца в другой таблице.

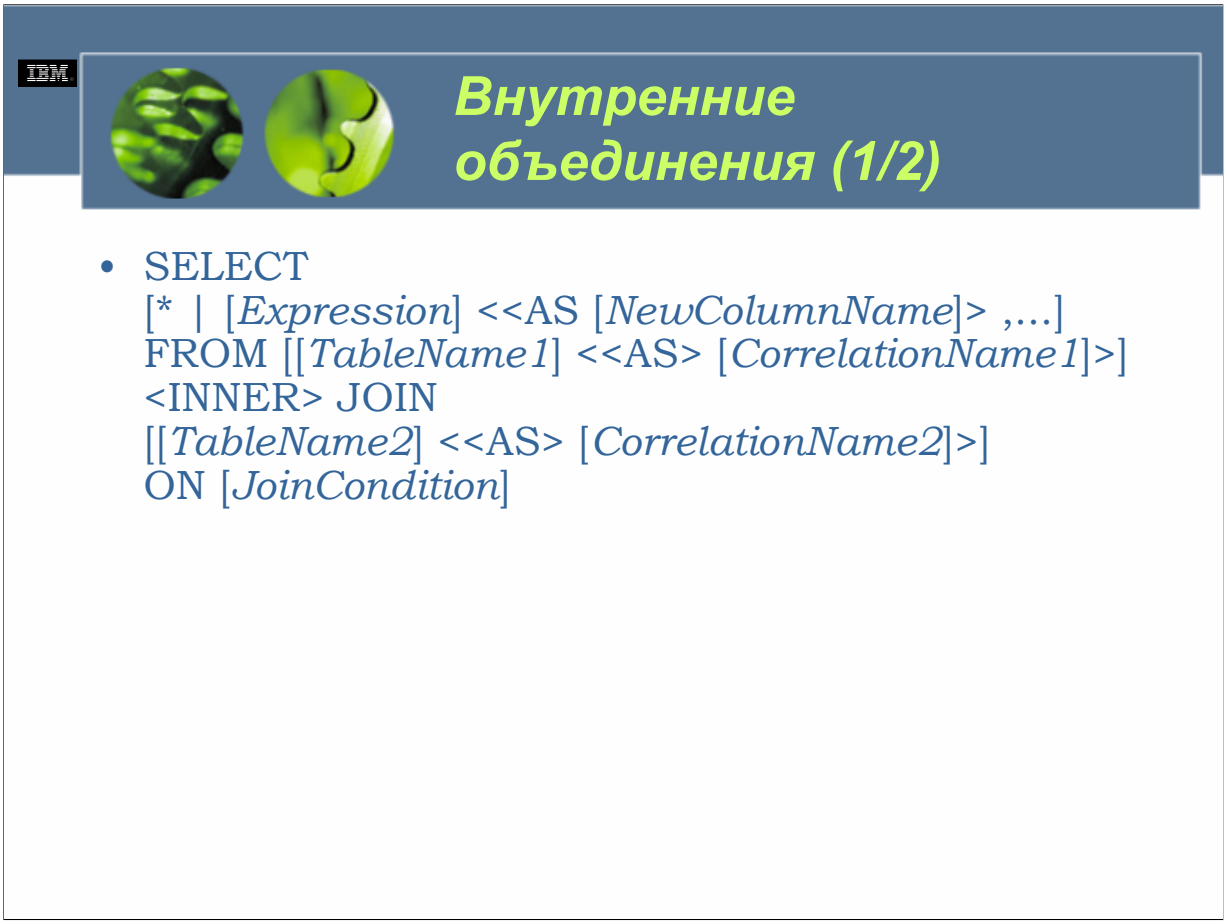
Объединение



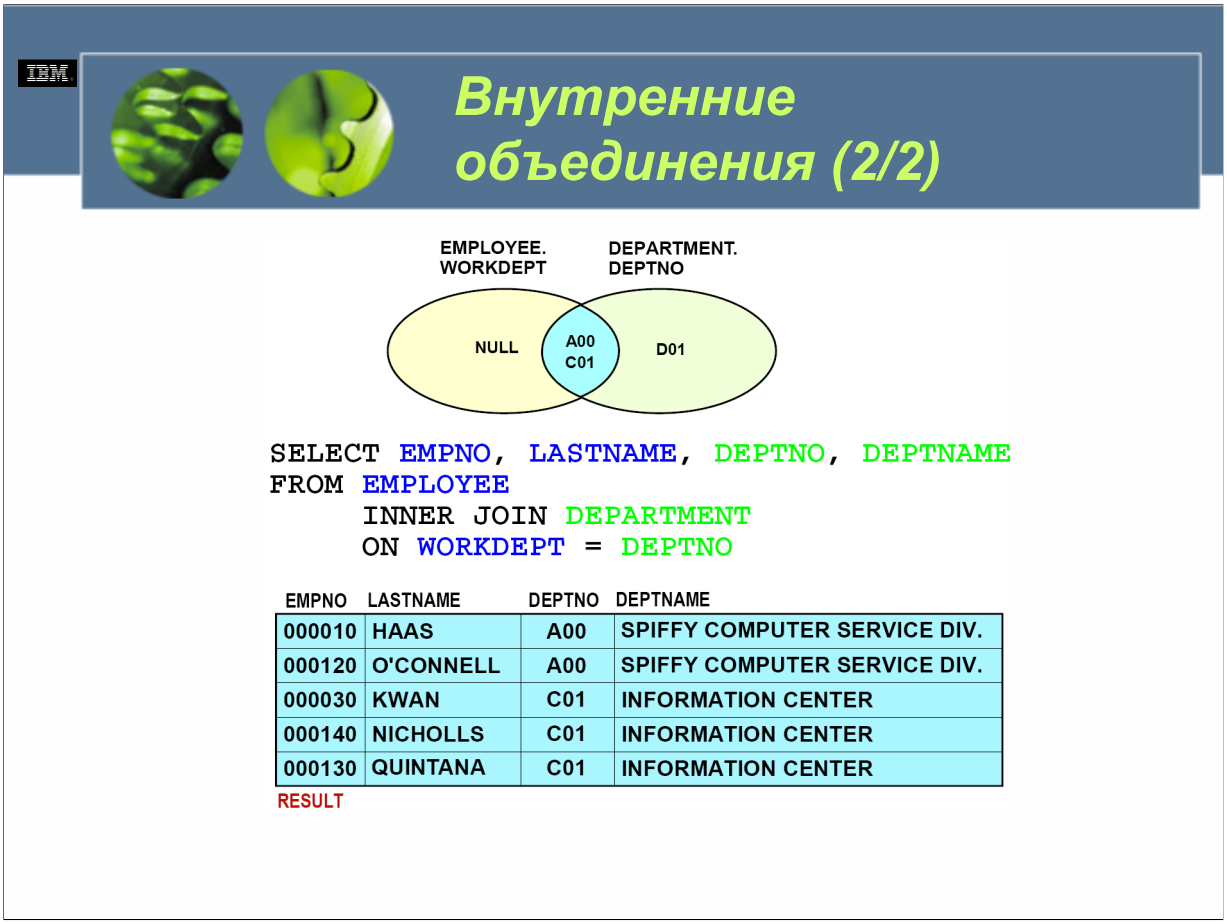

```
SELECT EMPNO, LASTNAME, DEPTNO, DEPTNAME
       FROM EMPLOYEE, DEPARTMENT
       WHERE WORKDEPT = DEPTNO
```

EMPNO	LASTNAME	DEPTNO	DEPTNAME
000010	HAAS	A00	SPIFFY COMPUTER SERVICE DIV.
000120	O'CONNELL	A00	SPIFFY COMPUTER SERVICE DIV.
000030	KWAN	C01	INFORMATION CENTER
000140	NICHOLLS	C01	INFORMATION CENTER
000130	QUINTANA	C01	INFORMATION CENTER

RESULT

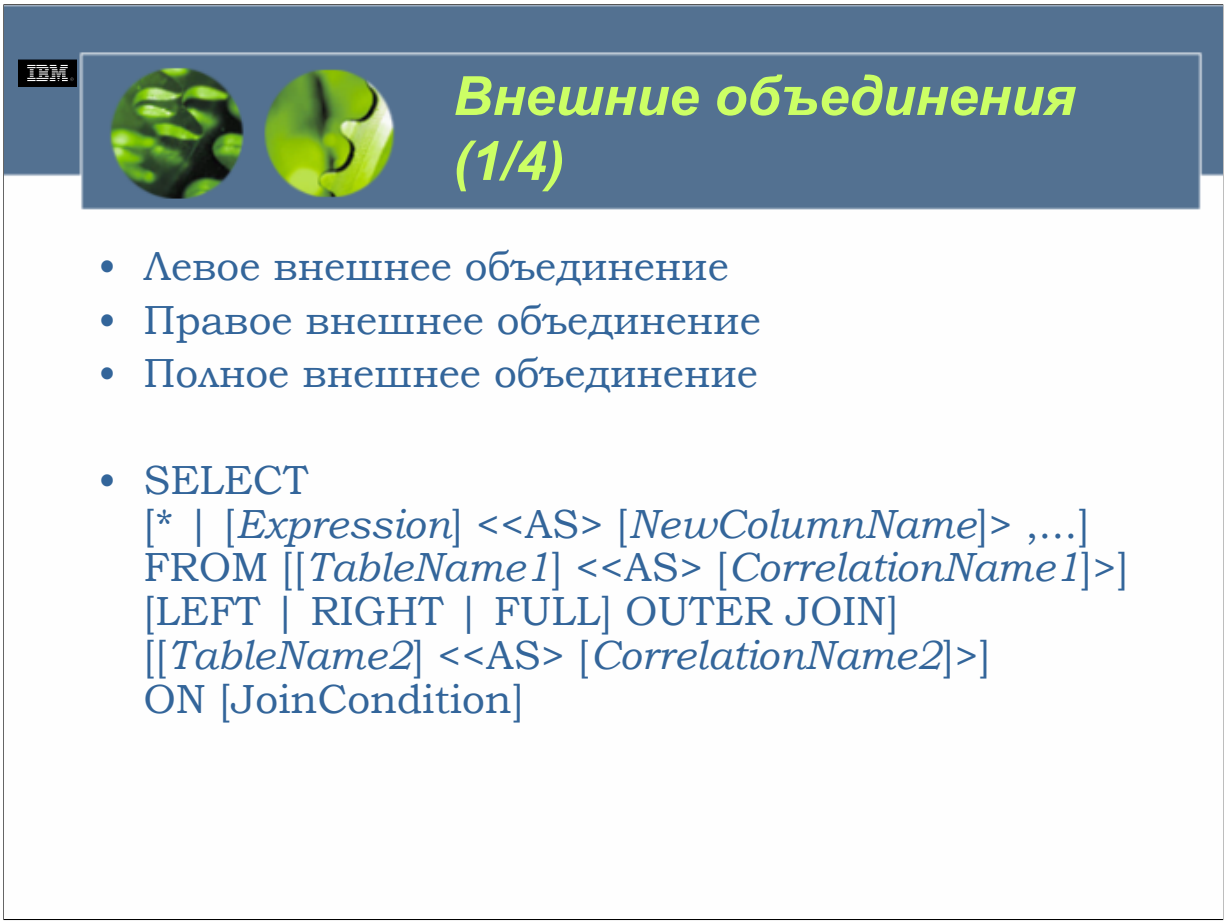


Expression определяет один или несколько столбцов, значения которых должны быть возвращены при выполнении оператора SELECT. Указанное для этой опции значение может быть любым действительным элементом языка SQL; однако обычно используются имена столбцов соответствующих таблиц или производных таблиц. *NewColumnName* определяет новое имя столбца, которое должно использоваться в возвращаемом оператором SELECT наборе результирующих данных вместо указанного имени столбца соответствующей таблицы или производной таблицы. *TableName1* определяет имя первой таблицы, из которой должны быть получены данные. *CorrelationName1* определяет сокращенное имя, которое может использоваться при ссылке на левую таблицу операции объединения. *TableName2* определяет имя второй таблицы, из которой должны быть получены данные. *CorrelationName2* определяет сокращенное имя, которое может использоваться при ссылке на правую таблицу операции объединения. *JoinCondition* определяет условие, которое должно использоваться для объединения двух указанных таблиц.



WCF03.0

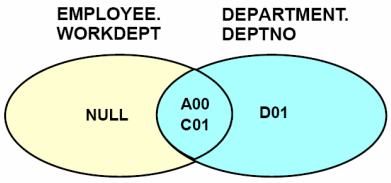
Expression определяет один или несколько столбцов, значения которых должны быть возвращены при выполнении оператора SELECT. Указанное для этой опции значение может быть любым действительным элементом языка SQL; однако обычно используются имена столбцов соответствующих таблиц или производных таблиц. *NewColumnName* определяет новое имя столбца, которое должно использоваться в возвращаемом оператором SELECT наборе результирующих данных вместо указанного имени столбца соответствующей таблицы или производной таблицы. *TableName1* определяет имя первой таблицы, из которой должны быть получены данные. *CorrelationName1* определяет сокращенное имя, которое может использоваться при ссылке на левую таблицу операции объединения. *TableName2* определяет имя второй таблицы, из которой должны быть получены данные. *CorrelationName2* определяет сокращенное имя, которое может использоваться при ссылке на правую таблицу операции объединения. *JoinCondition* определяет условие, которое должно использоваться для объединения двух указанных таблиц.



WCF03.0

Expression определяет один или несколько столбцов, значения которых должны быть возвращены при выполнении оператора SELECT. Указанное для этой опции значение может быть любым действительным элементом языка SQL; однако обычно используются имена столбцов соответствующих таблиц или производных таблиц. *NewColumnName* определяет новое имя столбца, которое должно использоваться в возвращенном оператором SELECT наборе результирующих данных вместо указанного имени столбца в соответствующей таблице или производной таблице. *TableName1* определяет имя первой таблицы, из которой должны быть получены данные. Эта таблица считается «левой» таблицей во внешнем объединении. *CorrelationName1* определяет сокращенное имя, которое может использоваться при ссылке на левую таблицу операции объединения. *TableName2* определяет имя второй таблицы, из которой должны быть получены данные. Эта таблица считается «правой» таблицей во внешнем объединении. *CorrelationName2* определяет сокращенное имя, которое может использоваться при ссылке на правую таблицу операции объединения. *JoinCondition* определяет условие, которое должно использоваться для объединения двух указанных таблиц.

Внешние объединения (2/4)




```
SELECT EMPNO, LASTNAME, DEPTNO, DEPTNAME
FROM EMPLOYEE
     RIGHT OUTER JOIN DEPARTMENT
     ON WORKDEPT = DEPTNO
```

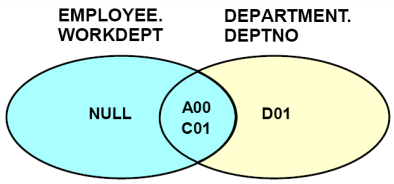
EMPNO	LASTNAME	DEPTNO	DEPTNAME
000010	HAAS	A00	SPIFFY COMPUTER SERVICE DIV.
000120	O'CONNELL	A00	SPIFFY COMPUTER SERVICE DIV.
000030	KWAN	C01	INFORMATION CENTER
000130	QUINTANA	C01	INFORMATION CENTER
000140	NICHOLLS	C01	INFORMATION CENTER
-	-	D01	DEVELOPMENT CENTER

RESULT

WCF03.0

Expression определяет один или несколько столбцов, значения которых должны быть возвращены при выполнении оператора SELECT. Указанное для этой опции значение может быть любым действительным элементом языка SQL; однако обычно используются имена столбцов соответствующих таблиц или производных таблиц. *NewColumnName* определяет новое имя столбца, которое должно использоваться в возвращенном оператором SELECT наборе результирующих данных вместо указанного имени столбца в соответствующей таблице или производной таблице. *TableName1* определяет имя первой таблицы, из которой должны быть получены данные. Эта таблица считается «левой» таблицей во внешнем объединении. *CorrelationName1* определяет сокращенное имя, которое может использоваться при ссылке на левую таблицу операции объединения. *TableName2* определяет имя второй таблицы, из которой должны быть получены данные. Эта таблица считается «правой» таблицей во внешнем объединении. *CorrelationName2* определяет сокращенное имя, которое может использоваться при ссылке на правую таблицу операции объединения. *JoinCondition* определяет условие, которое должно использоваться для объединения двух указанных таблиц.

Внешние объединения (3/4)



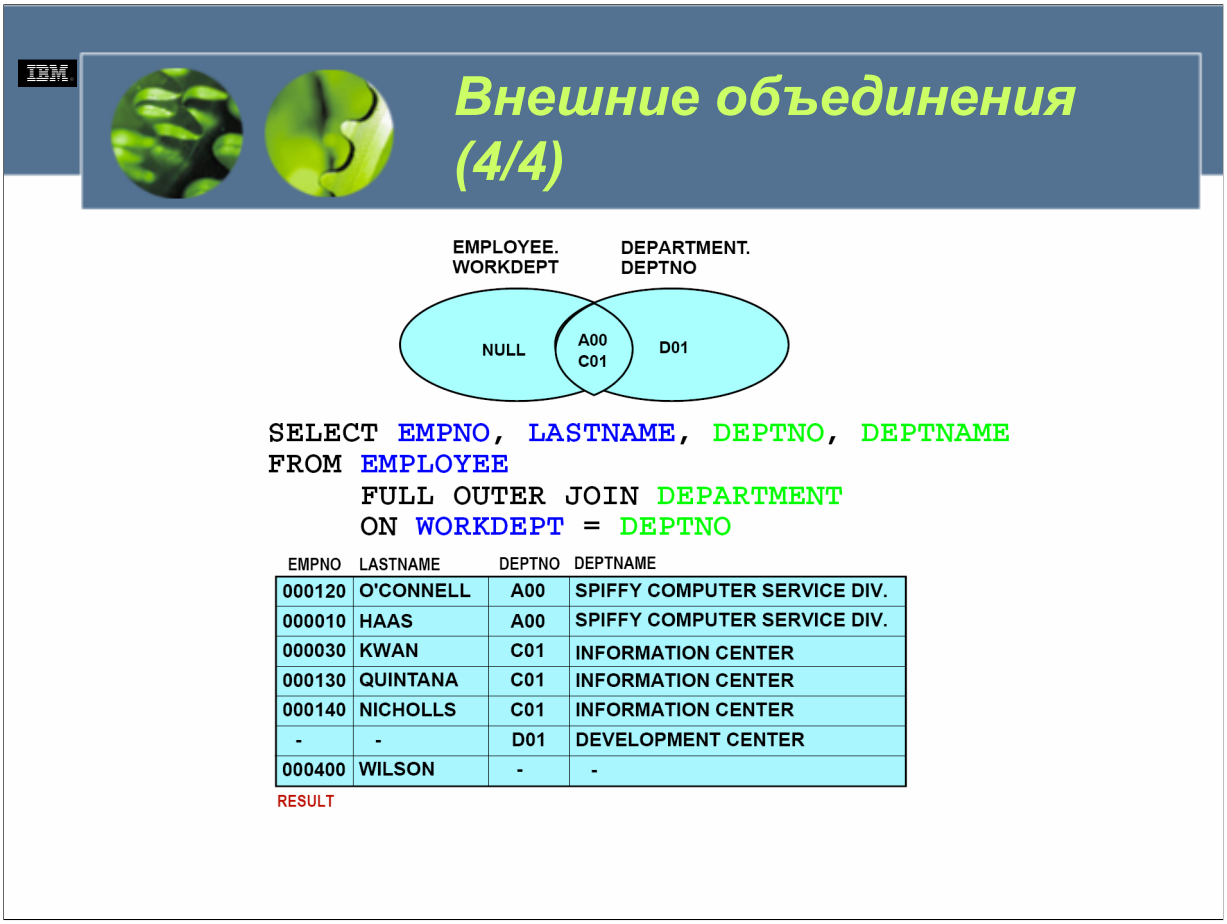

```
SELECT EMPNO, LASTNAME, DEPTNO, DEPTNAME
FROM EMPLOYEE
     LEFT OUTER JOIN DEPARTMENT
     ON WORKDEPT = DEPTNO
```

EMPNO	LASTNAME	DEPTNO	DEPTNAME
000010	HAAS	A00	SPIFFY COMPUTER SERVICE DIV.
000030	KWAN	C01	INFORMATION CENTER
000120	O'CONNELL	A00	SPIFFY COMPUTER SERVICE DIV.
000130	QUINTANA	C01	INFORMATION CENTER
000140	NICHOLLS	C01	INFORMATION CENTER
000400	WILSON	-	-

RESULT

WCF03.0

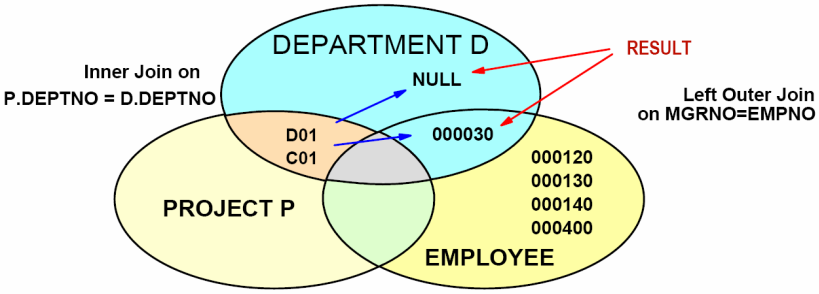
Expression определяет один или несколько столбцов, значения которых должны быть возвращены при выполнении оператора SELECT. Указанное для этой опции значение может быть любым действительным элементом языка SQL; однако обычно используются имена столбцов соответствующих таблиц или производных таблиц. *NewColumnName* определяет новое имя столбца, которое должно использоваться в возвращенном оператором SELECT наборе результирующих данных вместо указанного имени столбца в соответствующей таблице или производной таблице. *TableName1* определяет имя первой таблицы, из которой должны быть получены данные. Эта таблица считается «левой» таблицей во внешнем объединении. *CorrelationName1* определяет сокращенное имя, которое может использоваться при ссылке на левую таблицу операции объединения. *TableName2* определяет имя второй таблицы, из которой должны быть получены данные. Эта таблица считается «правой» таблицей во внешнем объединении. *CorrelationName2* определяет сокращенное имя, которое может использоваться при ссылке на правую таблицу операции объединения. *JoinCondition* определяет условие, которое должно использоваться для объединения двух указанных таблиц.



WCF03.0

Expression определяет один или несколько столбцов, значения которых должны быть возвращены при выполнении оператора SELECT. Указанное для этой опции значение может быть любым действительным элементом языка SQL; однако обычно используются имена столбцов соответствующих таблиц или производных таблиц. *NewColumnName* определяет новое имя столбца, которое должно использоваться в возвращенном оператором SELECT наборе результирующих данных вместо указанного имени столбца в соответствующей таблице или производной таблице. *TableName1* определяет имя первой таблицы, из которой должны быть получены данные. Эта таблица считается «левой» таблицей во внешнем объединении. *CorrelationName1* определяет сокращенное имя, которое может использоваться при ссылке на левую таблицу операции объединения. *TableName2* определяет имя второй таблицы, из которой должны быть получены данные. Эта таблица считается «правой» таблицей во внешнем объединении. *CorrelationName2* определяет сокращенное имя, которое может использоваться при ссылке на правую таблицу операции объединения. *JoinCondition* определяет условие, которое должно использоваться для объединения двух указанных таблиц.

Объединение нескольких таблиц

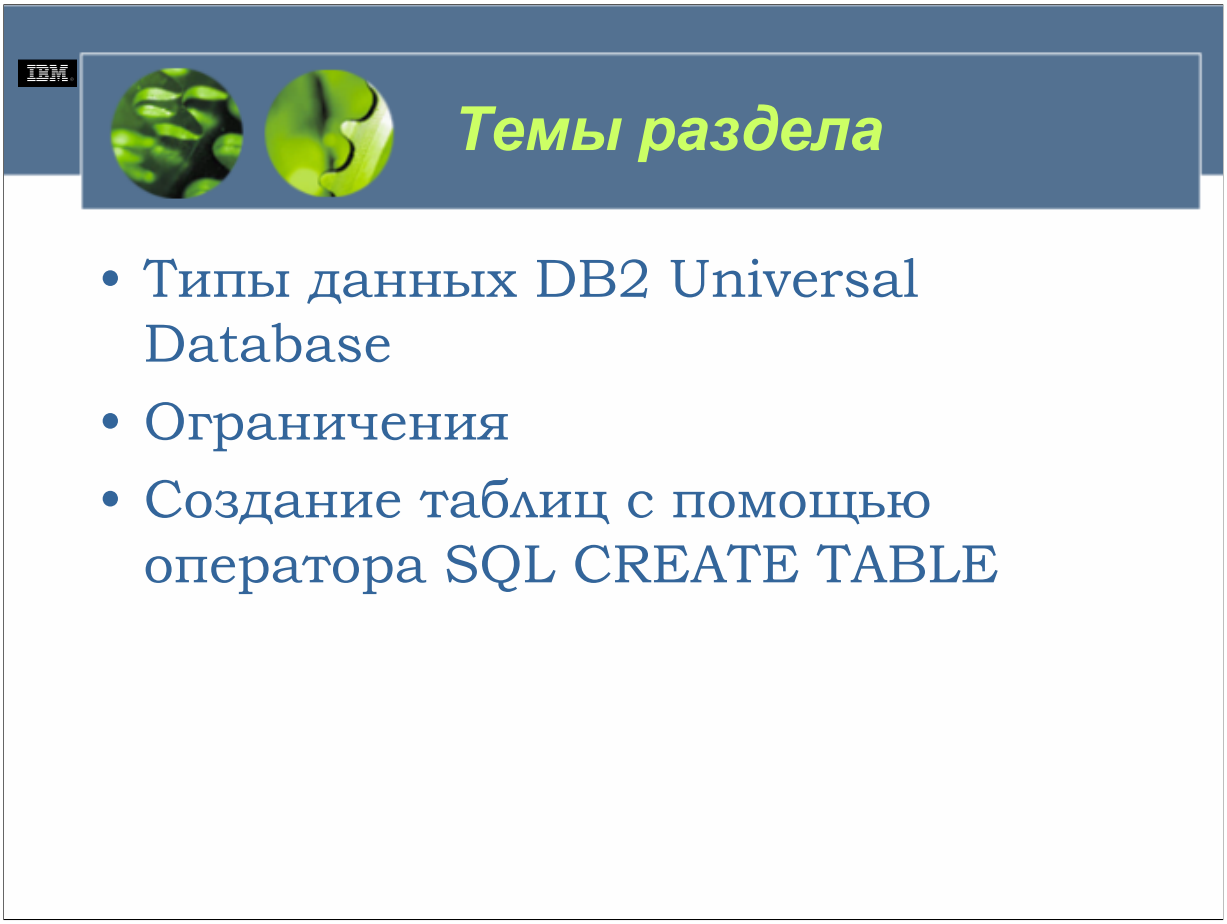


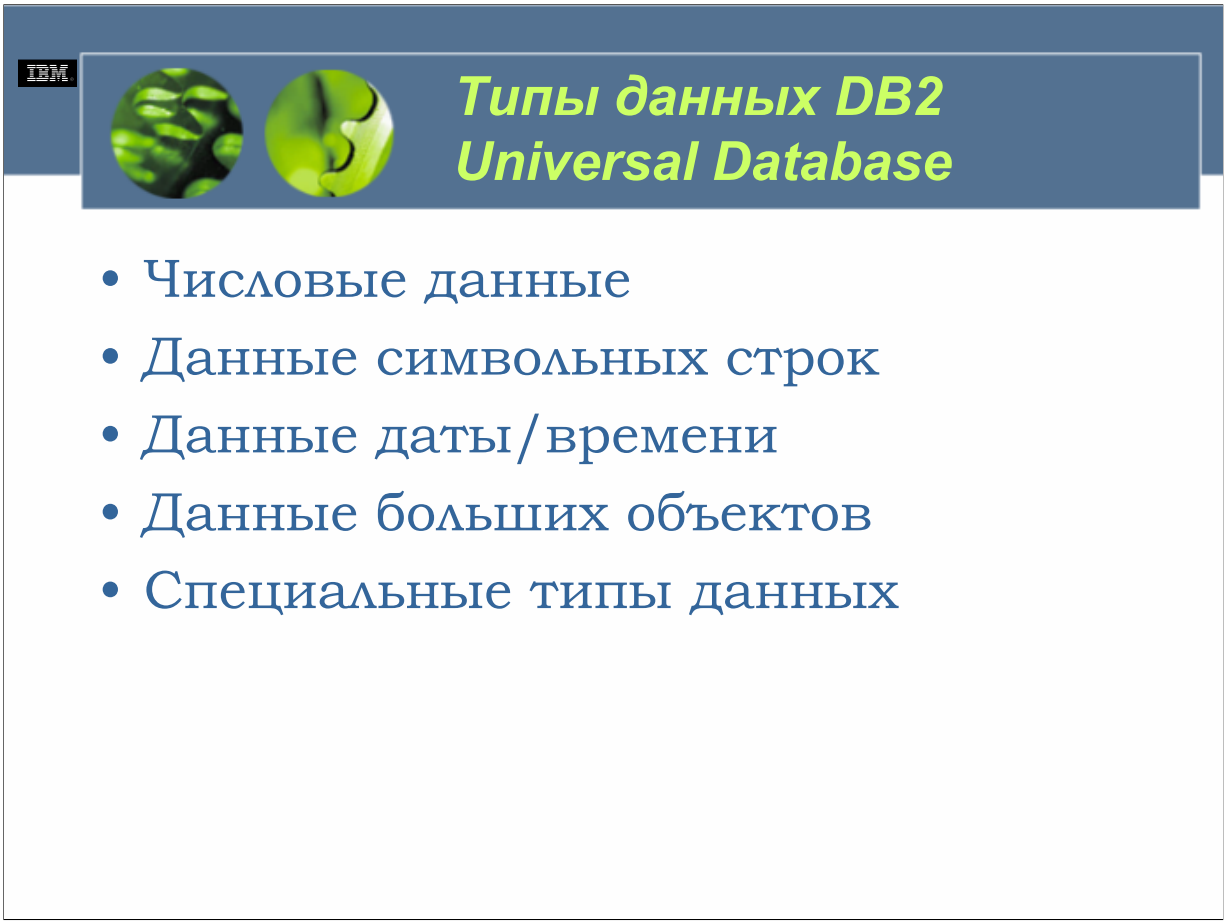

```
SELECT P.PROJNO,P.PROJNAME, D.DEPTNO, D.MGRNO,
       E.LASTNAME AS MGRNAME
FROM PROJECT P
     INNER JOIN DEPARTMENT D  ON P.DEPTNO = D.DEPTNO
     LEFT OUTER JOIN EMPLOYEE E ON D.MGRNO = E.EMPNO
```

RESULT

PROJNO	PROJNAME	DEPTNO	MGRNO	MGRNAME
IF2000	USER EDUCATION	C01	000030	KWAN
IF1000	QUERY SERVICES	C01	000030	KWAN
AD3100	ADMIN SERVICES	D01	-	-

WCF03.0





WCF03.0

Встроенные типы данных, доступные в DB2 UDB, классифицируются в соответствии с типом данных, для хранения которых они были предназначены.

•Числовые данные.

•Данные символьных строк.

•Данные даты/времени.

•Данные больших объектов.

Кроме этих «традиционных» типов данных, доступны также специальные типы данных, использующиеся с DataLinks и модулями расширения DB2 UDB.

WCF03.0

Как предполагает название, числовые типы данных используются для хранения числовых значений, а именно, числовых значений, у которых есть *знак (sign)* и *точность (precision)*. Знак считается положительным, если значение больше или равно нулю, и отрицательным, если значение меньше нуля, тогда как точность является действительным числом цифр, использующимся для представления значения. Числовые данные хранятся с использованием фиксированного объема пространства памяти, а объем необходимой памяти возрастает по мере повышения точности числа.

WCF03.0

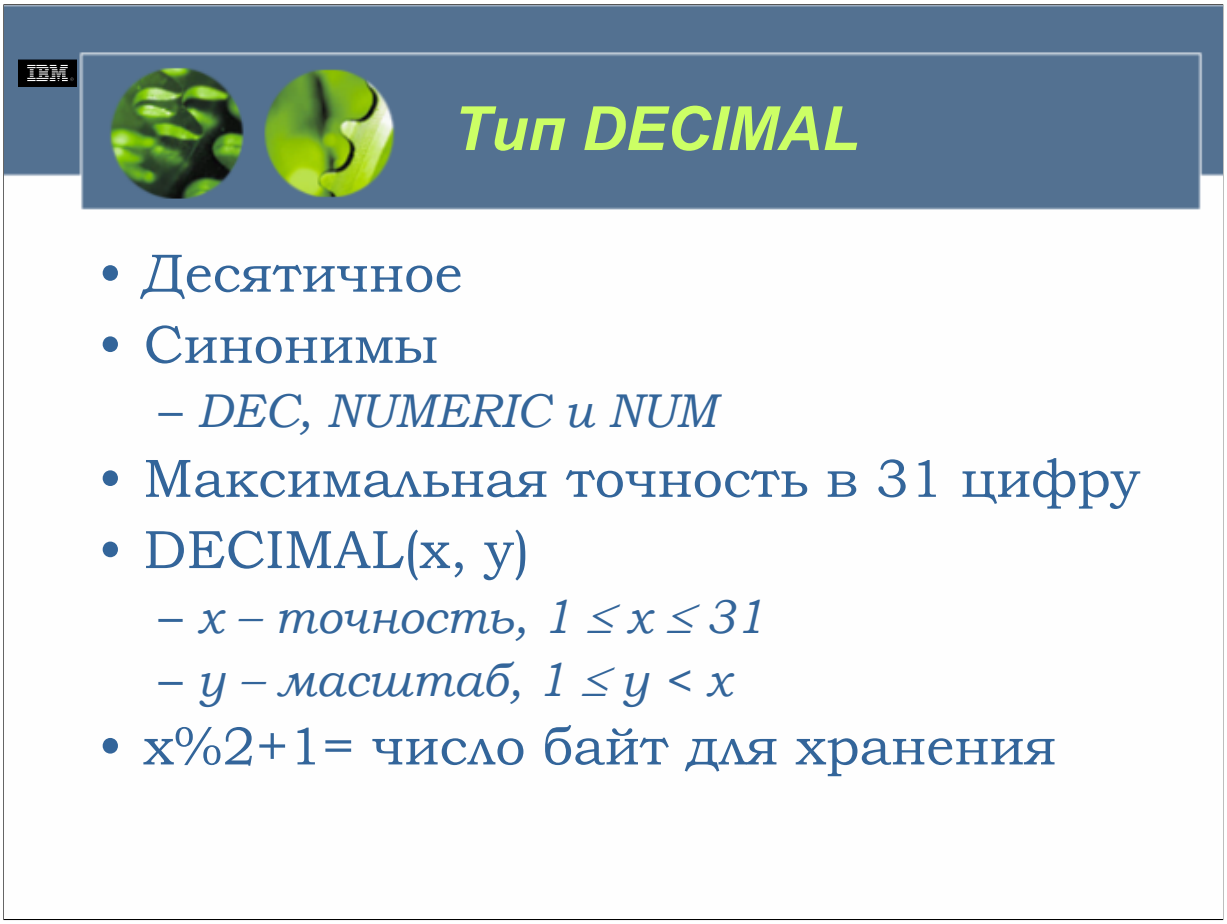
Тип данных короткого целого используется для хранения числовых значений, имеющих точность 5 цифр или менее. Диапазон для коротких целых от –32768 до 32767, для хранения каждого короткого целого требуется 2 байта пространства памяти. (В диапазоне положительных чисел на одно значение меньше, поскольку они начинаются со значения 0, тогда как отрицательные числа начинаются с –1.) Для обозначения типа данных короткого целого используется термин SMALLINT.

WCF03.0

Целый тип данных используется для хранения числовых значений, имеющих точность в 10 цифр. Диапазон целых значений от –2 147 483 648 до 2 147 483 647, а для хранения каждого целого значения используются 4 байта пространства памяти. Для обозначения целого типа данных используются термины INTEGER и INT.

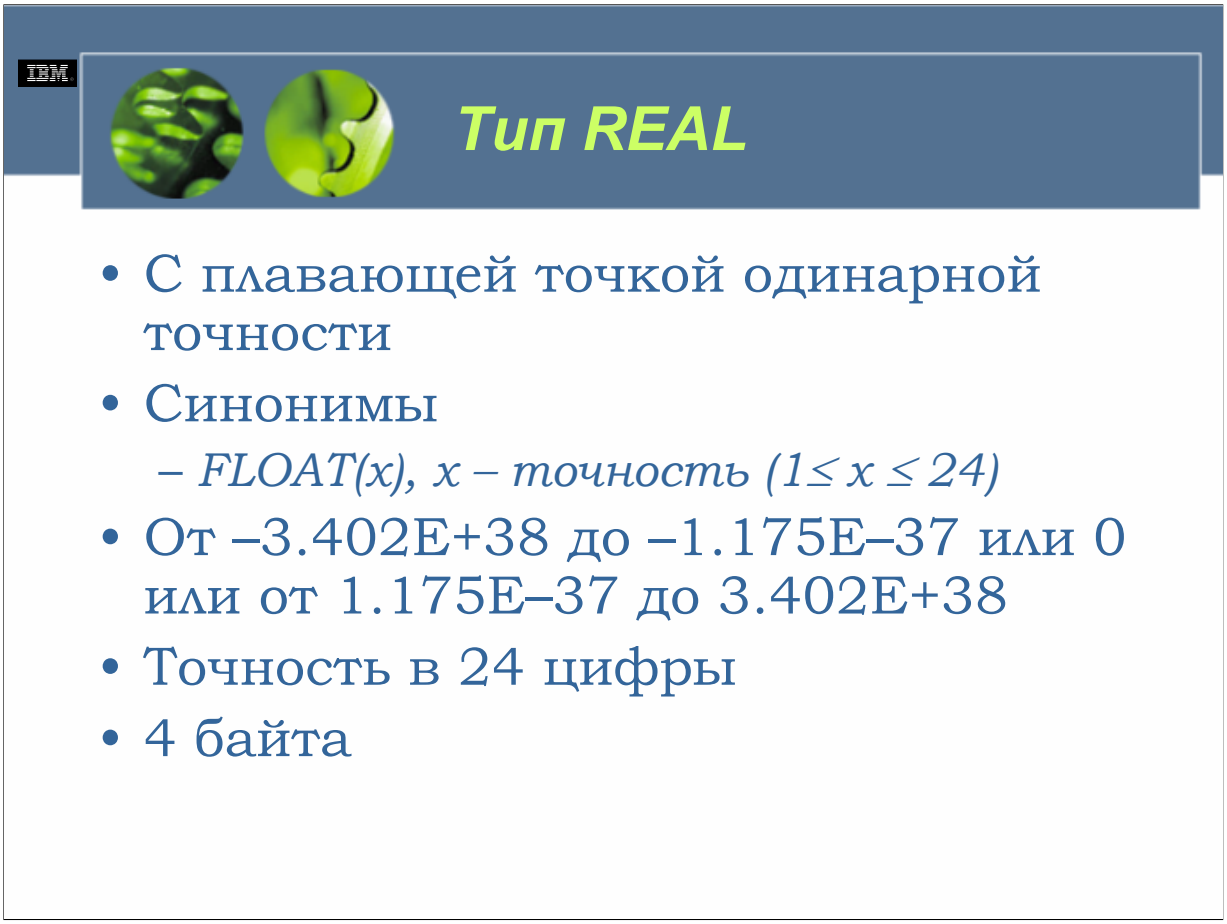
WCF03.0

Тип данных длинного целого используется для хранения числовых значений, имеющих точность в 19 цифр. Диапазон длинных целых значений от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807, а для хранения каждого длинного целого значения требуются 8 байтов пространства памяти. Для обозначения типа данных длинного целого используется термин BININT. (Этот тип данных обычно используется на системах, предоставляющих поддержку 64-разрядных целых; на таких системах обработка больших чисел, которые были сохранены в длинных целых, значительно более эффективна, а все выполняемые вычисления более точны.)



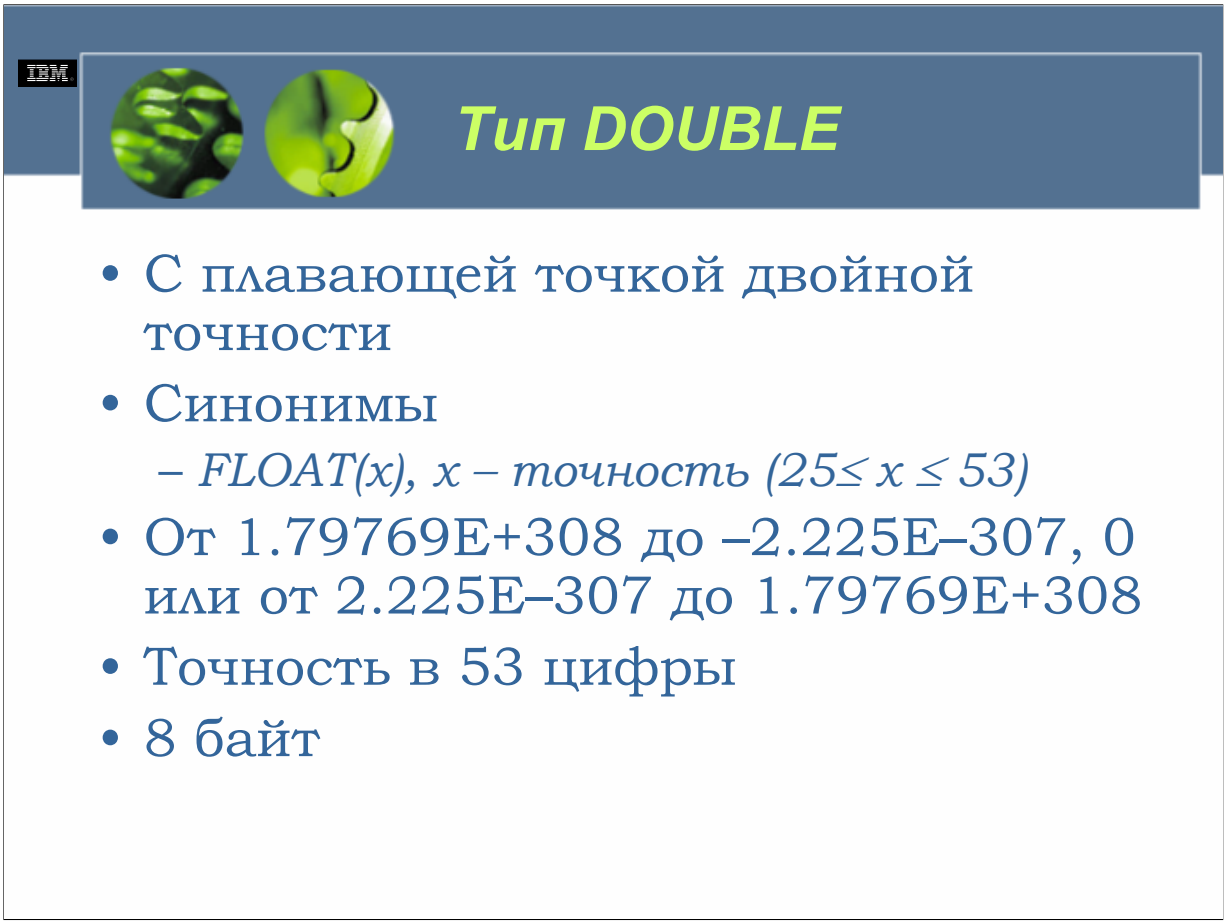
WCF03.0

Десятичный тип данных используется для хранения числовых значений, содержащих как целую, так и дробную части, разделенные десятичной точкой. Точное положение десятичной точки определяется по точности и масштабу значения (масштаб представляет собой число цифр, используемых дробной частью). Максимальная точность, разрешенная для десятичных значений, составляет 31 цифру, а соответствующий масштаб должен быть положительным числом, меньшим точности числа. Объем памяти, необходимый для хранения десятичного значения, можно определить, решив следующее уравнение: *Точность* ч 2 (усеченная) + 1 = *Требуемые байты*. (Например, значение 67,12345 имеет точность 7, 7 ч 2 равно 3, +1 дает 4; следовательно, для хранения значений 67,12345 требуется 4 байта). Для обозначения десятичного типа данных используются термины DECIMAL, DEC, NUMERIC и NUM.



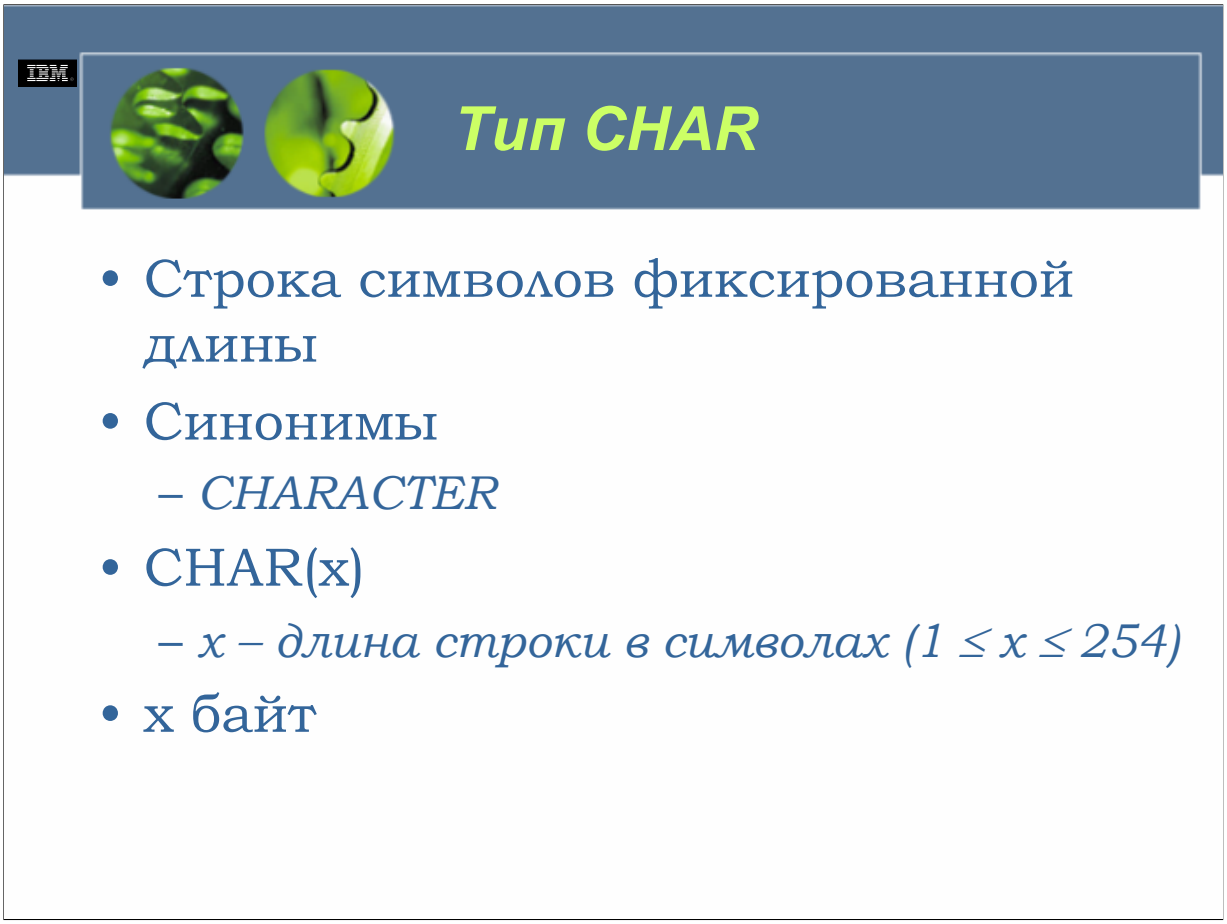
WCF03.0

Тип данных с плавающей точкой одинарной точности используется для хранения 32-разрядного *приближения* реального числа. Это число может быть нулем, или оно может попасть в диапазон от –3.402E+38 до –1.175E–37 или от 1.175E–37 до 3.402E+38. Каждое значение с плавающей точкой одинарной точности может быть до 24 цифр длиной, для хранения каждого значения требуются 4 байта пространства памяти. Для обозначения типа данных с плавающей точкой одинарной точности используются термины REAL и FLOAT.



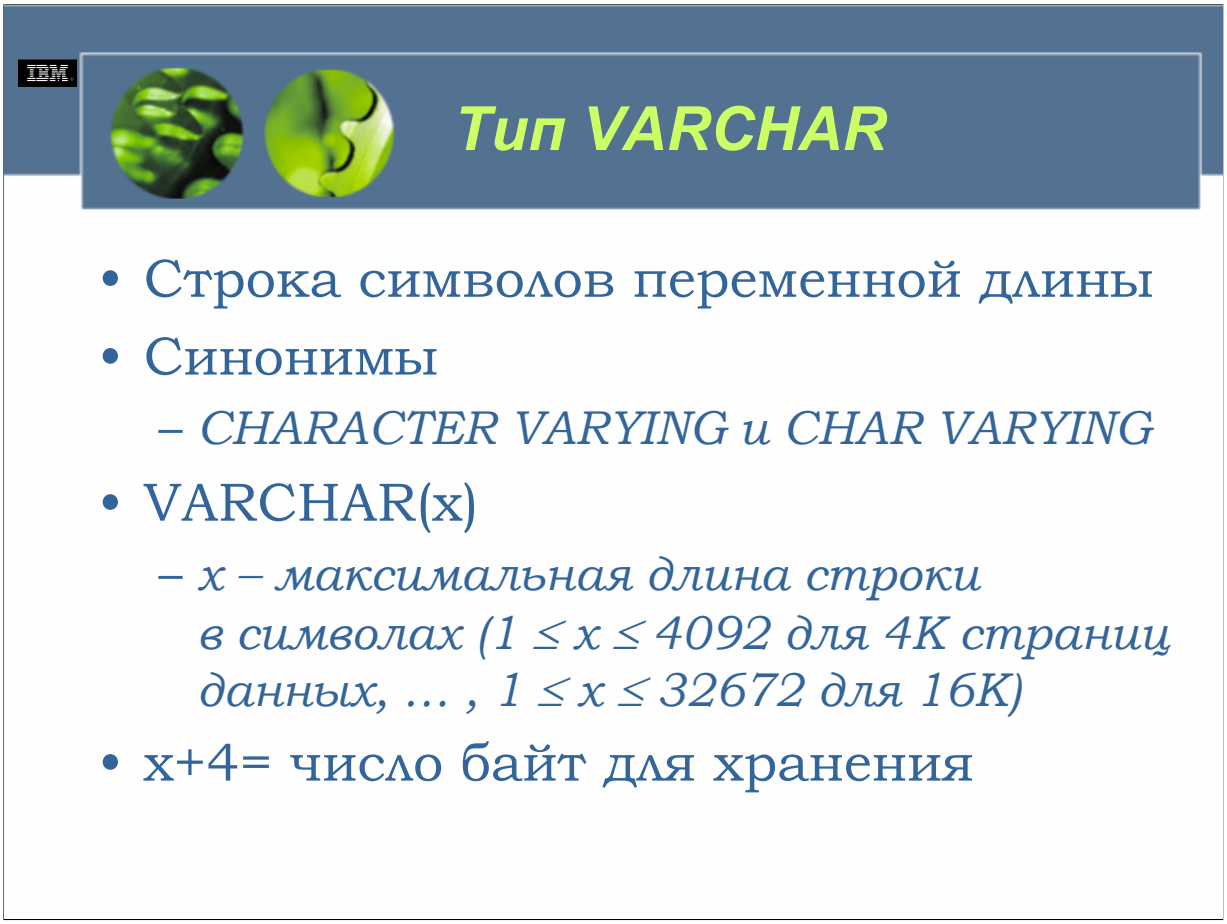
WCF03.0

Тип данных с плавающей точкой двойной точности используется для хранения 64-разрядного *приближения* реального числа. Это число может быть нулем, или оно может попасть в диапазон от –1.79769E+308 до –2.225E–307 или от 2.225E–307 до 1.79769E+308. Каждое значение с плавающей точкой двойной точности может быть до 53 цифр длиной, для его хранения требуются 8 байтов пространства памяти. Для обозначения типа данных с плавающей точкой двойной точности используются термины DOUBLE, DOUBLE PRECISION и FLOAT.



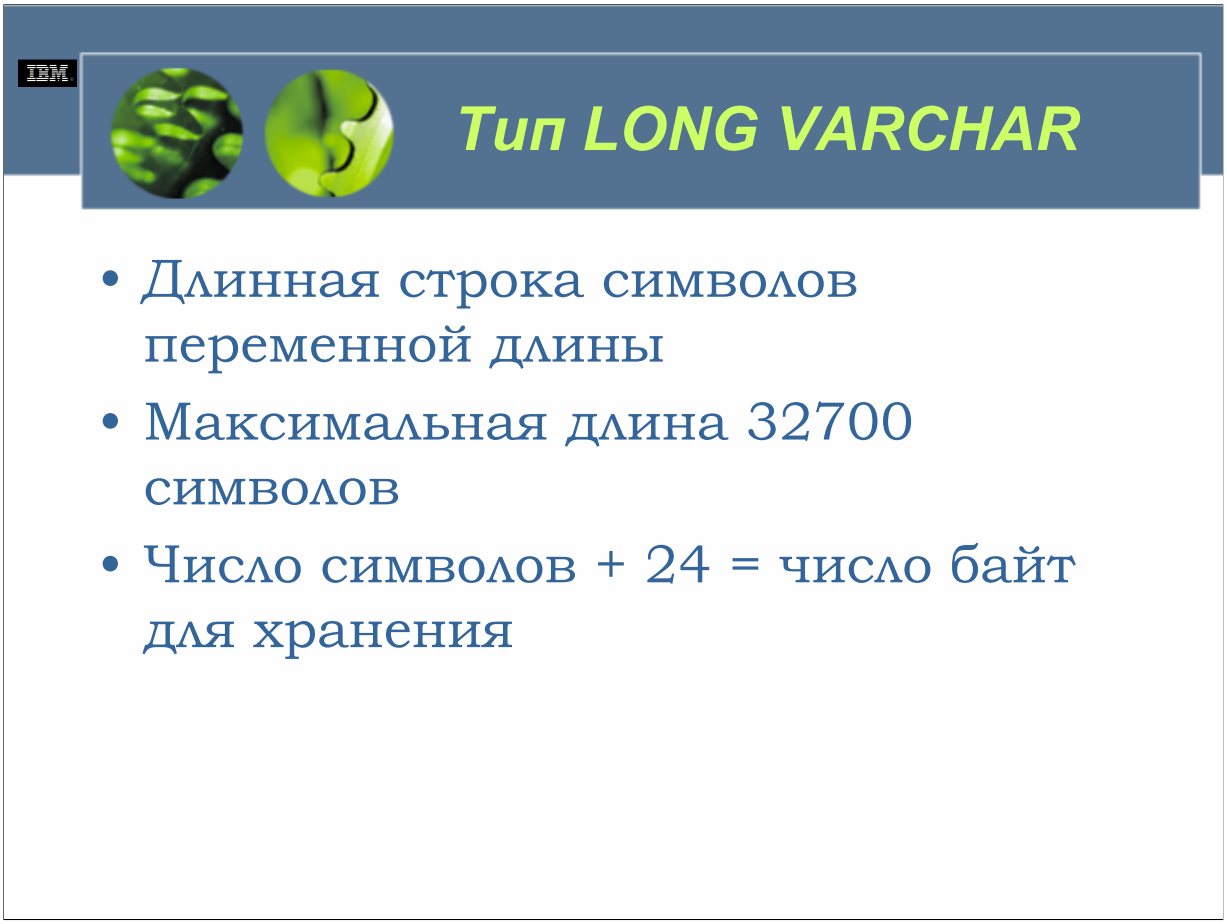
WCF03.0

Тип данных строки символов фиксированной длины используется для хранения строковых значений, имеющих длину от 1 до 254 символов. Объем, необходимый для хранения значения строки символов фиксированной длины, можно определить, решив следующее уравнение: (*Число символов* x 1) = *Число требуемых байтов*. (Выделяется фиксированный объем пространства памяти, даже если нужно не все выделенное пространство; короткие строки дополняются пробелами.) Для обозначения типа данных строки символов фиксированного размера используются термины CHARACTER и CHAR.



WCF03.0

Тип данных строки символов переменной длины используется для хранения значений данных символьных строк, имеющих длину до 32672 символов. Однако действительная допустимая длина определяется используемым размером страницы табличного пространства. Для таблиц, которые находятся в табличных пространствах, использующих 4 Кб страницы, размер значений строк символов переменной длины не может превышать 4092 символов; для таблиц, которые находятся в табличных пространствах, использующих 8 Кб страницы, размер значений строк символов переменной длины не может превышать 8188 символов и т. д. Объем пространства памяти, необходимого для хранения значения строки символов переменного размера, можно определить, решив следующее уравнение: (*Число символов* x 1) + 4 = *Число требуемых байтов*. (Выделяется лишь действительно необходимый объем пространства памяти плюс четыре байта для символа «конца строки»? строки не дополняются пробелами.) Для обозначения типа данных строк символов переменной длины используются термины CHARACTER VARYING, CHAR VARYING и VARCHAR.



WCF03.0

Тип данных длинной строки символов переменной длины используется для хранения значений символьных строк, имеющих длину до 32700 символов, независимо от размера страницы используемого табличного пространства. Объем пространства памяти, необходимой для хранения значения длинной строки символов переменной длины, можно определить, решив следующее уравнение: (*Число символов* x 1) + 24 = *Число требуемых байтов*. Для обозначения типа данных длинной строки символов переменной длины используется термин LONG VARCHAR.

WCF03.0

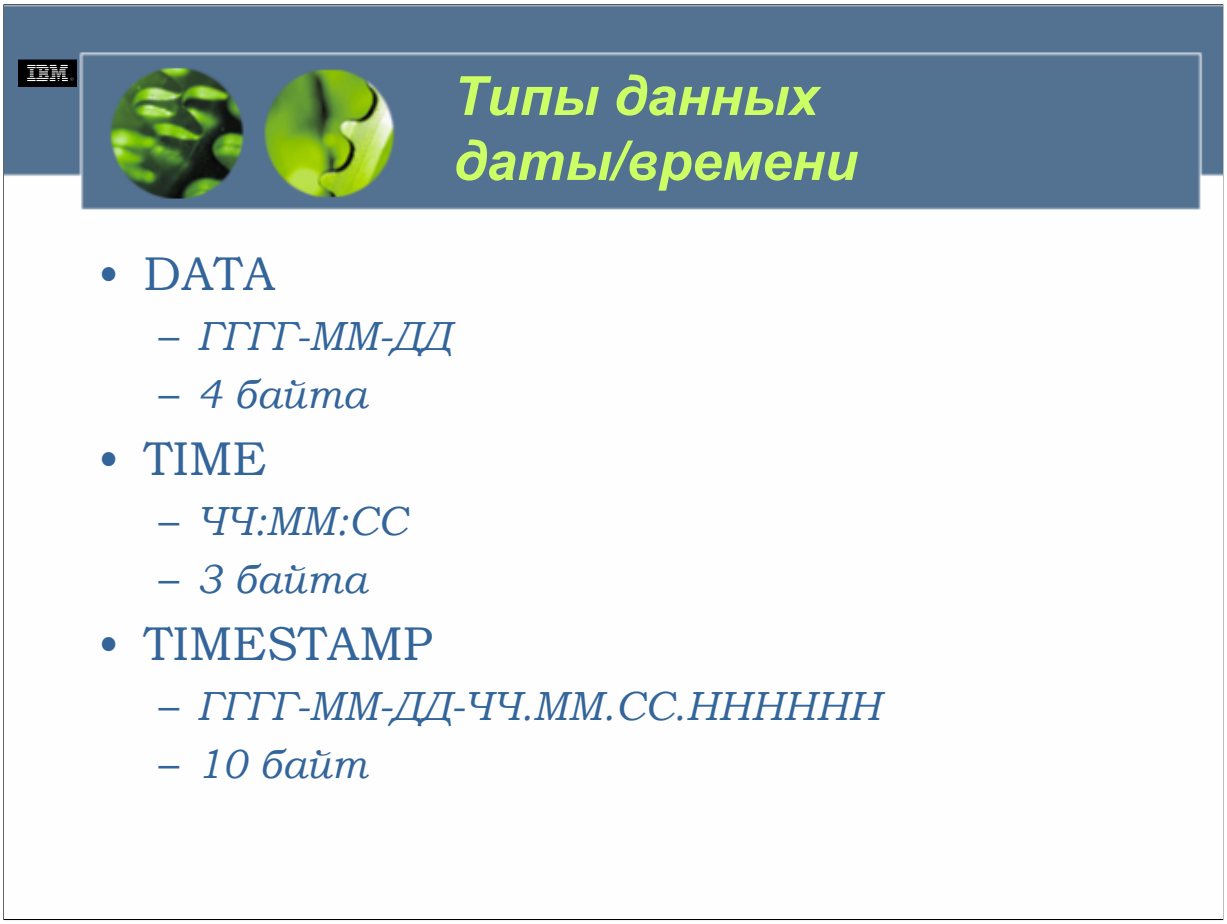
Тип данных строки двухбайтных символов фиксированной длины используется для хранения значений строк символов DBCS (набора двухбайтных символов), имеющих длину до 127 символов. (Большинство азиатских наборов символов является наборами двухбайтных символов.) Объем пространства памяти, необходимый для хранения значения строки двухбайтных символов фиксированной длины можно определить, решив следующее уравнение: (*Число символов* x 2) = *Число требуемых байтов*. Для обозначения типа данных строки двухбайтных символов фиксированной длины используется термин GRAPHIC.

WCF03.0

Тип данных строки двухбайтных символов переменной длины используется для хранения значений строк символов DBCS, имеющих длину до 16336 символов. Опять-таки, действительная допустимая длина определяется размером страницы используемого табличного пространства. Для таблиц, находящихся в табличных пространствах, использующих 4 Кб страницы, размер значения строк двухбайтных символов переменной длины не может превышать 2046 символов; для таблиц, которые находятся в табличных пространствах, использующих 8 Кб страницы, размер значений строк двухбайтных символов переменной длины не может превышать 4094 символов, и т. д. Объем пространства памяти, необходимый для хранения значения строки двухбайтных символов переменной длины, можно определить, решив следующее уравнение: (*Число символов* x 2) + 4 = *Число требуемых байтов*. Для обозначения типа данных строки двухбайтных символов переменной длины используется термин VARGRAPHIC.

WCF03.0

Тип данных длинной строки двухбайтных символов переменной длины используется для хранения значений строк символов DBCS, имеющих длину до 16350 символов, независимо от размера страницы используемого табличного пространства. Объем пространства памяти, необходимый для хранения значения длинной строки двухбайтных символов переменной длины, можно определить, решив следующее уравнение: (*Число символов* x 2) + 24 = *Число требуемых байтов*. Для обозначения типа данных длинной строки двухбайтных символов переменной длины используется термин LONG VARGRAPHIC.



WCF03.0

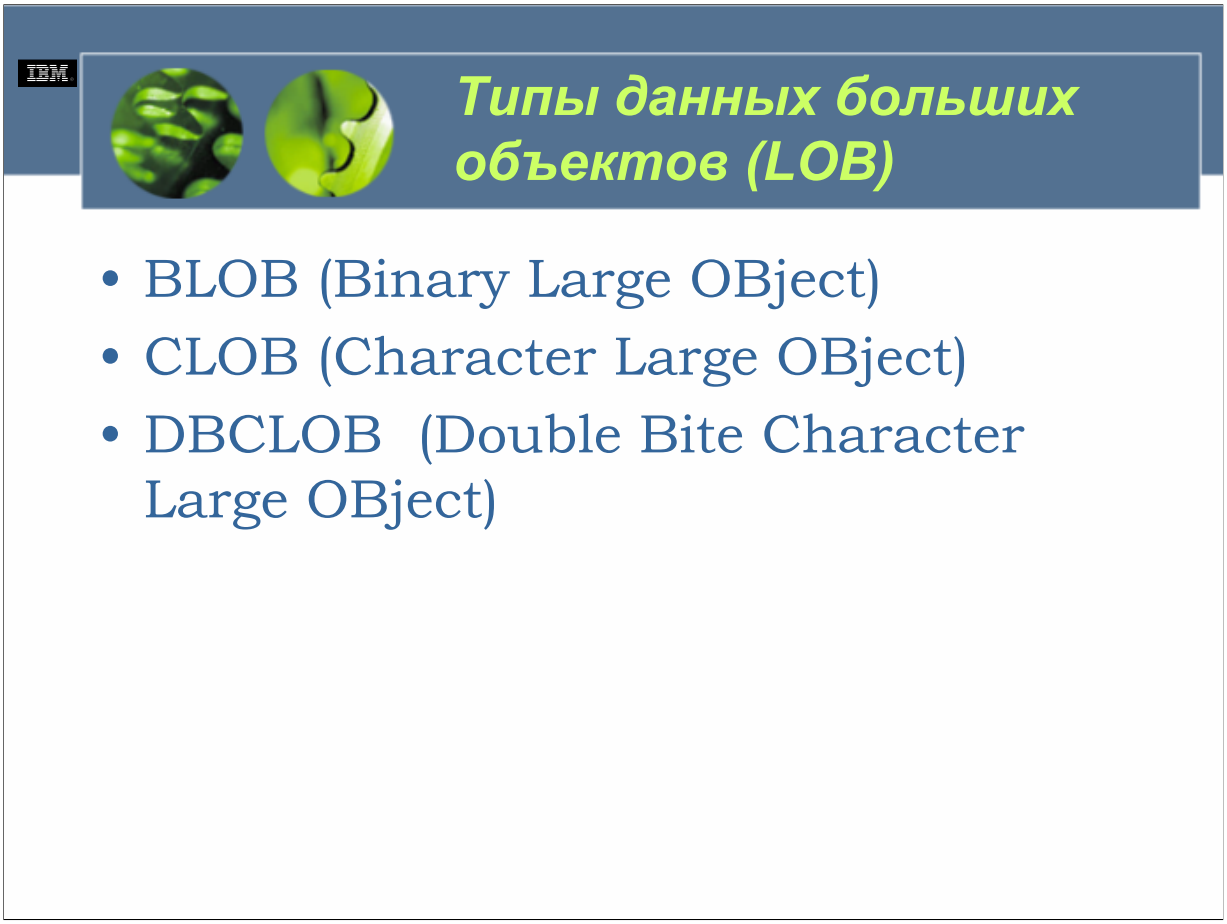
Дата. Тип данных даты используется для хранения значений, состоящих из трех частей (года, месяца и дня), которые представляют календарные даты. Диапазон значений для года составляет от 0001 до 9999; диапазон значений для месяца – от 1 до 12; а диапазон значений для дня от – 1 до 28, 29, 30 или 31 в зависимости от указанного значения месяца и от того, является год високосным или нет. Внешне значения даты представлены как значения символьных строк фиксированной длины с размером в 10 символов. Однако для хранения каждого значения даты требуется лишь 4 байта пространства памяти. Для обозначения типа данных даты используется термин DATA.

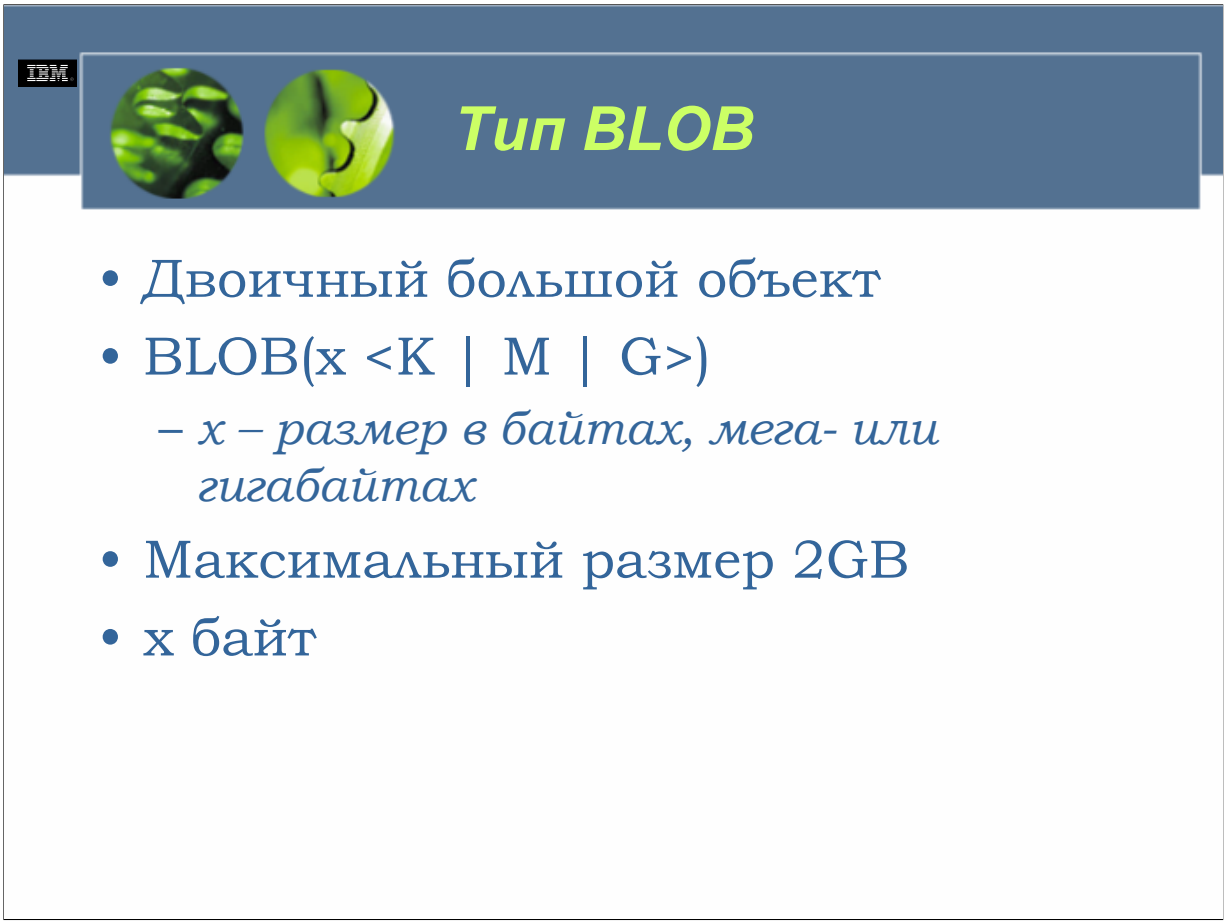
Время. Тип данных времени используется для хранения значений, составленных из трех частей (часов, минут и секунд), которые представляют время, с использованием 24-часовых часов. Диапазон значений для часов составляет от 0 до 24; диапазон для минут – от 0 до 59; а диапазон для секунд – от 0 до 59. Внешне значения времени представлены как строковые значения фиксированной длины, имеющие размер 8 символов. Однако для хранения каждого значения времени требуется лишь 3 байта пространства памяти. Для обозначения типа данных времени используется термин TIME.

Отметка времени. Тип данных отметки времени используется для хранения значений, состоящих из семи частей (года, месяца, дня, часов, минут, секунд и микросекунд), которые представляют определенную календарную дату и время (с использованием 24-часовых часов). Диапазон значений для года составляет от 0001 до 9999; диапазон для месяца – от 1 до 12; диапазон для дня – от 1 до 28, 29, 30 или 31 в зависимости от указанного значения месяца и от того, является указанный год високосным или нет; диапазон для часов – от 0 до 24; диапазон для минут – от 0 до 59; диапазон для секунд – от 0 до 59; и диапазон для микросекунд – от 0 до 999999. Внешне значения временных отметок представлены в виде символьных строк фиксированной длины размером 26 символов (эта строка отображается в виде *ГГГГ-ММ-ДД-ЧЧ.ММ.СС.КККККК*, которая переводится в Год-Месяц-День-Час.Минута.Секунда.Микросекунда). Однако для хранения каждого значения отметки времени требуется лишь 10 байтов пространства памяти. Для обозначения типа данных отметки времени используется термин TIMESTAMP.

Форматы даты и времени DB2 UDB

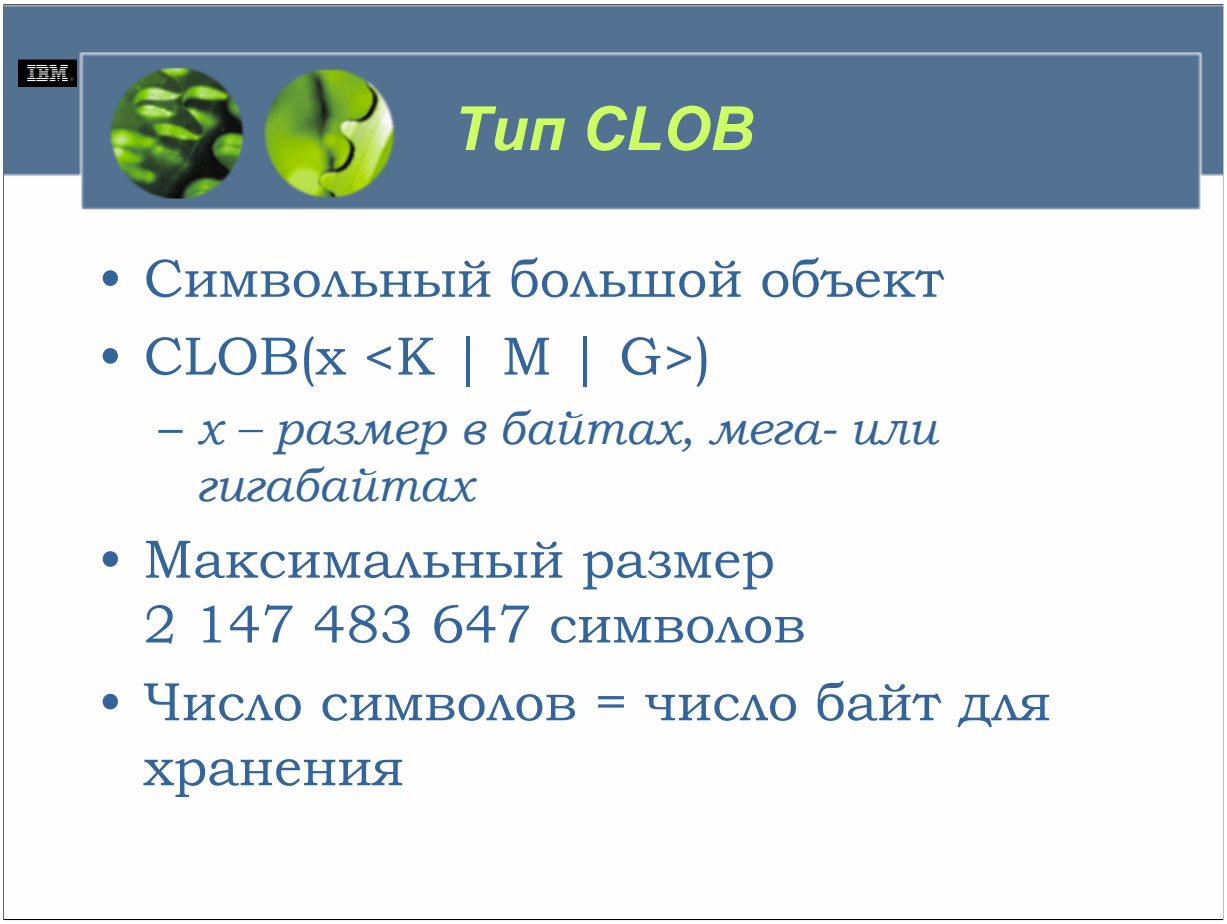
Название формата	Сокращение	Формат строк даты	Формат строк времени
Международная организация по стандартизации	ISO	ГГГГ-ММ-ДД	ЧЧ.ММ.СС
Стандарт IBM США	USA	ММ/ДД/ГГГГ	ЧЧ:ММ AM или PM
Европейский стандарт IBM	EUR	ДД.ММ.ГГГГ	ЧЧ.ММ.СС
Промышленный стандарт Японии	JIS	ГГГГ-ММ-ДД	ЧЧ:ММ:СС
Специфично для места	LOC	Основан на коде территории и страны базы данных	Основан на коде территории и страны базы данных





WCF03.0

Тип данных двоичного большого объекта используется для хранения значений двоичных данных (таких, как документы, графические изображения, картины, аудио и видео), которые имеют размер до 2 Гигабайт. Для обозначения типа данных двоичного большого объекта используются термины BINARY LARGE OBJECT и BLOB. Объем пространства памяти, зарезервированного для хранения значения двоичного большого объекта, определяется спецификацией размера, предоставляемой при определении типа данных двоичного большого объекта. Например, для определения BLOB(800) будет зарезервировано 800 байтов пространства памяти.

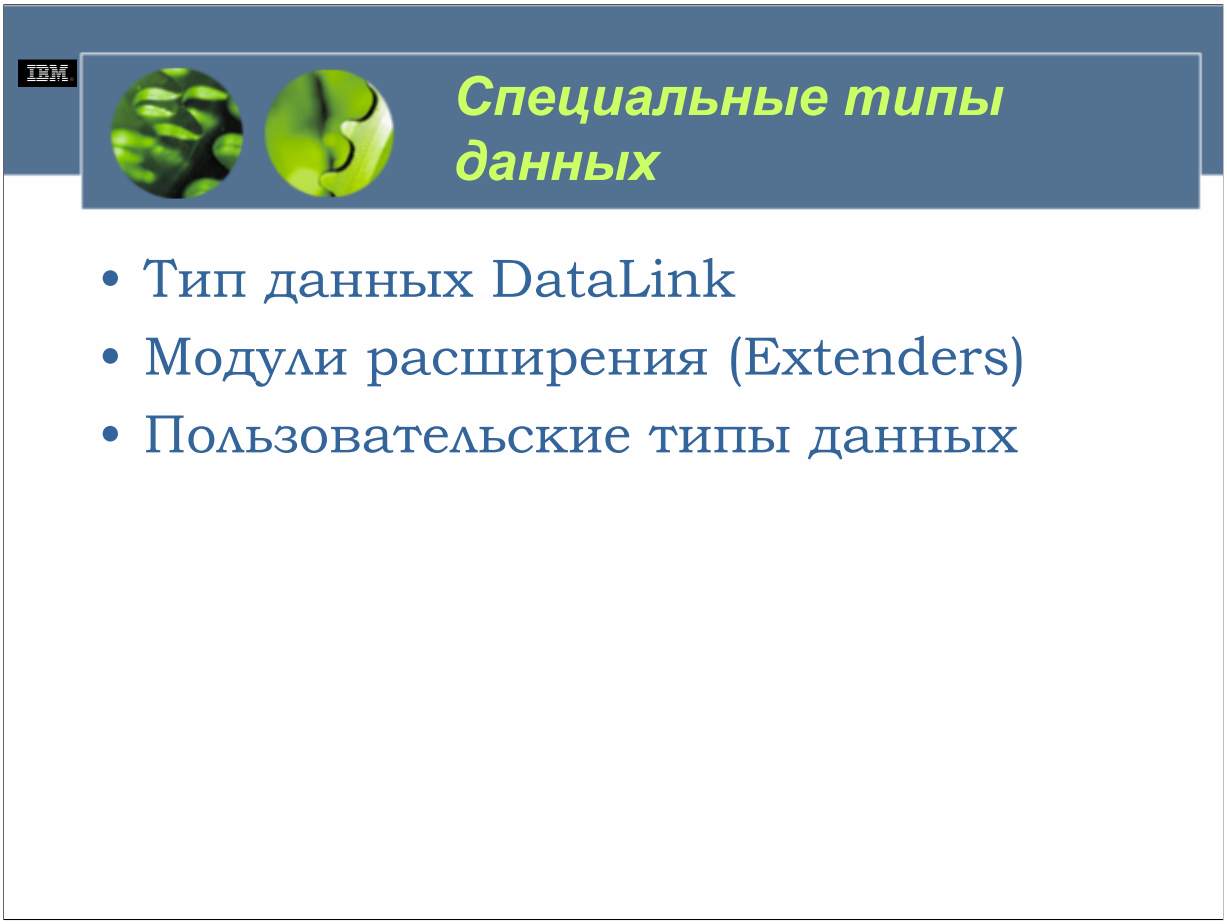


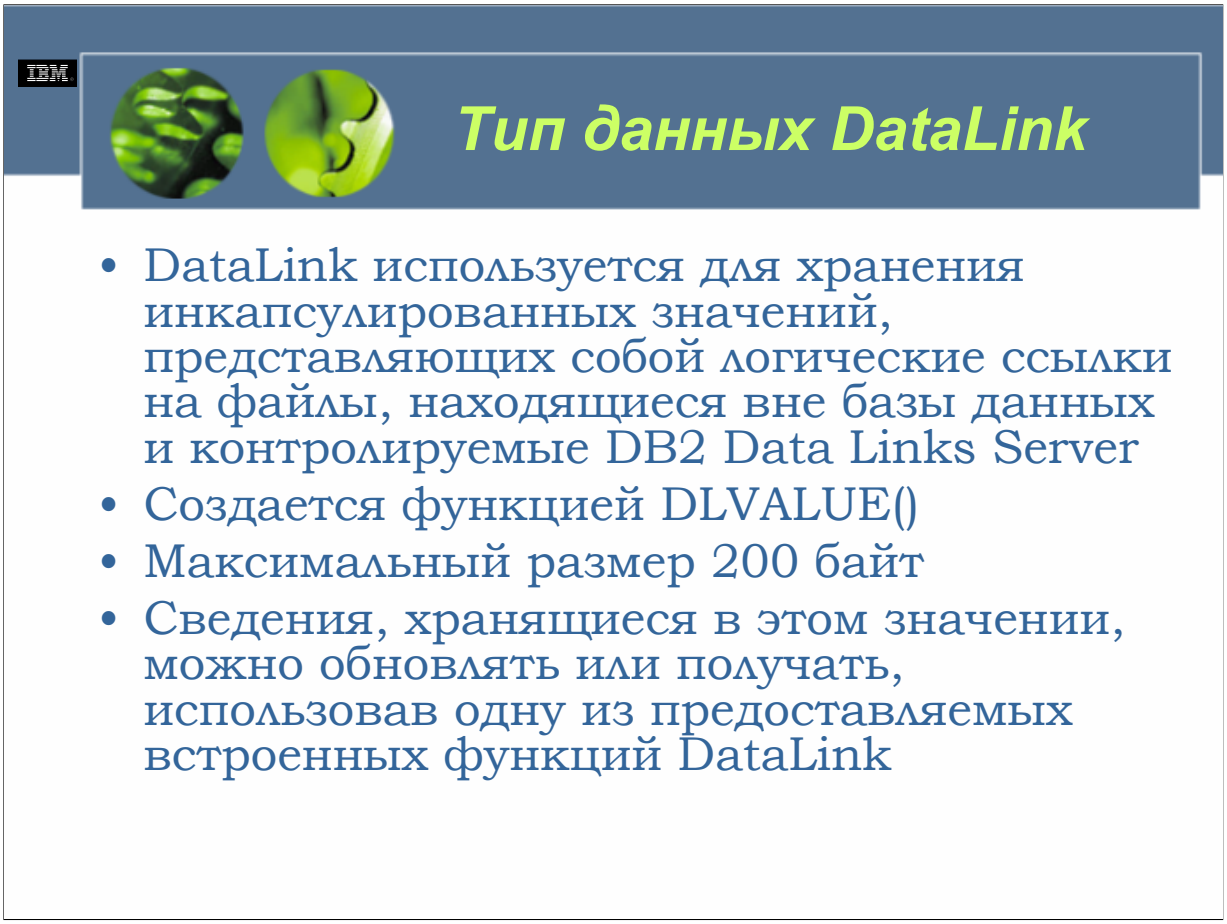
WCF03.0

Тип данных символьного большого объекта используется для хранения значений строк символов SBCS (набор однобайтных символов) или MBCS (набор многобайтных символов), имеющих размеры от 32700 до 2 147 483 647 символов. Для обозначения типа данных символьных больших объектов используются термины CHARACTER LARGE OBJECT, CHAR LARGE OBJECT и CLOB. Объем пространства памяти, резервируемой для хранения значения символьного большого объекта, определяется спецификацией размера, предоставленной при определении типа данных символьного большого объекта. Например, для определения CLOB(800) было бы зарезервировано 800 байтов пространства памяти.

WCF03.0

Тип данных двухбайтного символьного большого объекта используется для хранения значений строк символов DBCS (набора двухбайтных символов), имеющих размер от 16350 до 1 073 741 823 символов. Для обозначения типа данных двухбайтных символьных больших объектов используется термин DBCLOB. Объем пространства памяти, зарезервированной для значения двухбайтного символьного большого объекта, определяется спецификацией размера, представленной при определении типа данных двухбайтного символьного большого объекта. Например, для определения DBCLOB(400) было бы зарезервировано 800 байтов.



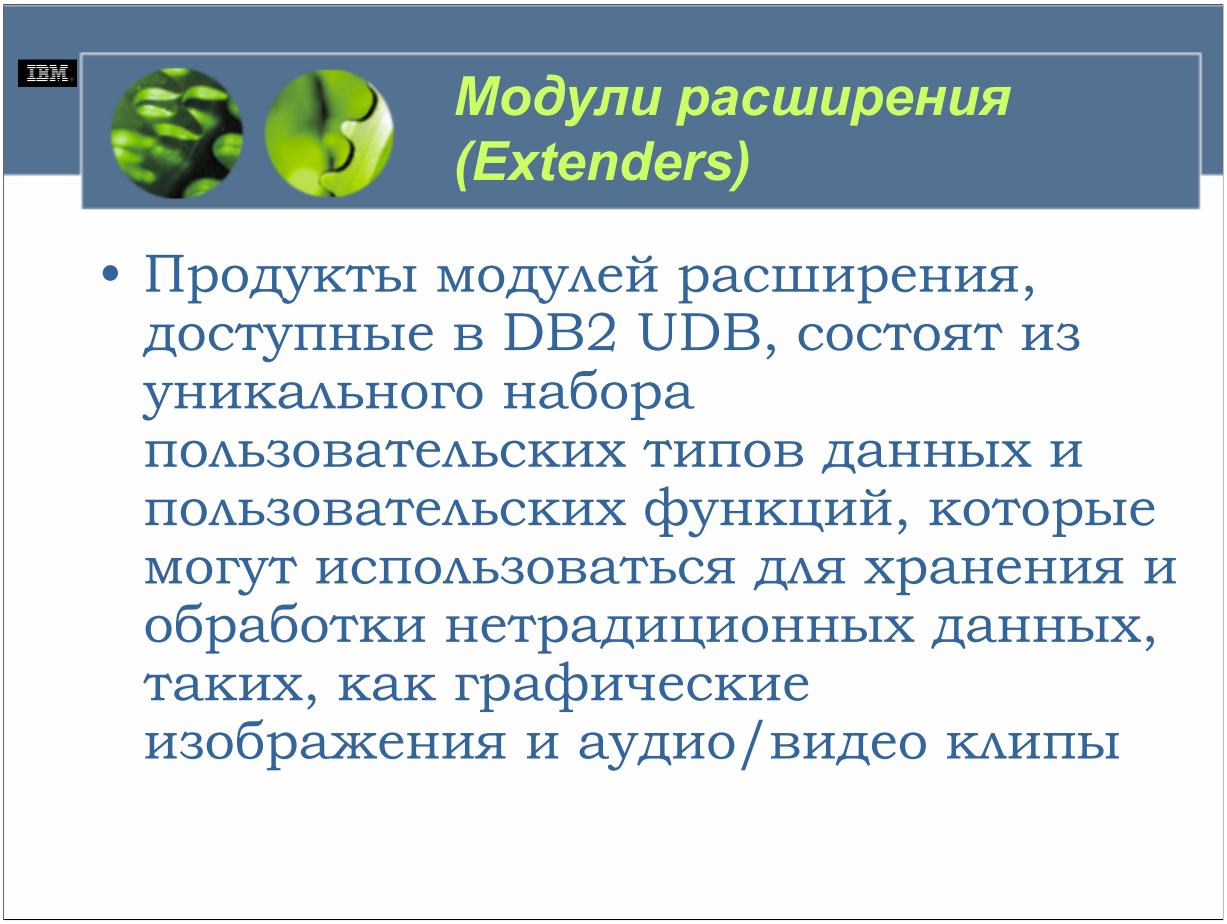


WCF03.0

Тип данных DataLink используется для хранения инкапсулированных значений, представляющих собой логические ссылки на файлы, находящиеся вне базы данных и контролируемые DB2 Data Links Server. (DB2 Data Links Server является добавочным пакетом, который захватывает управление над файловой системой от имени базы данных DB2 UDB.) Значения DataLink служат в качестве «значений привязки», содержащих ссылочную информацию, позволяющую базе данных устанавливать и поддерживать связь с внешними данными. Для обозначения типа данных DataLink используется термин DATALINK.

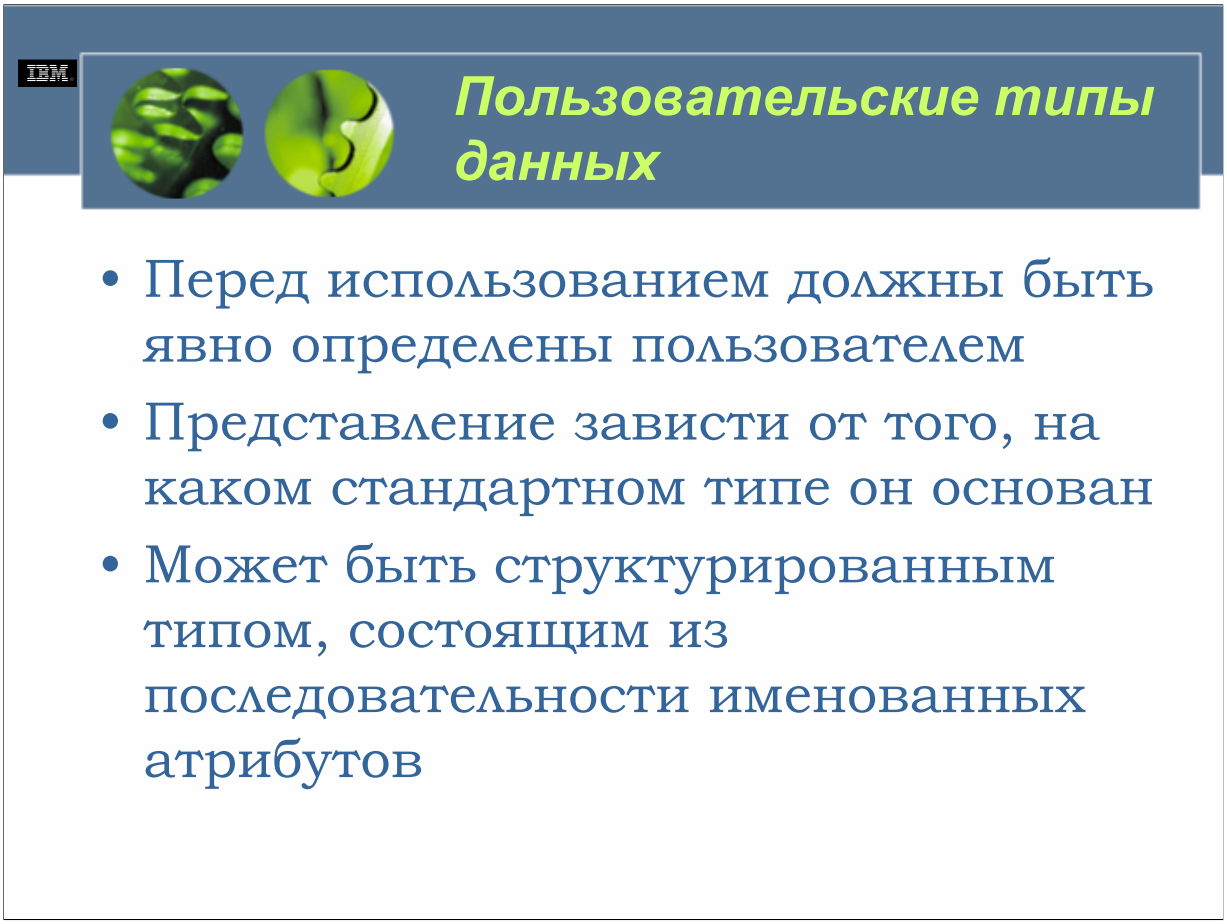
В средах File System Migrator (FSM), NT File System (NTFS), Journaled File System (JFS) и UNIX File System (UFS) значения DataLink состоят из имени сервера Data Links Manager, который содержит файл, который должен быть связан извне, вместе с именем самого файла, объединенных для создания адреса универсального указателя ресурса (URL). Затем этот адрес может быть объединен с текстом описательного комментария, который может иметь размер до 200 символов. Однако окончательное созданное значение DataLink не может превышать 200 байтов (для хранения каждого значения DataLink требуется 200 байтов пространства памяти).

Поскольку значения DataLink являются инкапсулированными значениями, содержащими несколько порций информации, они должны создаваться с использованием встроенной функции DLVALUE(). Когда создано инкапсулированное значение DataLink, специфические сведения, хранящиеся в этом значении, можно обновлять или получать, использовав одну из предоставляемых встроенных функций DataLink (DLCOMMENT(), DLLINKTYPE(), DLNEWCOPY(), DLPREVIOUSCOPY(), DLREPLACECONTENT(), DLURLCOMPLETE(), DLURLCOMPLETEONLY(), DLURLCOMPLETEWRITE(), DLURLPATH(), DLURLPATHONLY(), DLURLSCHEME() и DLURLSERVER()).



WCF03.0

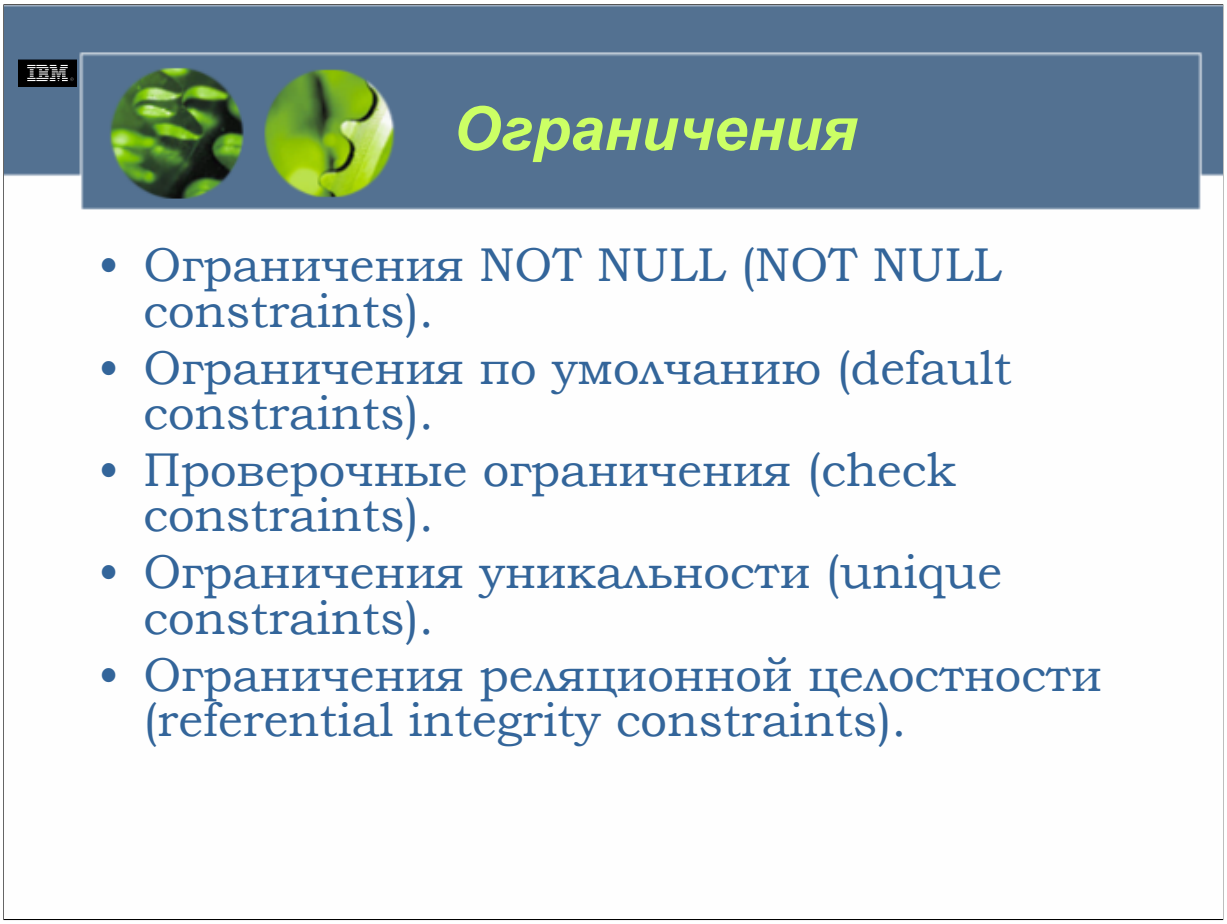
Продукты модулей расширения, доступные в DB2 UDB, состоят из уникального набора пользовательских типов данных и пользовательских функций, которые могут использоваться для хранения и обработки нетрадиционных данных, таких, как графические изображения и аудио/видео клипы. Поскольку многие из предоставляемых модулями расширения DB2 типы данных основаны на встроенных типах данных, ограничения размеров и требования к памяти типа данных модуля расширения часто соответствуют прототипам встроенных типов данных – при условии, что данные модуля расширения действительно хранятся в базе данных. Некоторые из модулей расширения DB2 позволяют хранить свои данные во внешних файлах, которые находятся вне базы данных, в то время как информация о размещении самих файлов хранится в таблице базы данных. В таком случае требования к памяти для типа данных модуля расширения может быть гораздо меньше, чем у соответствующего эквивалента встроенного типа данных.



WCF03.0

пользовательские типы данных (UDT) являются типами данных, создаваемых явным образом пользователем базы данных. Пользовательский тип данных может быть особым типом данных, который совместно использует общее представление с одним из встроенных типов данных, предоставляемых DB2 UDB, или он может быть структурированным типом, состоящим из последовательности именованных атрибутов, у каждого из которых свой собственный тип данных. Структурированные типы данных могут также создаваться в виде подтипов других структурированных типов, определяя тем самым иерархию типов.

Пользовательские типы данных являются предметом строгой типизации данных, что означает, что даже хотя они могут совместно использовать общее представление с другими встроенными или пользовательскими типами данных, значение одного пользовательского типа данных совместимо лишь со значениями того же самого типа (или другого пользовательского типа данных в рамках той же самой иерархии типов). В результате пользовательские типы данных не могут быть использованы в качестве аргументов для большинства встроенных функций. Однако могут быть созданы пользовательские функции и операторы, дублирующие функциональные возможности, предоставляемые встроенными функциями.



WCF03.0

В большинстве предприятий данные часто должны придерживаться определенного набора правил и ограничений. Например, компании обычно имеют определенный формат и числовую последовательность, которую они используют, создавая заказы покупок. Аналогично триггерам, ограничения позволяют вам размещать логику, необходимую для обеспечения таких бизнес-правил, непосредственно в базу данных, а не в приложения, которые работают с базой данных. В сущности, триггеры являются наборами действий, которые должны выполняться каждый раз, когда для определенной таблицы проводится операция вставки, изменения или удаления. Ограничения, с другой стороны. являются правилами, которые управляют тем, как к таблице данных могут быть добавлены значения данных, а также как эти значения могут быть изменены после того, как они были добавлены. Доступны следующие типы ограничений.

•Ограничения NOT NULL (NOT NULL constraints).

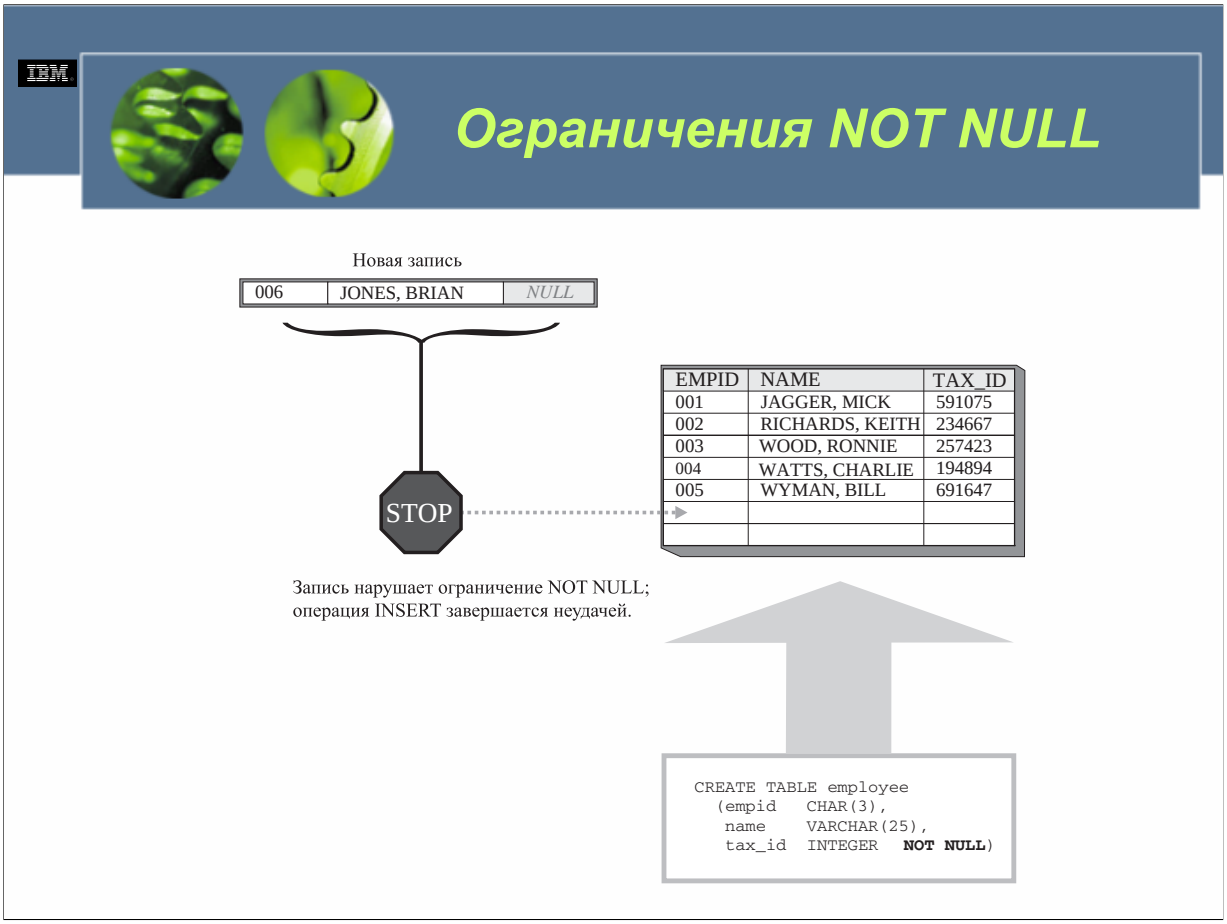
•Ограничения по умолчанию (default constraints).

•Проверочные ограничения (check constraints).

•Ограничения уникальности (unique constraints).

•Ограничения реляционной целостности (referential integrity constraints).

Ограничения обычно определяются в ходе создания таблицы; однако ограничения могут также быть добавлены в существующие таблицы с использованием оператора SQL ALTER TABLE.



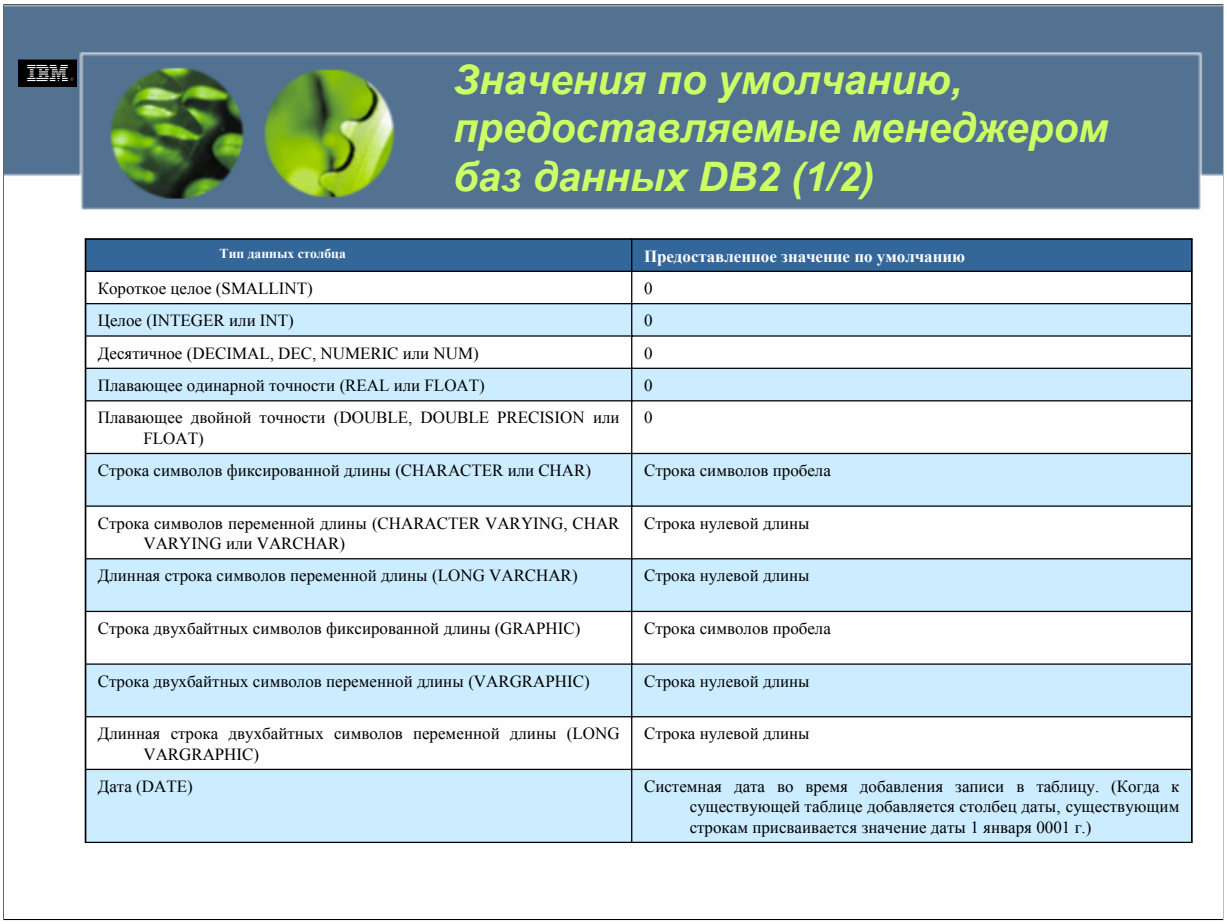
WCF03.0

В DB2 UDB пустые (null) значения (не путать с пустыми строками) используются для представления отсутствующих или неизвестных данных и/или состояний. И по умолчанию каждый столбец в таблице принимает пустое значение. Это дает вам возможность добавлять данные в таблицу, когда известны не все значения, принадлежащие записи. Однако бывают случаи, когда это поведение неприемлемо (например, для каждого работающего в компании сотрудника могут требоваться налоговые идентификационные номера). При возникновении такой ситуации для обеспечения того, что определенному столбцу в базовой таблице никогда не будет присвоено пустое значение, может быть использовано ограничение NOT NULL; после определения для столбца ограничения NOT NULL каждая операция, которая пытается поместить в эту строку пустое значение, завершится неудачей.

Поскольку ограничения NOT NULL связаны с определенным столбцом в базовой таблице, они обычно определяются в ходе процесса создания таблицы.

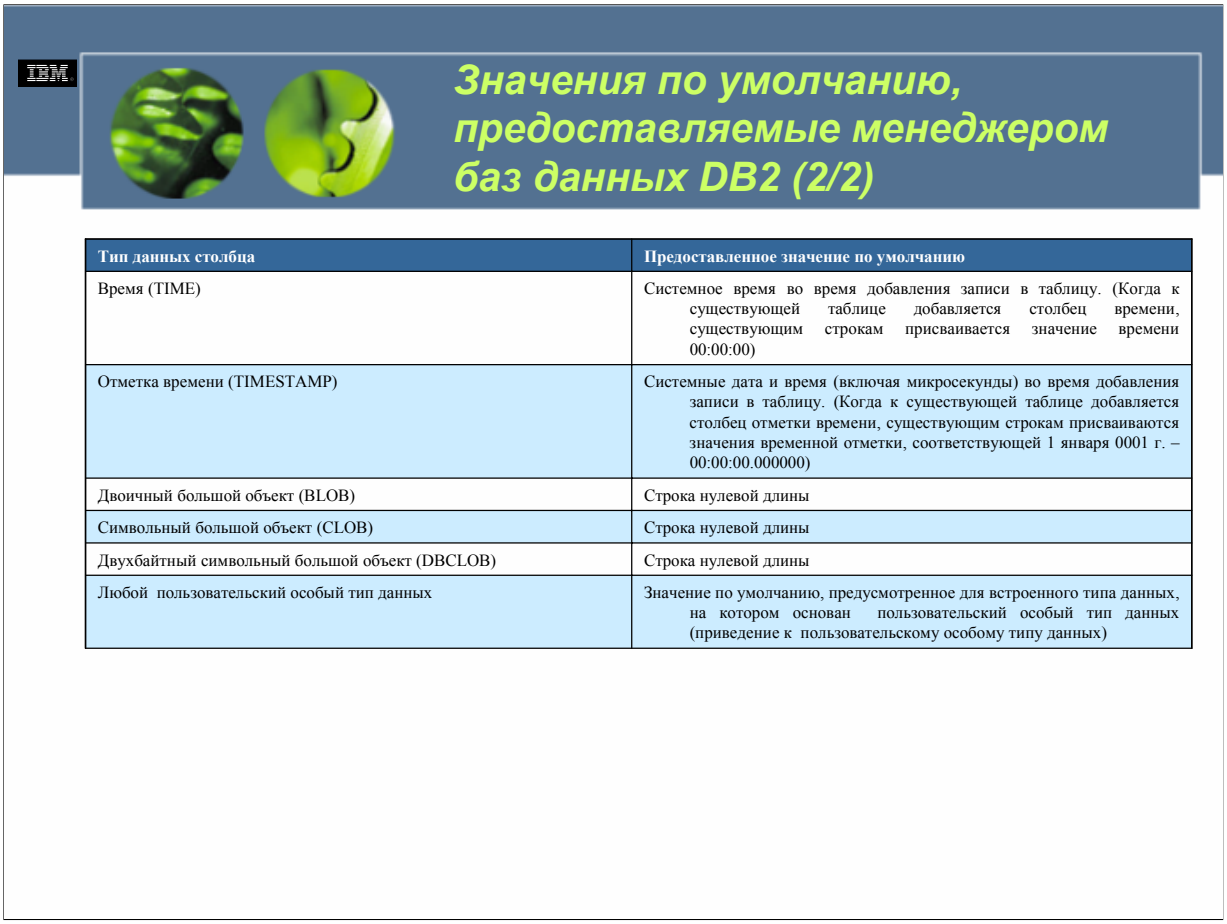
WCF03.0

Точно так же, как бывают ситуации, когда нежелательно принимать пустые значения, могут быть ситуации, когда желательно, чтобы система предоставляла для вас определенное значение (например, вы могли бы захотеть автоматически назначать определенному столбцу текущую дату каждый раз при добавлении к таблице новой записи). В таких случаях можно использовать ограничение по умолчанию, чтобы обеспечить назначение определенному столбцу в базовой таблице предопределенного значения (если только это значение не заменяется) каждый раз при добавлении записи к таблице. Предоставляемое предопределенное значение могло бы быть пустым (если для столбца не определено ограничение NOT NULL), предоставленным пользователем значением, тип которого совместим с типом данных столбца, или значением, предоставленным менеджером баз данных DB2.

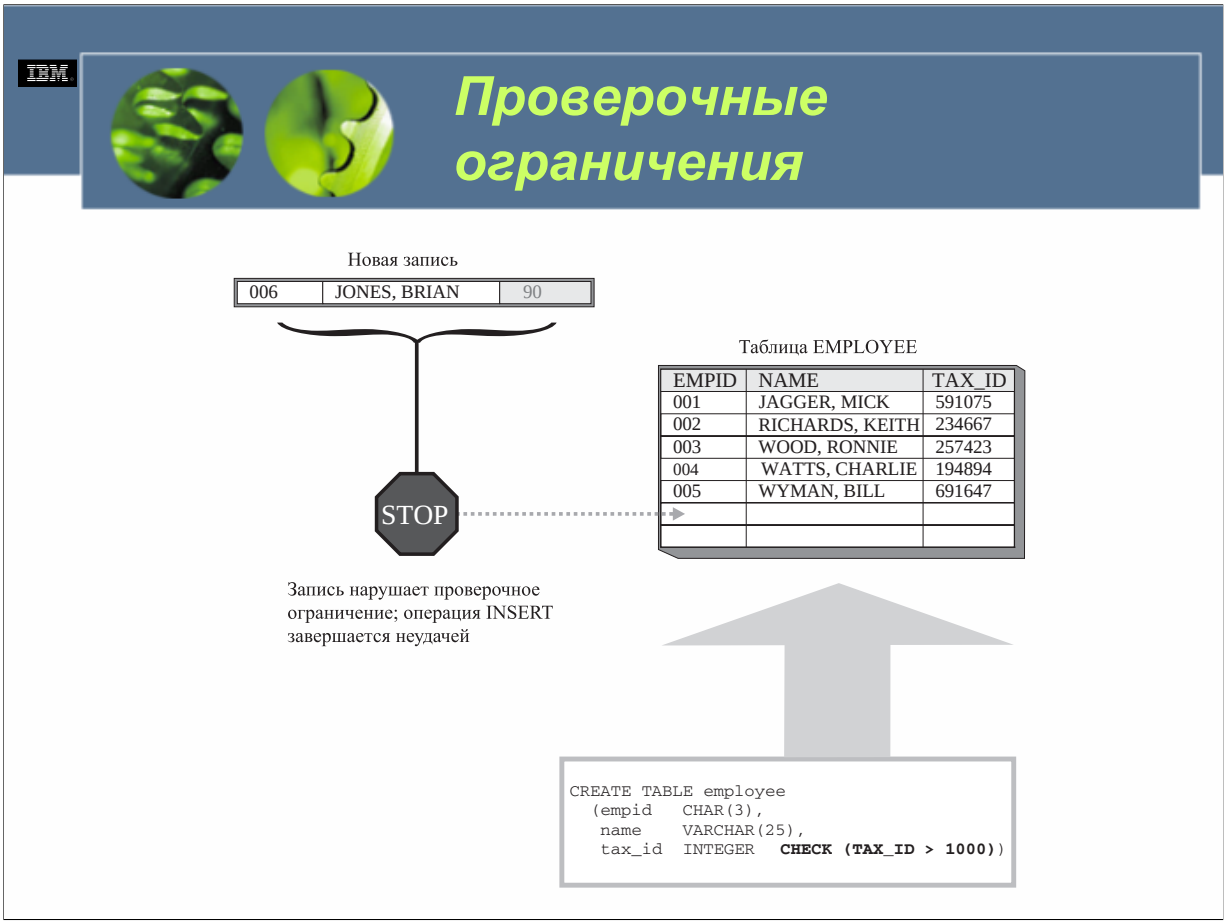


Значения по умолчанию, предоставляемые менеджером баз данных DB2 (1/2)

Тип данных столбца	Предоставленное значение по умолчанию
Короткое целое (SMALLINT)	0
Целое (INTEGER или INT)	0
Десятичное (DECIMAL, DEC, NUMERIC или NUM)	0
Плавающее одинарной точности (REAL или FLOAT)	0
Плавающее двойной точности (DOUBLE, DOUBLE PRECISION или FLOAT)	0
Строка символов фиксированной длины (CHARACTER или CHAR)	Строка символов пробела
Строка символов переменной длины (CHARACTER VARYING, CHAR VARYING или VARCHAR)	Строка нулевой длины
Длинная строка символов переменной длины (LONG VARCHAR)	Строка нулевой длины
Строка двухбайтных символов фиксированной длины (GRAPHIC)	Строка символов пробела
Строка двухбайтных символов переменной длины (VARGRAPHIC)	Строка нулевой длины
Длинная строка двухбайтных символов переменной длины (LONG VARGRAPHIC)	Строка нулевой длины
Дата (DATE)	Системная дата во время добавления записи в таблицу. (Когда к существующей таблице добавляется столбец даты, существующим строкам присваивается значение даты 1 января 0001 г.)



Тип данных столбца	Предоставленное значение по умолчанию
Время (TIME)	Системное время во время добавления записи в таблицу. (Когда к существующей таблице добавляется столбец времени, существующим строкам присваивается значение времени 00:00:00)
Отметка времени (TIMESTAMP)	Системные дата и время (включая микросекунды) во время добавления записи в таблицу. (Когда к существующей таблице добавляется столбец отметки времени, существующим строкам присваиваются значения временной отметки, соответствующей 1 января 0001 г. – 00:00:00.000000)
Двоичный большой объект (BLOB)	Строка нулевой длины
Символьный большой объект (CLOB)	Строка нулевой длины
Двухбайтный символьный большой объект (DBCLOB)	Строка нулевой длины
Любой пользовательский особый тип данных	Значение по умолчанию, предусмотренное для встроенного типа данных, на котором основан пользовательский особый тип данных (приведение к пользовательскому особому типу данных)



WCF03.0

Иногда желательно контролировать то, какие значения для определенного элемента будут приняты, а какие нет (например, компания могла бы решить, что всем не освобожденным от налогов сотрудникам следует платить как минимум минимальную федеральную заработную плату). В таких случаях логика, необходимая для определения того, приемлемо ли значение, может быть включена непосредственно в программу ввода данных, которая используется для сбора данных. Лучшим способом достичь тех же самых целей является определение проверочного ограничения для столбца в базовой таблице, который должен получить значение данных. Проверочное ограничение (известное также под именем *проверочного ограничения таблицы*) может использоваться для обеспечения того, что определенному столбцу в базовой таблице никогда не будет присвоено неприемлемое значение – после определения для столбца проверочного ограничения любая операция, пытающаяся ввести в столбец значение, не удовлетворяющее определенным требованиям, завершится неудачей.

Проверочные ограничения составлены из одного или нескольких предикатов (соединенных ключевыми словами AND или OR), которые вместе называются *условием проверКи (check condition)*. Это условие проверки сравнивается затем с представленным значением данных, а результат этого сравнения возвращается в виде значения «TRUE», «FALSE» или «Unknown». Если проверочное ограничение возвращает значение «TRUE», значение приемлемо, поэтому оно добавляется в базу данных. Если, с другой стороны, проверочное ограничение возвращает значение «FALSE» или «Unknown», операция, пытающаяся поместить это значение в базу данных, завершается неудачей, а от всех изменений, сделанных этой операцией, база данных отказывается. Однако важно отметить, что когда результаты определенной операции откатываются из-за нарушения проверочного ограничения, транзакция, вызвавшая эту операцию, не завершается, и другие операции в рамках этой транзакции не затрагиваются.

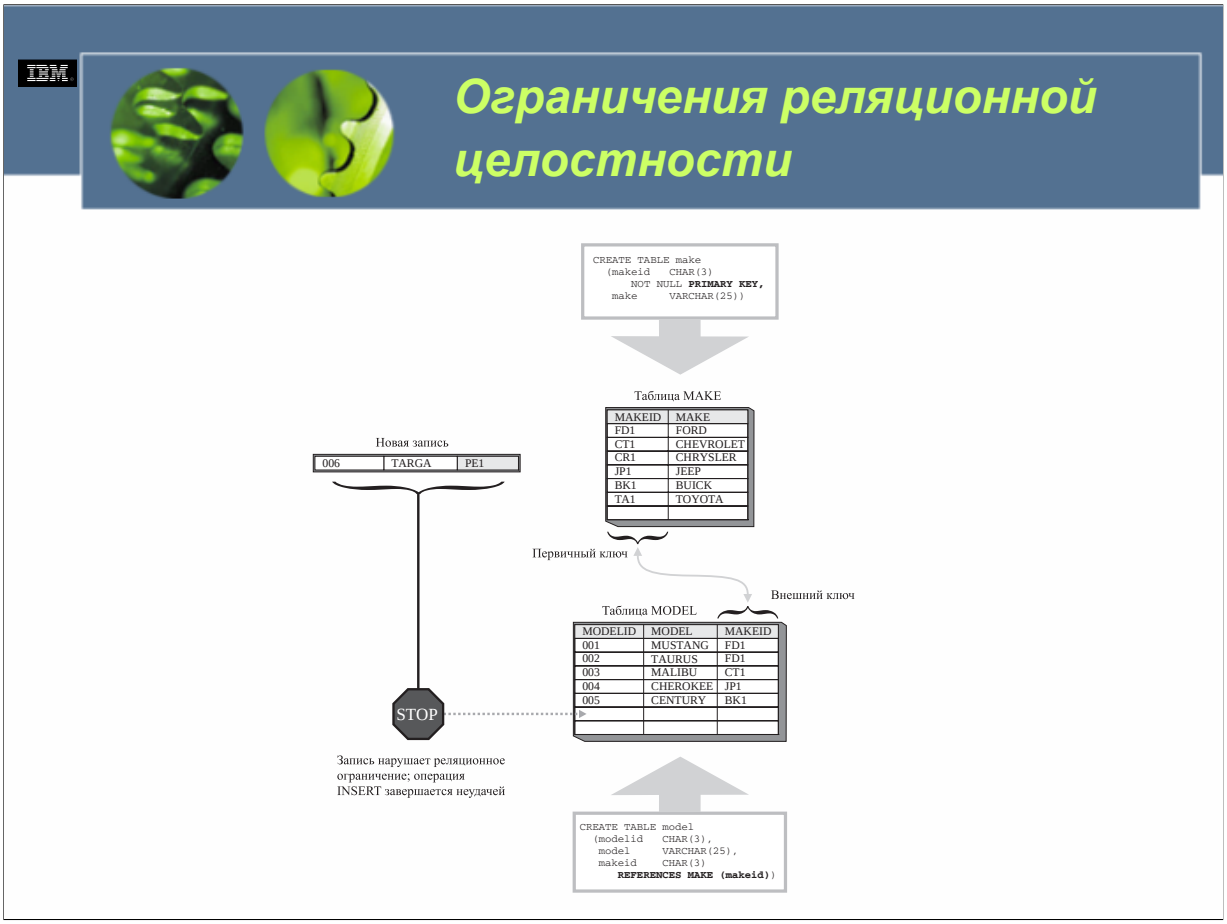
WCF03.0

По умолчанию записи, добавляемые в базовую таблицу, могут иметь одни и те же значения, которые назначены любому доступному столбцу любое число раз. Пока записи, хранящиеся в таблице, не содержат информацию, которая не должна дублироваться, такой вид поведения допустим. Однако бывают ситуации, когда определенная часть информации, составляющая запись, должна быть уникальной (например, если идентификационный номер сотрудника назначен каждому служащему, который работает в определенной компании, каждый номер должен быть уникальным – двум сотрудникам никогда не следует присваивать один и тот же идентификационный номер сотрудника). В таких ситуациях может использоваться ограничение уникальности для обеспечения того, что значение(-я), присваиваемое(-ые) одному или нескольким столбцам при добавлении записи в базовую таблицу, всегда уникально; после определения для одного или нескольких столбцов ограничения уникальности любая операция, пытающаяся поместить в эти столбцы дублированные значения, потерпит неудачу.

В отличие от ограничений NOT NULL, ограничений по умолчанию и проверочных ограничений, которые связаны лишь с одним столбцом в базовой таблице, ограничения уникальности могут быть связаны с отдельным столбцом или с группой столбцов. Однако подобно другим ограничениям, ограничения уникальности обычно определяются в процессе создания таблицы.

Независимо от того, когда определено ограничение уникальности, когда оно создано, менеджер баз данных DB2 проверяет, существует ли уже индекс для столбцов, на которые ссылается ограничение уникальности. Если да, этот индекс помечается как уникальный и системный. Если нет, создается соответствующий индекс и помечается как уникальный и системный. Затем этот индекс используется для обеспечения уникальности каждый раз при добавлении новых записей в столбец (столбцы), для которых было определено ограничение уникальности.

Таблица может иметь любое число ограничений уникальности; однако в таблице может быть не более одного ограничения уникальности, определенного для одного и того же набора столбцов. А поскольку ограничения уникальности обеспечиваются индексами, все ограничения, накладываемые на индексы (например, максимум в 16 столбцов с общей допустимой длиной 255 байтов; ни один из использованных столбцов не может иметь тип данных большого объекта или длинной строки символов и т. д.), накладываются и на ограничения уникальности.



WCF03.0

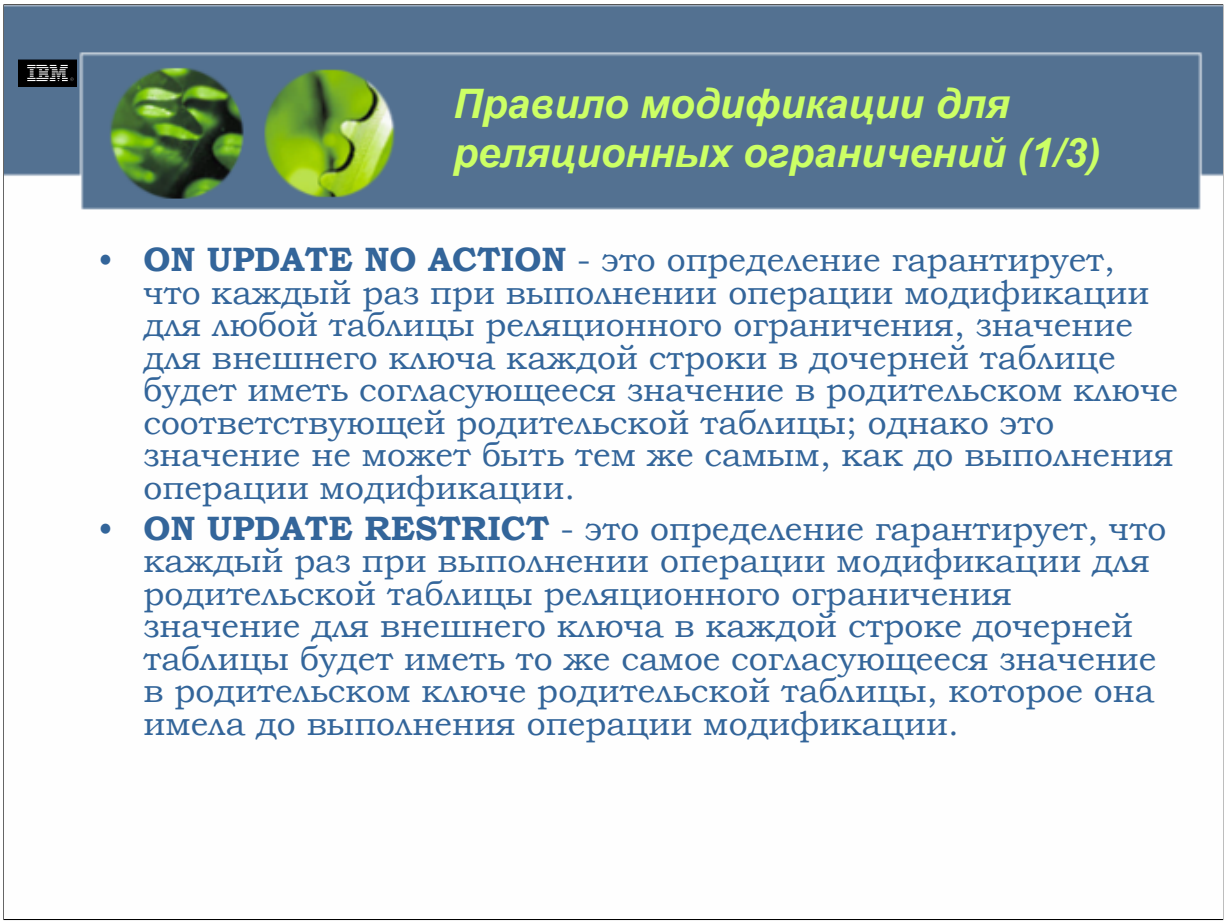
Понять, как работают реляционные ограничения, помогает пример. Предположим, вы владеете небольшим магазином авто запчастей, и для отслеживания имеющихся в распоряжении товаров вы используете базу данных. Многие из запчастей, которые вы храните, годятся лишь для определенных «сборок» и «моделей» автомобилей; следовательно, в вашей базе данных есть одна таблица MAKE для хранения информации о сборке и другая таблица MODEL для хранения информации о модели. Поскольку эти две таблицы связаны (каждая модель должна принадлежать определенной сборке), реляционное ограничение может использоваться для обеспечения того, что каждая запись, хранящаяся в таблице MODEL, имеет соответствующую запись в таблице MAKE; отношение между этими двумя таблицами устанавливается путем сравнения значений, которые должны быть добавлены в столбец «MAKE» таблицы MODEL (известные как *внешний ключ (foreign key) дочерней таблицы (child table)*, со значениями, которые существуют в настоящее время для набора столбцов, составляющих первичный ключ таблицы MAKE (известные как *родительский ключ (parent key) родительской таблицы (parent table)*. Для создания только что описанного реляционного ограничения нужно было бы определить первичный ключ, используя один или несколько столбцов в таблице MAKE, и определить внешний ключ для одного или нескольких соответствующих столбцов в таблице MODEL, которая ссылается на первичный ключ таблицы MAKE. В предположении, что столбец MAKEID используется для создания первичного ключа для таблицы MAKE, а столбец, также названный MAKEID, используется для создания внешнего ключа для таблицы MODEL.

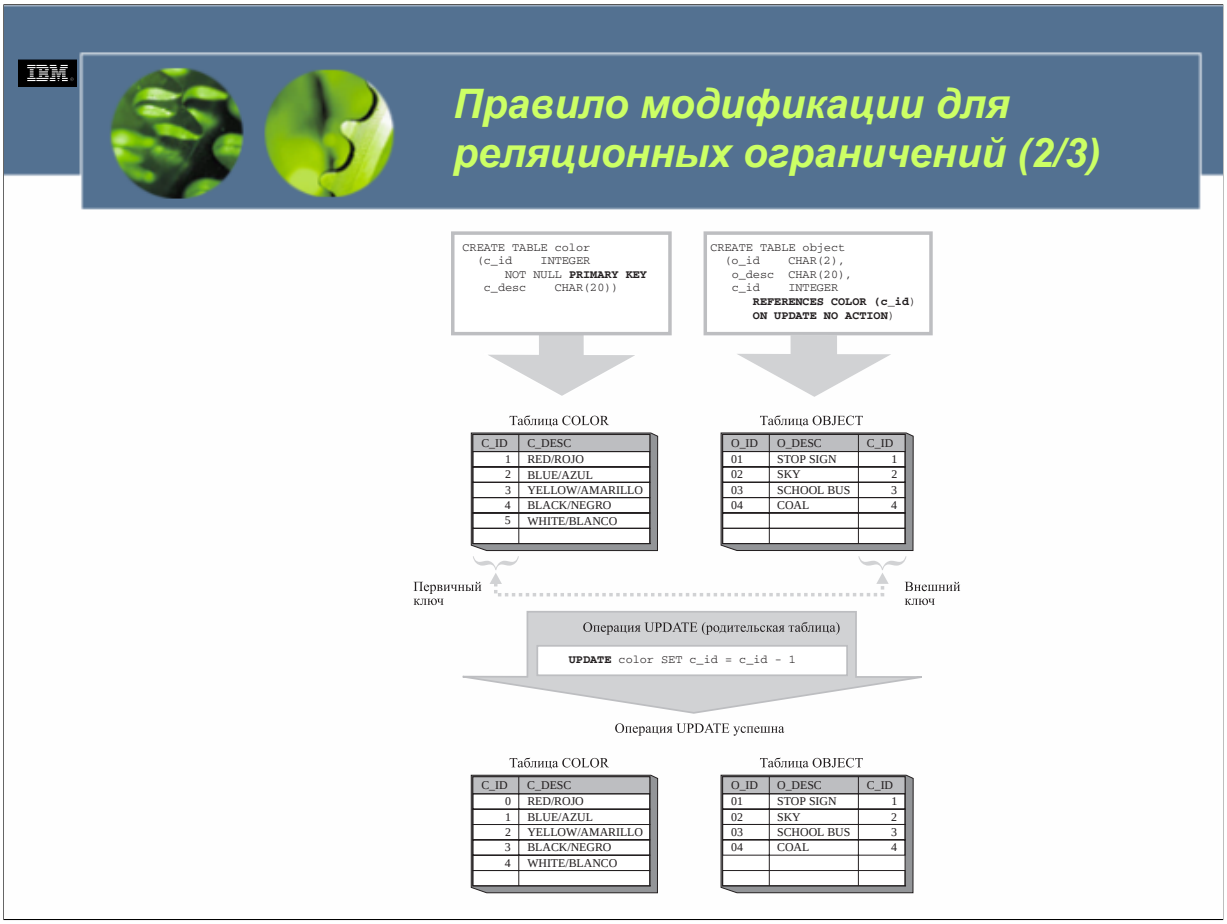
WCF03.0

Правило вставки гарантирует, что во внешний ключ дочерней таблицы никогда не будет введено значение, если оно не может быть найдено в соответствующем родительском ключе связанной родительской таблицы. Любая попытка ввести в дочернюю таблицу записи, нарушающие эти правила, приведут к ошибке, и операция вставки завершится неудачей. В отличие от этого, при добавлении записей к родительскому ключу родительской таблицы не выполняется никаких проверок.

WCF03.0

Важно отметить, что поскольку правило вставки существует, записи в родительский ключ родительской таблицы должны быть вставлены до того, как соответствующие записи могут быть вставлены в дочернюю таблицу. (Возвращаясь к нашему примеру MAKE/MODEL, это означает, что запись для нового MAKE должна быть добавлена в таблицу MAKE *до того*, как в таблицу MODEL может быть добавлена запись, ссылающаяся на новый MAKE.)

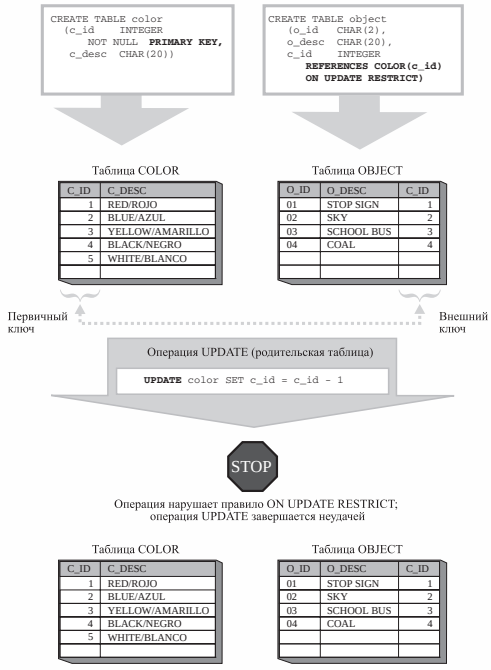




WCF03.0

Аналогично правилу вставки, правило модификации для реляционного ограничения неявно создается при создании самого реляционного ограничения. Если при определении реляционного ограничения определение правила модификации не представлено, по умолчанию используется определение ON UPDATE NO ACTION. Независимо от того, какая форма правила модификации используется, если условие правила не удовлетворяется, операция модификации завершается неудачей, отображается сообщение об ошибке, и будет сделан отказ от всех изменений, сделанных с данными в любой таблице, участвующей в реляционном ограничении.

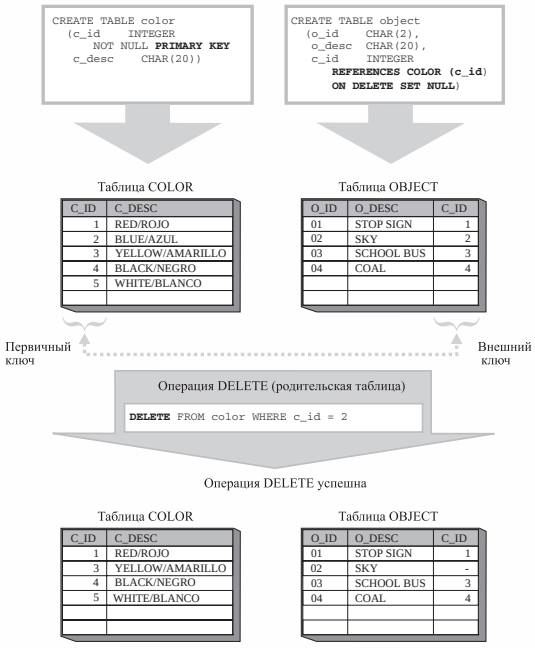
Правило модификации для реляционных ограничений (3/3)



WCF03.0

Если используется правило ON DELETE CASCADE, и удаление родительской строки в родительской таблице вызывает удаление одной или нескольких зависимых строк из соответствующей дочерней таблицы, говорят, что операция удаления *распространяется (propagated)* на дочернюю таблицу. В такой ситуации говорят, что дочерняя таблица *связана по удалению (delete-connected)* с родительской таблицей. Поскольку связанная по удалению дочерняя таблица также может быть родительской таблицей в другом реляционном ограничении, операция удаления, распространяющаяся на одну дочернюю таблицу, может в свою очередь распространиться на другую дочернюю таблицу и т. д. Таким образом, удаление одной родительской строки из одной родительской таблицы может привести к удалению нескольких сот строк из любого числа таблиц в зависимости от того, как связаны по удалению таблицы. Следовательно, правило удаления ON DELETE CASCADE следует использовать с чрезвычайной осторожностью, когда база данных пронизана иерархией реляционных ограничений.

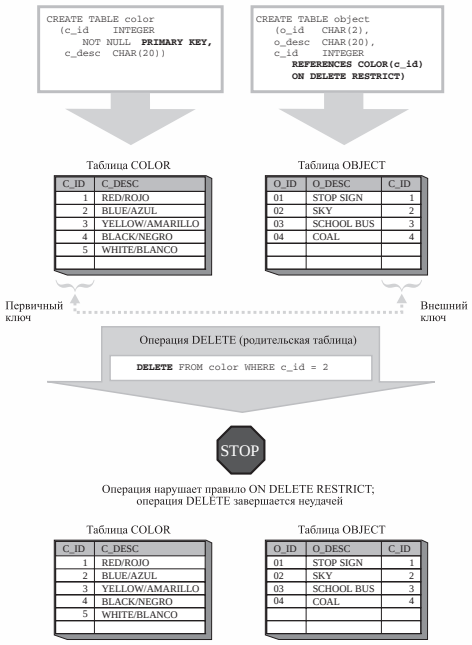
Правило удаления для реляционного ограничения (3/5)

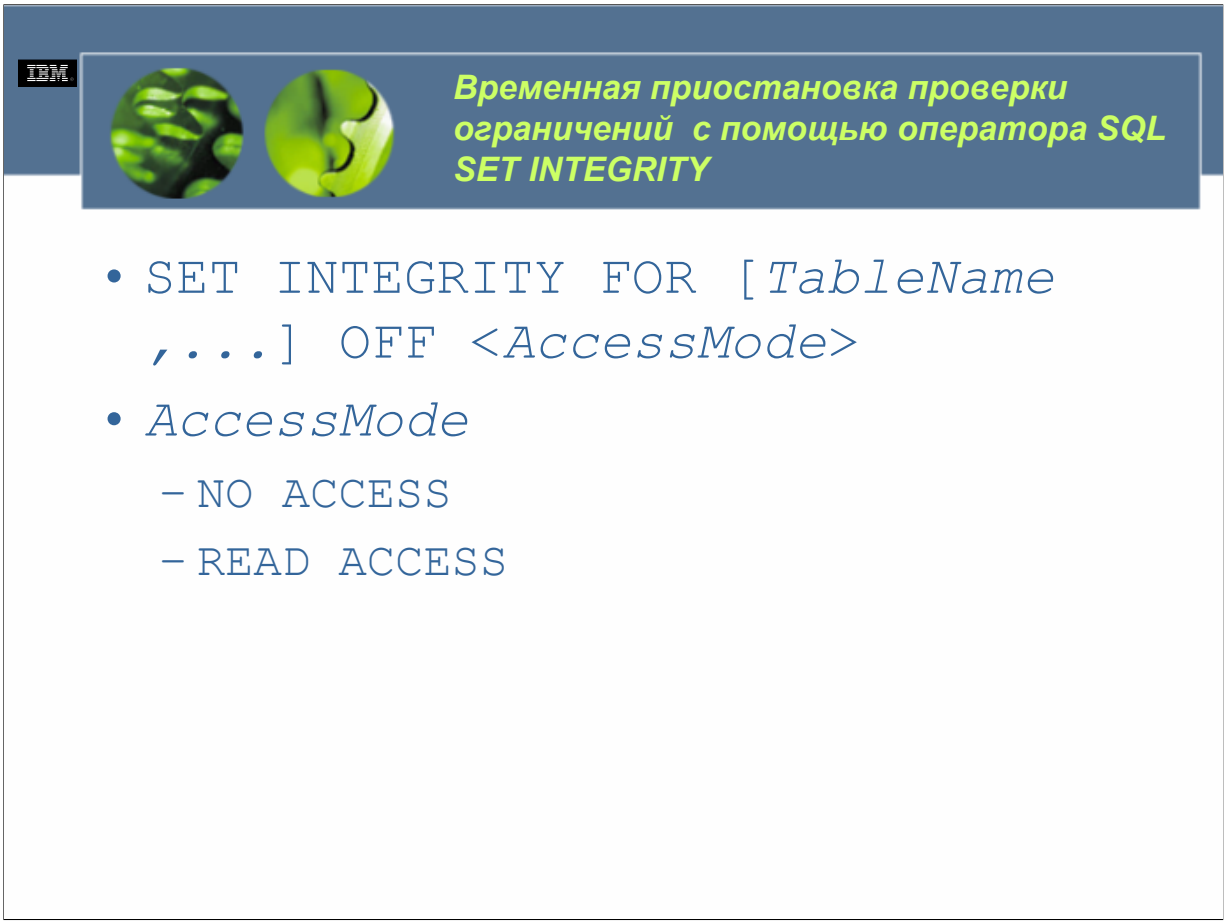


WCF03.0

Аналогично правилу вставки и правилу модификации, правило удаления для реляционного ограничения создается неявно при создании самого реляционного ограничения. Если правило удаления при определении реляционного ограничения не представлено, по умолчанию используется определение ON DELETE NO ACTION. Вне зависимости от того, какая форма правила удаления используется, если условие правила удаления не удовлетворяется, будет отображено сообщение об ошибке, а операция удаления завершится неудачей.

Правило удаления для реляционного ограничения (5/5)



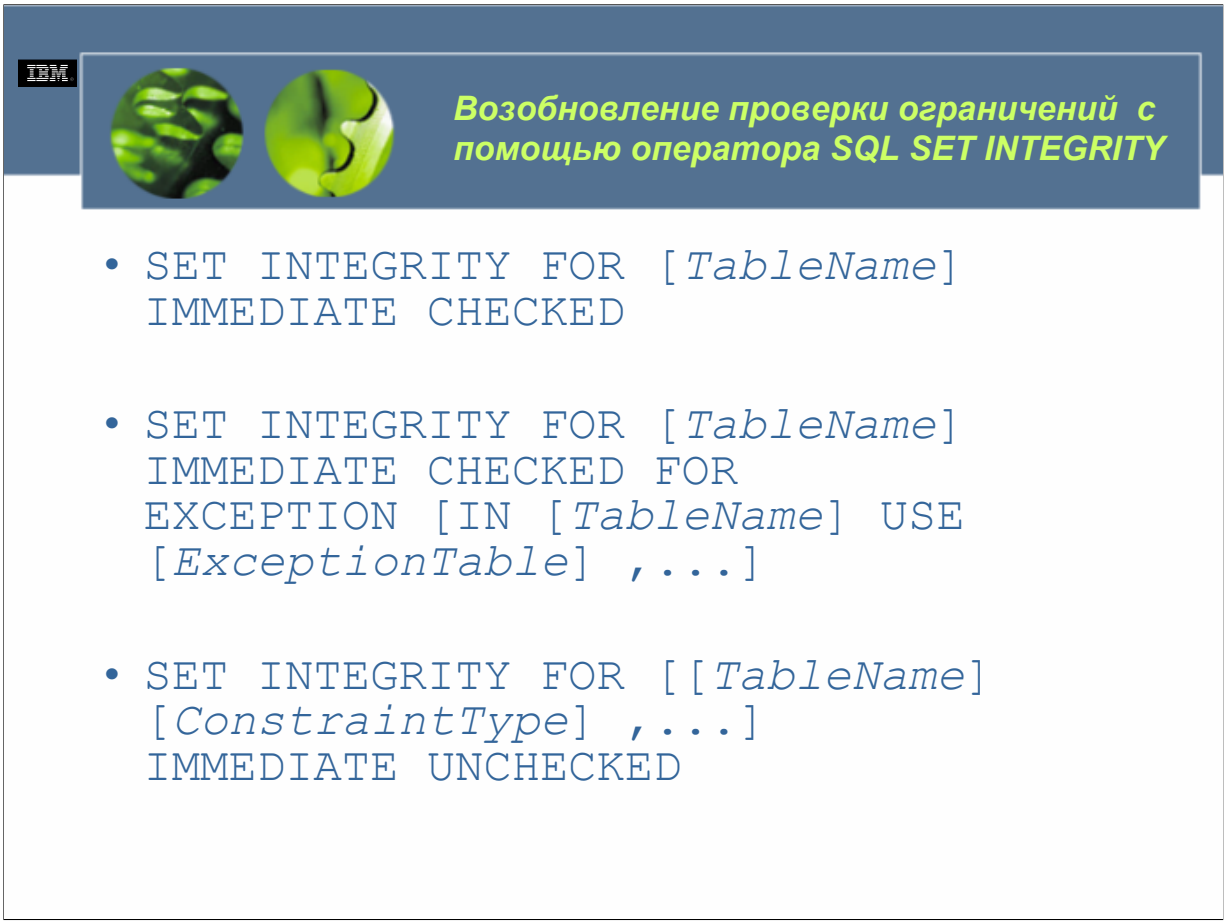


WCF03.0

Проверку ограничений для таблицы можно временно приостановить, выполнив оператор SQL SET INTEGRITY. При использовании для приостановки проверки ограничений синтаксис для простейшей формы этого оператора следующий.

SET INTEGRITY FOR [*TableName ,...*] OFF <*AccessMode*>

TableName определяет имя одной или нескольких базовых таблиц, для которых должна быть временно приостановлена проверка ограничений. *AccessMode* указывает, можно ли получить доступ только для чтения к указанным таблицам во время приостановки проверки ограничений. (Действительные значения включают в себя NO ACCESS и READ ACCESS – если режим доступа не указан, по умолчанию используется NO ACCESS.)



WCF03.0

Точно так же, как одна форма оператора SET INTEGRITY используется для временной приостановки проверки ограничений, другая форма используется для ее возобновления. В этом случае синтаксис для простейшей формы оператора SET INTEGRITY следующий.

SET INTEGRITY FOR [*TableName*] IMMEDIATE CHECKED FOR

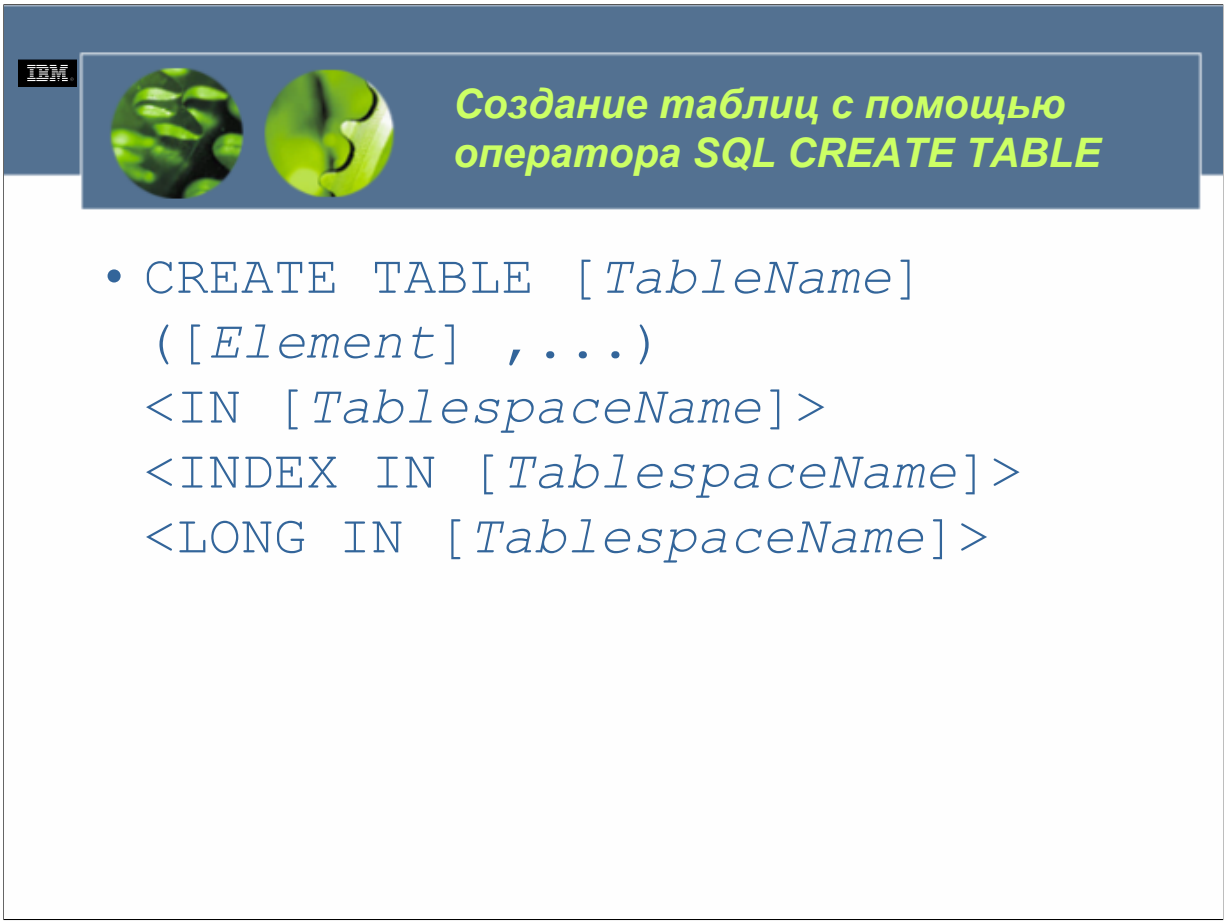
EXCEPTION [IN [*TableName*] USE [*ExceptionTable*] ,...]

или

SET INTEGRITY FOR [[*TableName*] [*ConstraintType*] ,...]

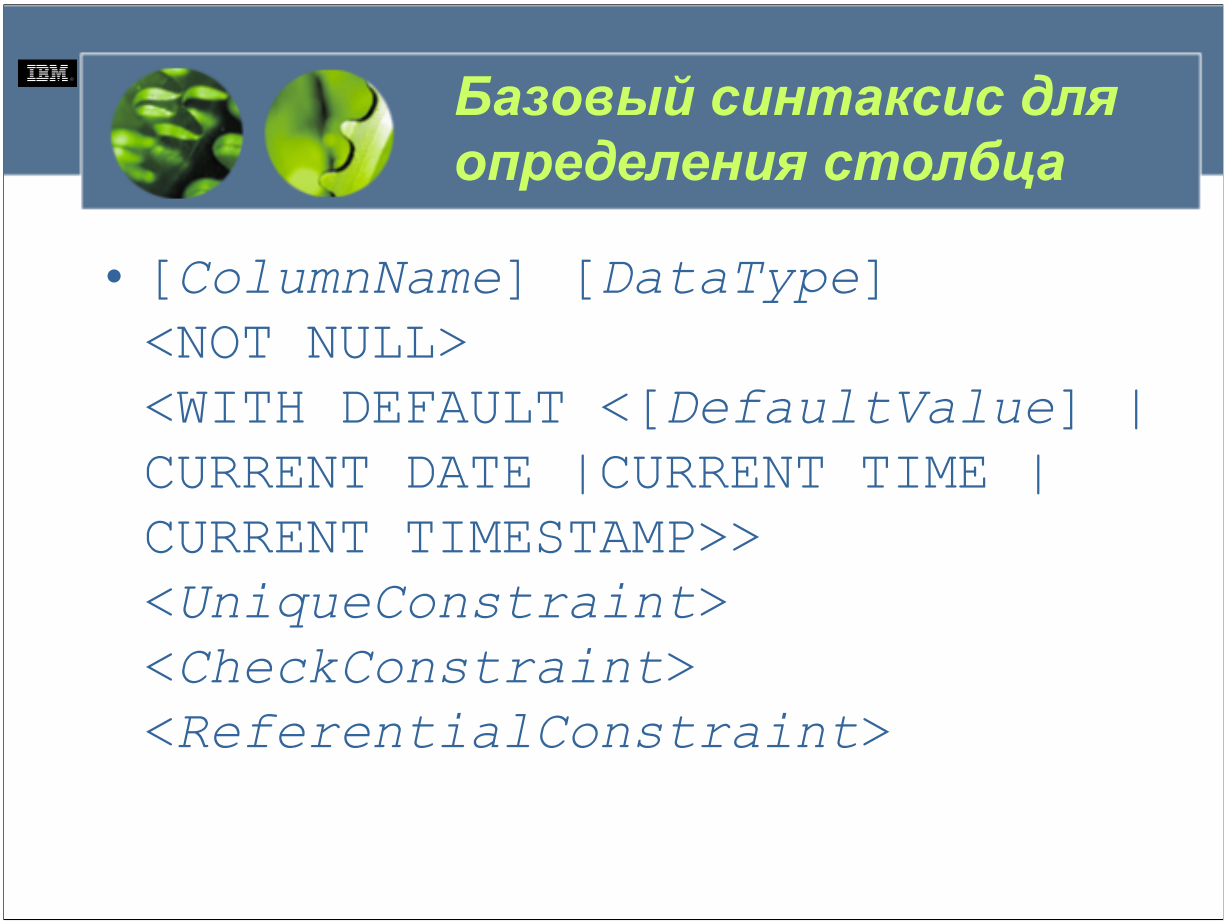
IMMEDIATE UNCHECKED

TableName определяет имя одной или нескольких базовых таблиц, для которых должна быть возобновлена проверка ограничений, а также одной или нескольких базовых таблиц, из которых должны быть скопированы все строки, нарушающие реляционное или проверочное ограничение.*ExceptionTable* определяет имя базовой таблицы, в которую должны быть скопированы все строки, нарушающие реляционное или проверочное ограничение. *ConstraintType* указывает тип ограничений, проверка которых должна быть возобновлена. (Действительные значения включают в себя FOREIGN KEY, CHECK, DATALINK RECONCILE PENDING, MATERILIZED QUERY, GENERATED COLUMN, STAGING и ALL.)

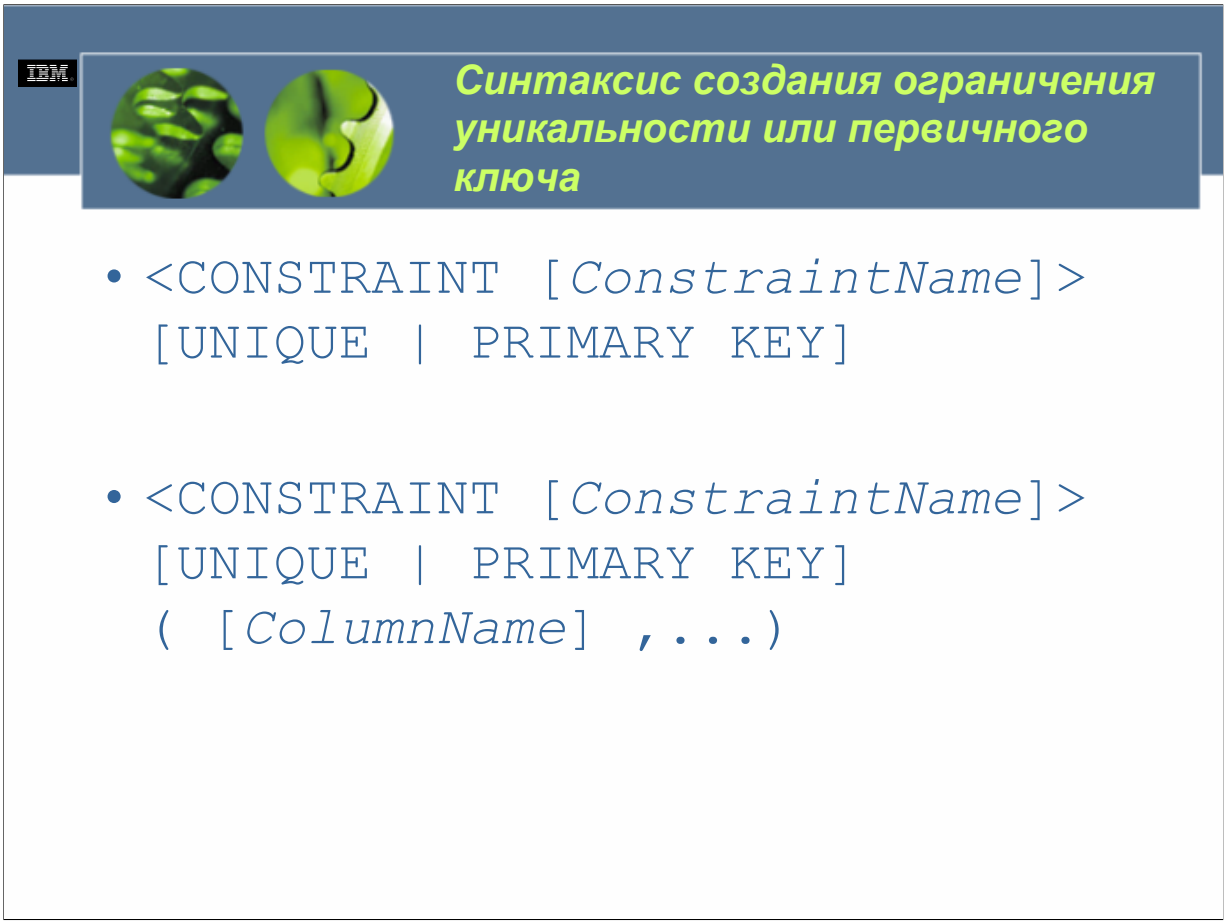


WCF03.0

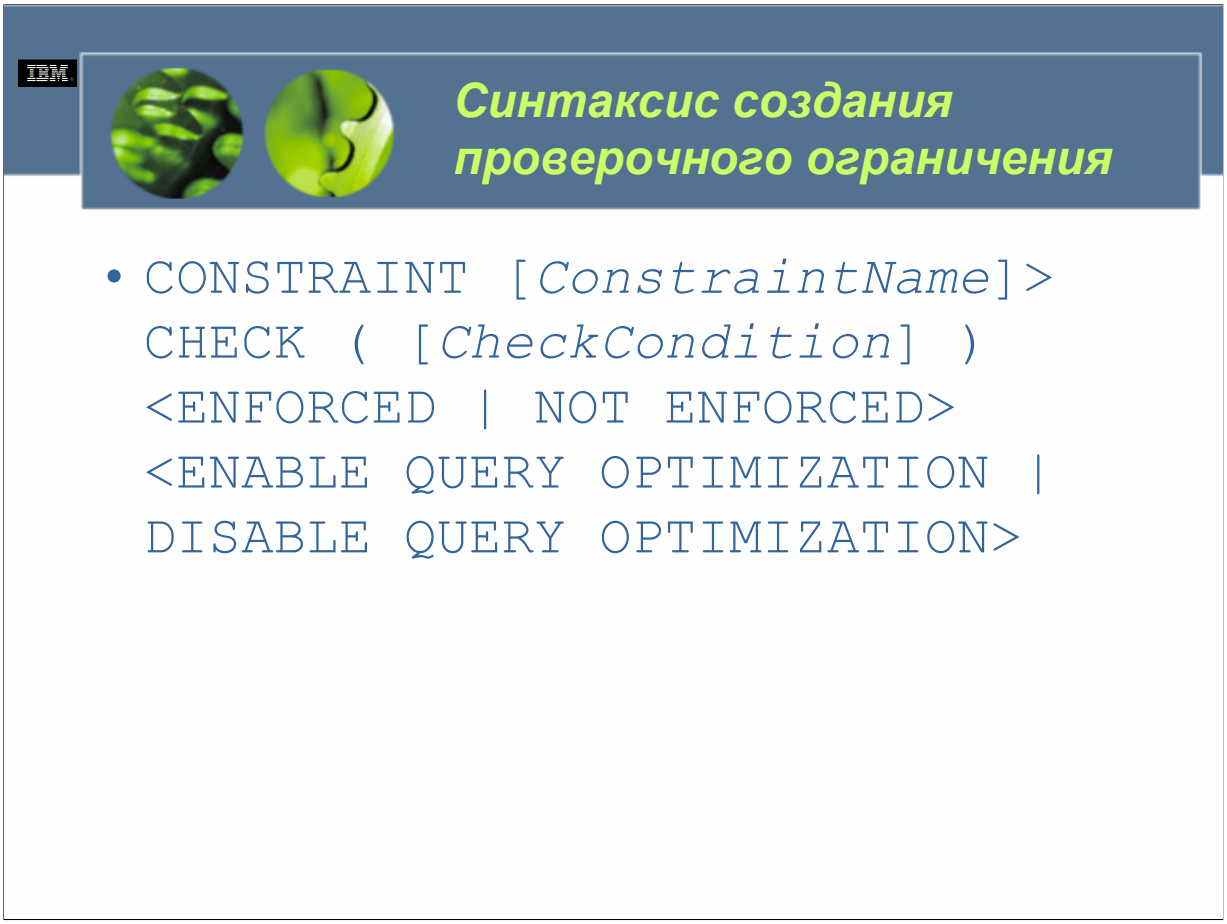
TableName определяет имя, которое должно быть присвоено создаваемой таблице. (Имя таблицы должно быть уникальным в пределах схемы, в которой должна быть создана таблица.) *Element* определяет один или несколько столбцов, ограничения уникальности/первичного ключа, реляционные ограничения и/или проверочные ограничения, которые должны быть включены в определение таблицы. Синтаксис, используемый для определения каждого из этих элементов, варьирует в зависимости от определяемого элемента. *TablespaceName* определяет табличное пространство, в котором должны храниться таблица и ее обычные данные, индексы и/или длинные данные/данные больших объектов. (Обычные данные, индексы и длинные данные/данные больших объектов могут храниться в разных табличных пространствах, только если используются табличные пространства DMS.)



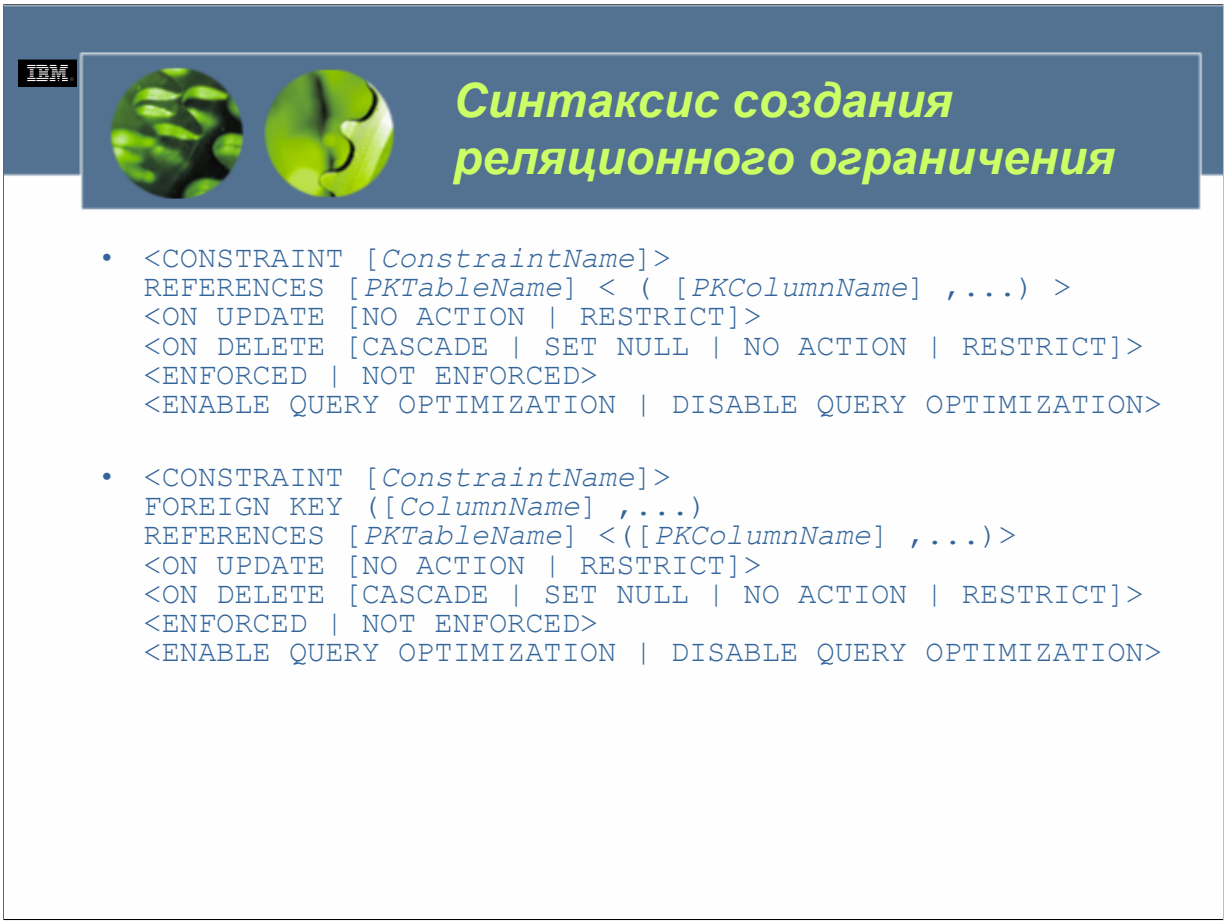
ColumnName определяет уникальное имя, которое должно быть присвоено создаваемому столбцу. *DataType* определяет тип данных (встроенный или пользовательский), который должен быть назначен создаваемому столбцу; указанный тип данных определяет разновидность значений данных, которые можно хранить в столбце. *DefaultValue* определяет значение, которое должно быть предусмотрено для столбца на случай, если для столбца не будет указано значения при выполнении операции вставки или изменения в таблице. *UniqueConstraint* определяет ограничение уникальности или первичного ключа, которое должно быть связано со столбцом. *CheckConstraint* определяет проверочное ограничение, которое должно быть связано со столбцом. *ReferentialConstraint* определяет реляционное ограничение, которое должно быть связано со столбцом.



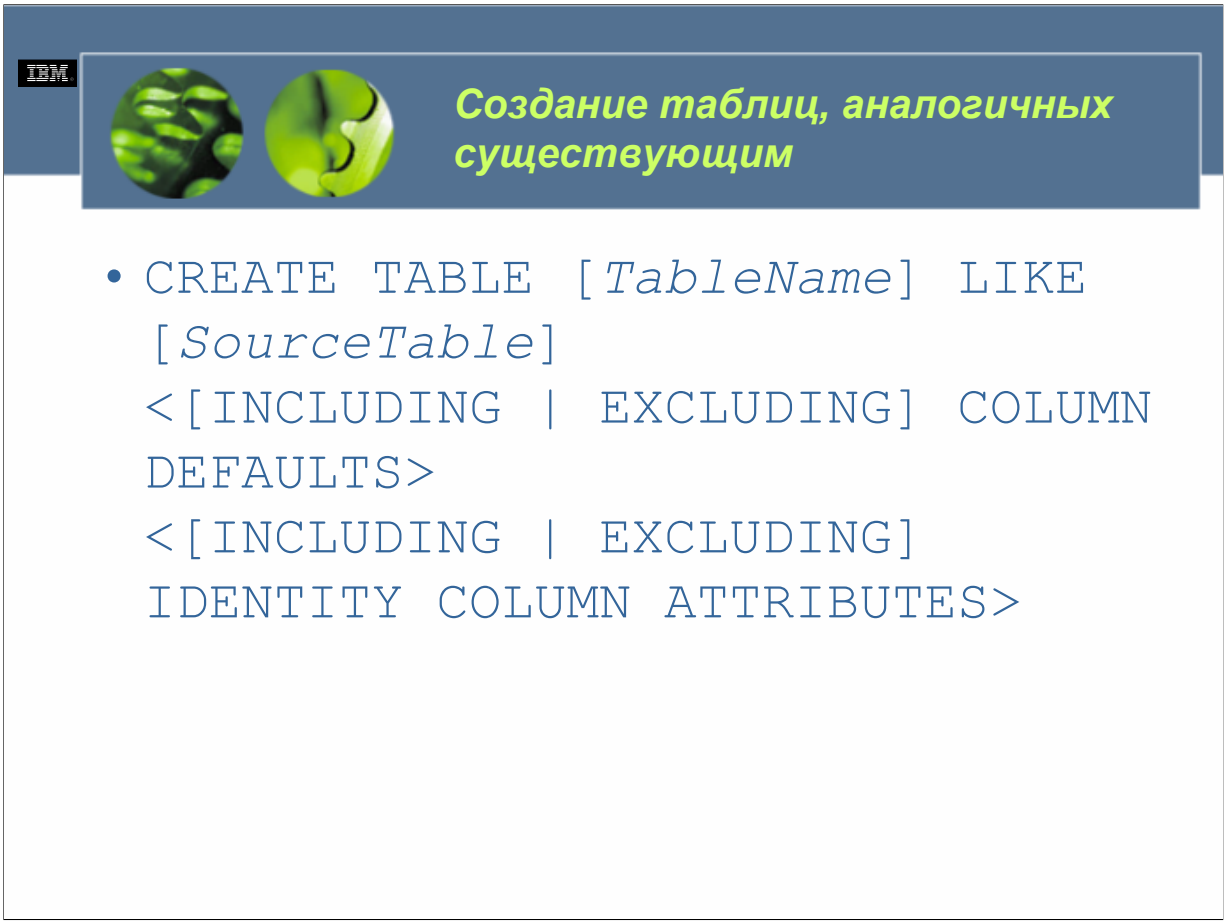
ConstraintName определяет уникальное имя, которое должно быть присвоено создаваемому ограничению. *ColumnName* определяет один или более столбцов, которые должны быть частью создаваемого ограничения уникальности или первичного ключа.



ConstraintName определяет уникальное имя, которое должно быть присвоено создаваемому ограничению. *CheckCondition* определяет условие или тест, который должен дать результат TRUE, прежде чем представленное для столбца значение действительно будет сохранено в таблице.



ConstraintName определяет уникальное имя, которое должно быть присвоено создаваемому ограничению.*PKTableName* определяет имя родительской таблицы, которая должна участвовать в реляционном ограничении. *PKColumnName* определяет столбец, составляющий родительский ключ родительской таблицы, который должен участвовать в реляционном ограничении. *ColumnName* определяет один или несколько столбцов, которые должны быть частью создаваемого реляционного ограничения.



WCF03.0

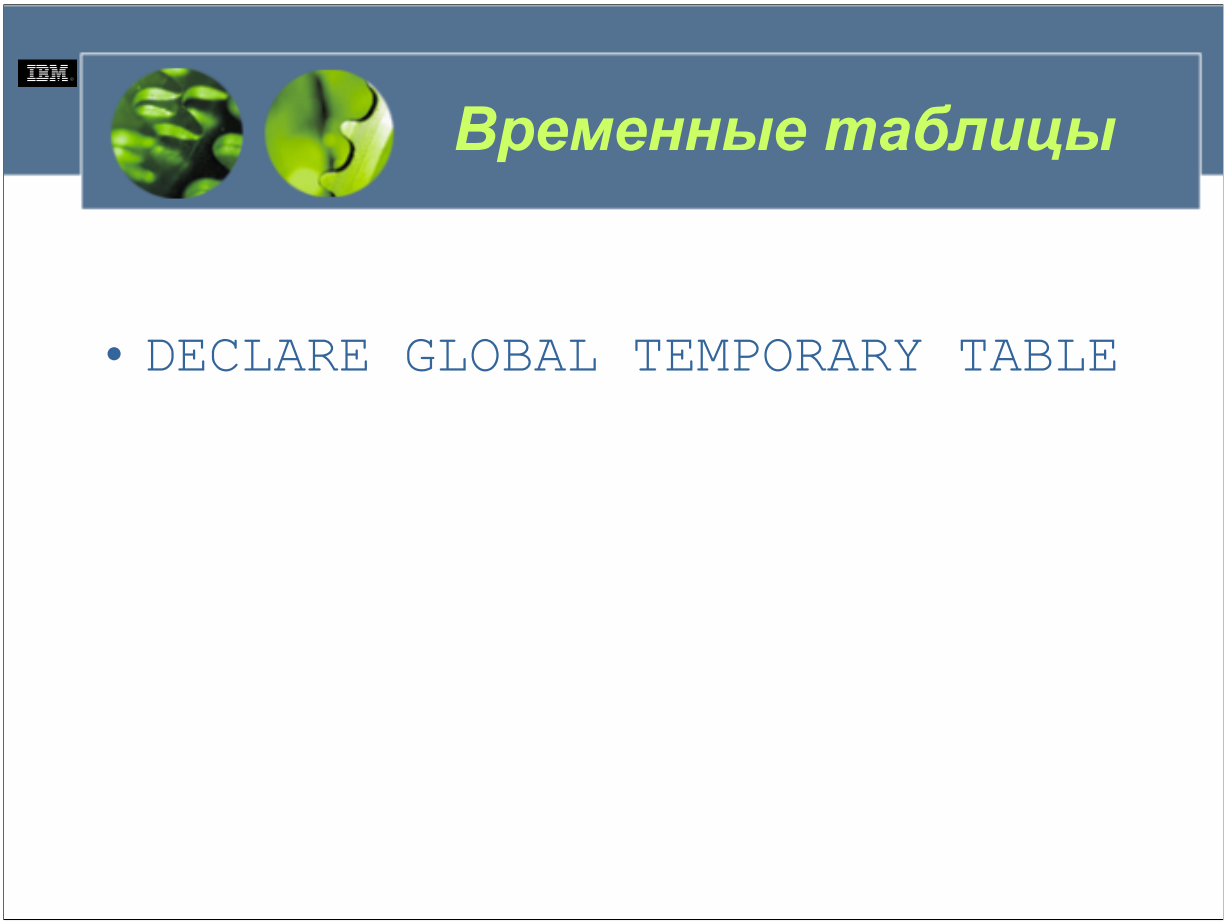
Время от времени может быть нужно создавать новую таблицу, имеющую то же самое определение, что и существующая таблица. Для выполнения такой операции вы могли бы выполнить оператор CREATE TABLE, который выглядит идентичным оператору CREATE TABLE, использованному для определения первоначальной таблицы. Или еще лучше, вы могли бы использовать специальную форму оператора CREATE TABLE. Синтаксис для этой формы CREATE TABLE следующий.

CREATE TABLE [*TableName*] LIKE [*SourceTable*]

<[INCLUDING | EXCLUDING] COLUMN DEFAULTS>

<[INCLUDING | EXCLUDING] IDENTITY COLUMN ATTRIBUTES>

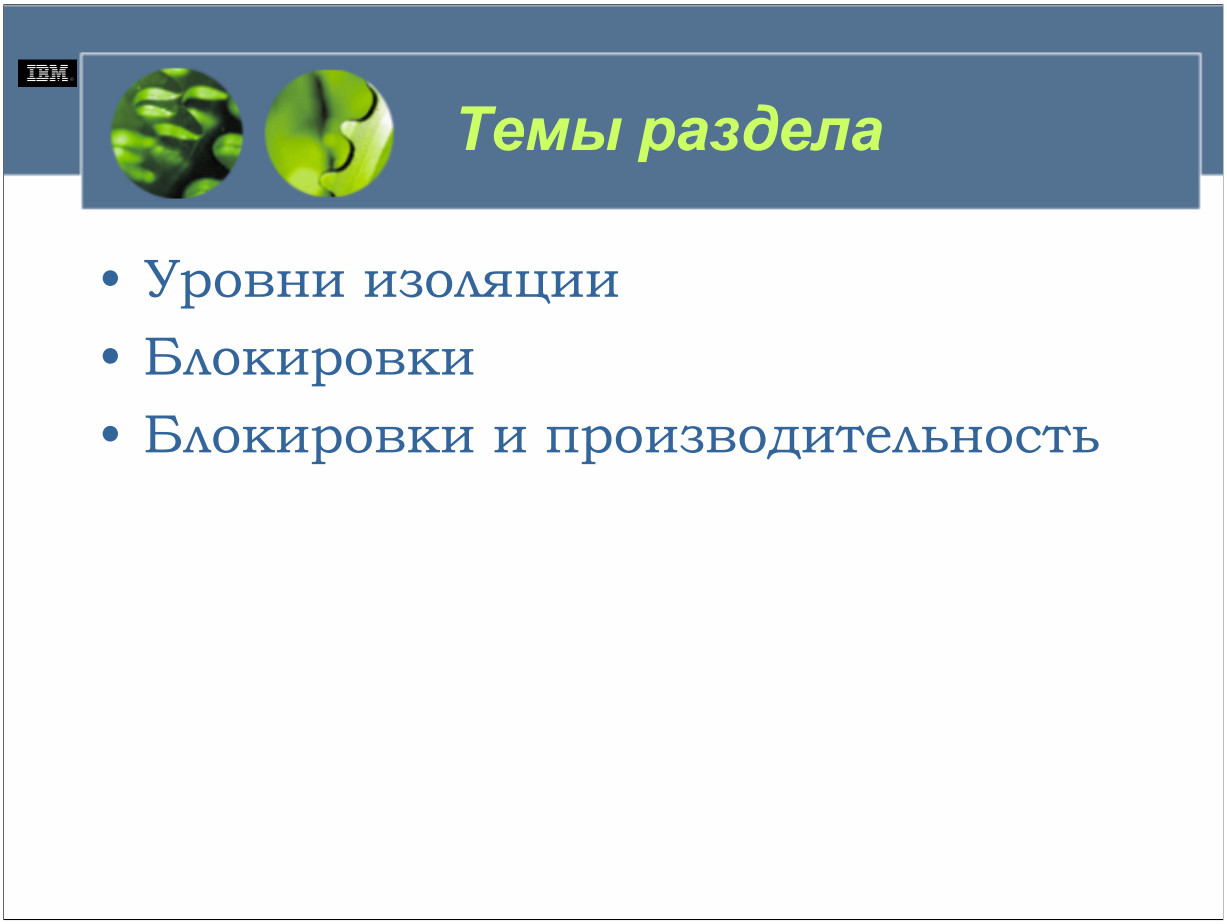
TableName определяет уникальное имя, которое должно быть присвоено создаваемой таблице. *SourceTable* определяет имя существующей таблицы, структура которой должна быть использована для определения создаваемой таблицы. Когда выполняется данная форма CREATE TABLE, таблица, которая в конечном счете создается, будет иметь то же число столбцов, что и указанная исходная таблица, и эти столбцы будут иметь те же самые имена, типы данных и свойства допустимости пустых значений, что и в исходной таблице. Кроме того, если не указана опция EXCLUDING COLUMN DEFAULTS, все ограничения по умолчанию, определенные для столбцов в исходной таблице, будут также скопированы и в новую таблицу. Однако другие атрибуты исходной таблицы дублированы не будут. Таким образом таблица, которая создается, не будет содержать ограничений уникальности, реляционных ограничений, триггеров или индексов, которые были определены для используемой исходной таблицы.

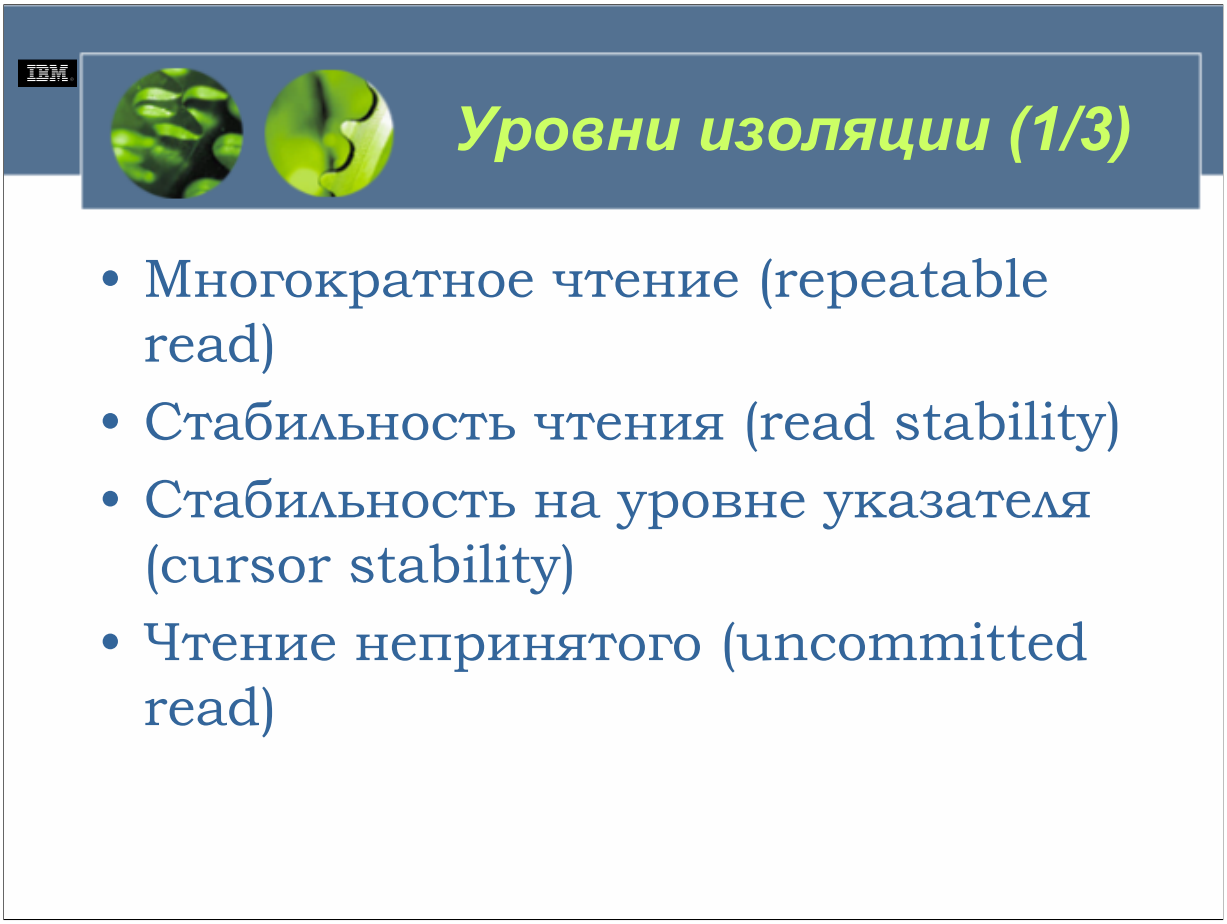


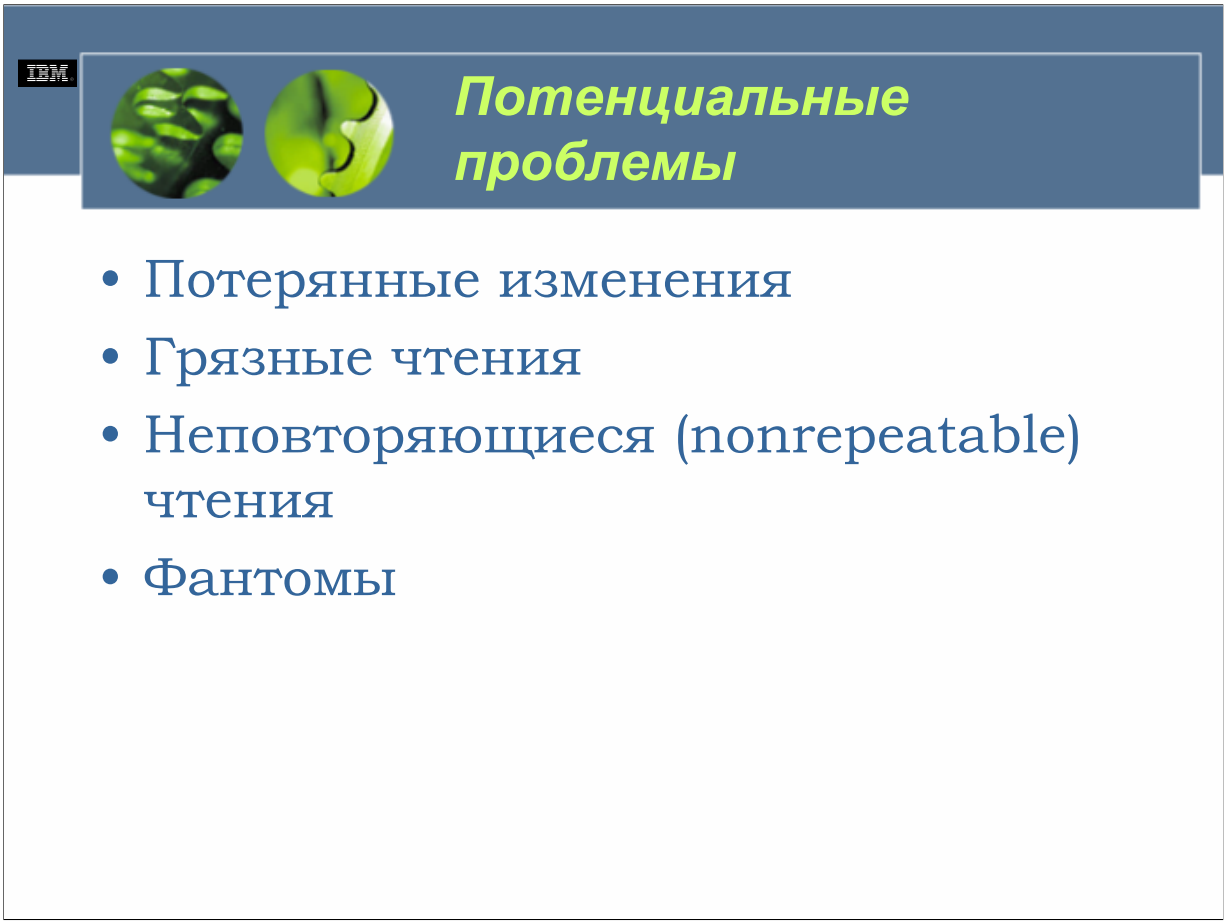
WCF03.0

Прежде чем мы взглянем на некоторые более сложные примеры оператора CREATE TABLE, следует упомянуть о другом виде обычно используемых таблиц. Этот вид таблиц известен, как объявленные временные таблицы (declared temporary table). В отличие от базовых таблиц, определения и ограничения которых хранятся в таблицах системного каталога той базы данных, к которой они принадлежат, объявленные временные таблицы не являются постоянными и могут использоваться лишь тем приложением, которое их создало, и лишь в течение времени жизни приложения. Когда приложение, создавшее объявленную временную таблицу, завершается, строки таблицы удаляются, а описание таблицы уничтожается. Тогда как базовые таблицы создаются оператором SQL CREATE TABLE, объявленные временные таблицы создаются оператором DECLARE GLOBAL TEMPORARY TABLE.

WCF03.0







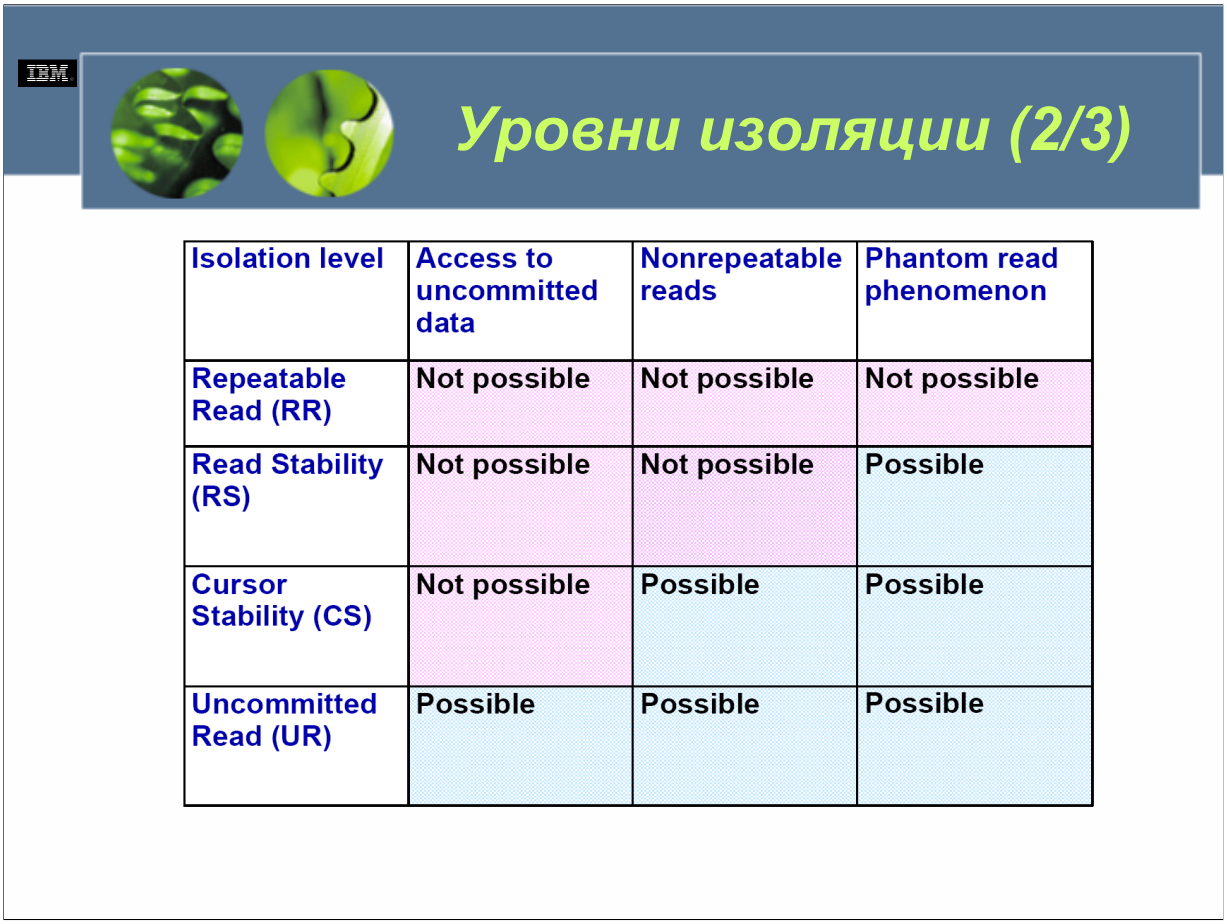
WCF03.0

Потерянные изменения. Это событие возникает, когда две транзакции читают одни и те же данные, обе пытаются изменить эти данные, и одно из изменений теряется. Например: транзакция A и транзакция B читают одну и ту же сроку данных и вычисляют новые значения для этой строки на основе первоначально считанных данных. Если транзакция A изменит значение в строке на новое, а затем транзакция B обновит ту же самую строку, операция изменения, выполненная транзакцией A, будет потеряна.

Грязные чтения. Это событие возникает, когда транзакция читает данные, которые еще не были подтверждены. Например: транзакция A изменяет строку данных, а транзакция B читает измененную строку прежде, чем транзакция подтвердит изменение. Если транзакция A сделает откат изменения, транзакция B окажется прочитавшей данные, которые теоретически никогда не существовали.

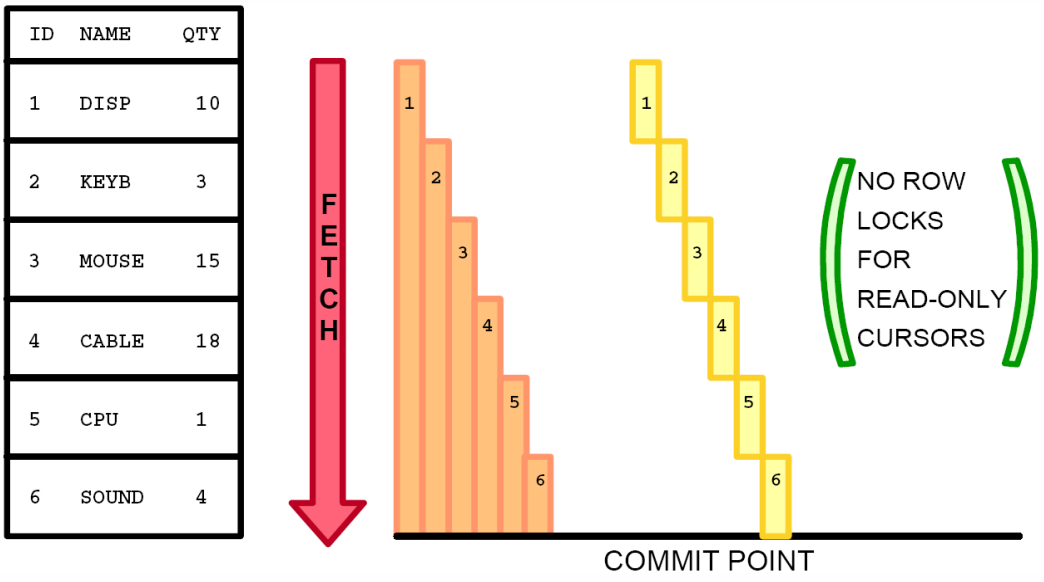
Неповторяющиеся (nonrepeatable) чтения. Это событие возникает, когда транзакция читает одну и ту же строку данных дважды, но каждый раз получает различные результаты. Например: транзакция A читает строку данных, а затем транзакция B изменяет или удаляет эту строку и подтверждает изменение. Когда транзакция A пытается повторно прочесть строку, она получит другие значения данных (если строка была изменена) или обнаружит, что строка больше не существует (если строка была удалена).

Фантомы. Это событие возникает, когда строка данных удовлетворяет некоторым критериям поиска, но первоначально не видна. Например: транзакция A получает набор строк, которые удовлетворяют некоторому критерию поиска, а затем транзакция B вставляет новую строку, которая удовлетворяет критерию поиска для запроса транзакции A. Если транзакция A повторно выполнит запрос, создавший первоначальный набор строк, будет получен другой набор строк – теперь в набор возвращенных строк будет включена строка, добавленная транзакцией B.

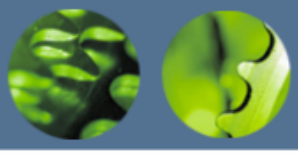


Isolation level	Access to uncommitted data	Nonrepeatable reads	Phantom read phenomenon
Repeatable Read (RR)	Not possible	Not possible	Not possible
Read Stability (RS)	Not possible	Not possible	Possible
Cursor Stability (CS)	Not possible	Possible	Possible
Uncommitted Read (UR)	Possible	Possible	Possible

Уровни изоляции (3/3)

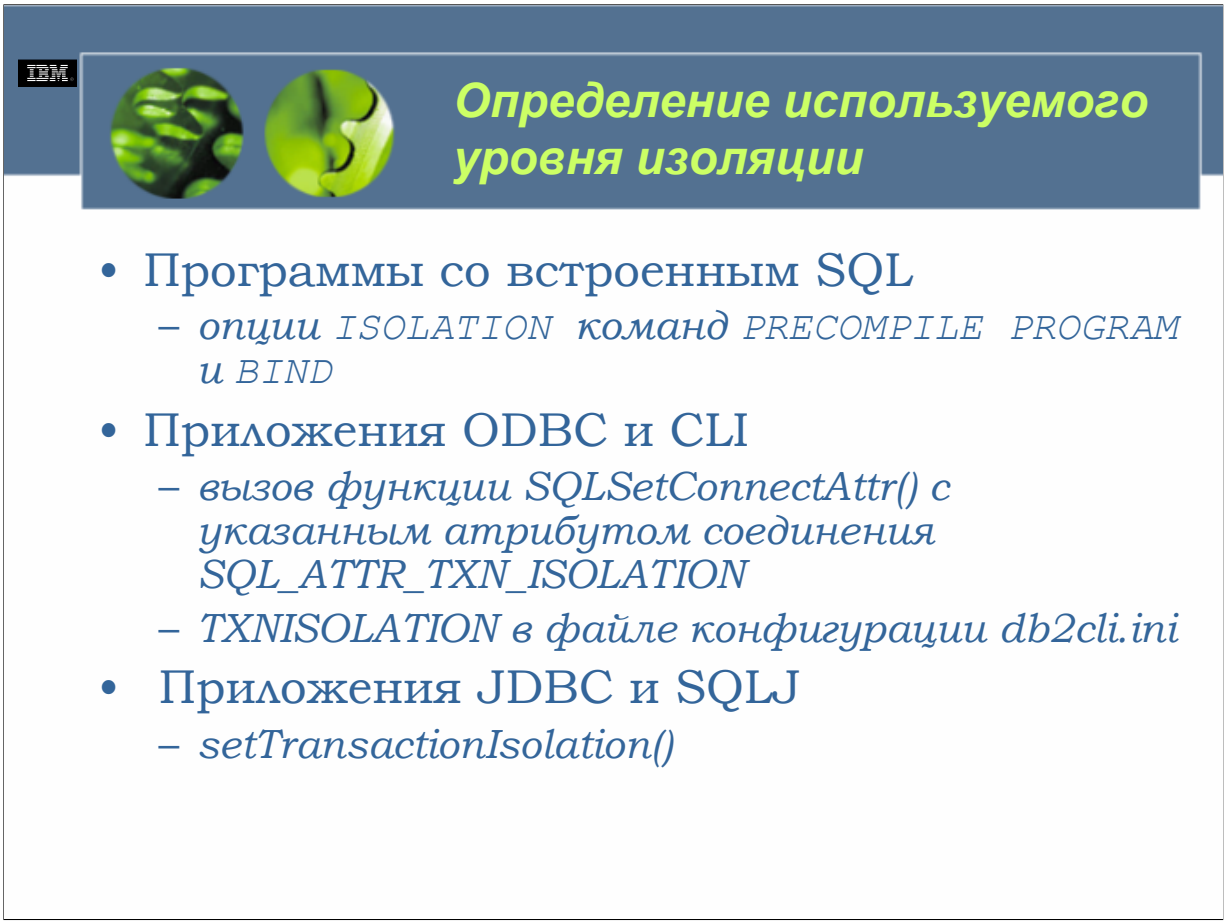


WCF03.0



Выбор подходящего уровня изоляции

- Использовать уровень изоляции многократных чтений, если вы выполняете запросы и не хотите, чтобы другие одновременные транзакции имели возможность делать изменения, которые могли бы вызвать при неоднократном выполнении возвращение одним и тем же запросом различных результатов.
- Использовать уровень изоляции стабильности чтения, когда вы хотите иметь между приложениями некоторый уровень одновременности и в то же время хотите, чтобы ограниченный набор строк оставался неизменным на протяжении отдельной транзакции.
- Использовать уровень изоляции стабильности на уровне указателя, когда вы хотите максимальную одновременность между приложениями и в то же время не хотите, чтобы запросы возвращали непринятые значения данных.
- Использовать уровень изоляции чтения непринятого, если вы выполняете запросы в базах данных, используемых только для чтения, или если вы не беспокоитесь о том, возвращает ли запрос непринятые значения данных.

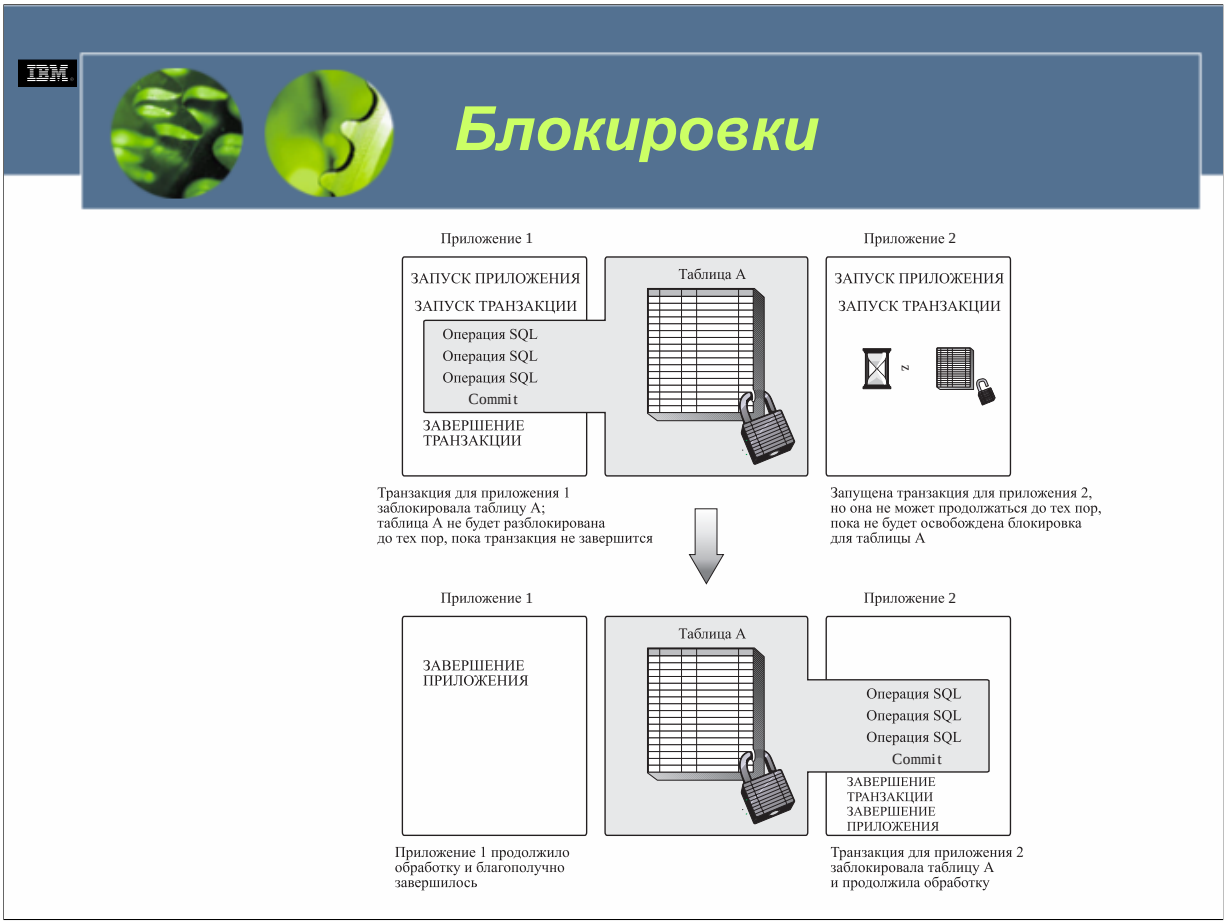


WCF03.0

Хотя уровни изоляции контролируют одновременность на уровне транзакций, в действительности они устанавливаются на уровне приложений. Поэтому в большинстве случаев указанный для определенного приложения уровень изоляции применяется к каждой транзакции, инициированной этим приложением. (Важно отметить, что приложение может быть составлено из нескольких отдельных частей, и каждой части может быть назначен другой уровень изоляции, в этом случае к каждой транзакции, созданной в этой части, применяется уровень изоляции, определенный для этой отдельной части.)

Для приложений встроенного SQL используемый уровень изоляции определяется во время прекомпилирования или при связывании приложения с базой данных (если используется отложенное связывание). Уровень изоляции для приложений встроенного SQL, написанных на поддерживаемом компилируемом языке (таком, как C и C++), устанавливается посредством опции ISOLATION команд PRECOMPILE PROGRAM и BIND. Уровень изоляции для приложений ODBC и CLI устанавливается во время выполнения приложения путем вызова функции SQLSetConnectAttr() с указанным атрибутом соединения SQL_ATTR_TXN_ISOLATION. В качестве альтернативы уровень изоляции для приложений ODBC/CLI может быть установлен путем присваивания значения ключевому слову TXNISOLATION в файле конфигурации *db2cli.ini*; однако этот подход не обеспечивает гибкость изменения уровней изоляции для различных транзакций внутри приложений, который обеспечивает первый подход. Уровень изоляции для приложений JDBC и SQLJ устанавливается во время выполнения приложения путем вызова метода setTransactionIsolation(), который находится в интерфейсе соединения универсальной базы данных DB2 *java.sql*.

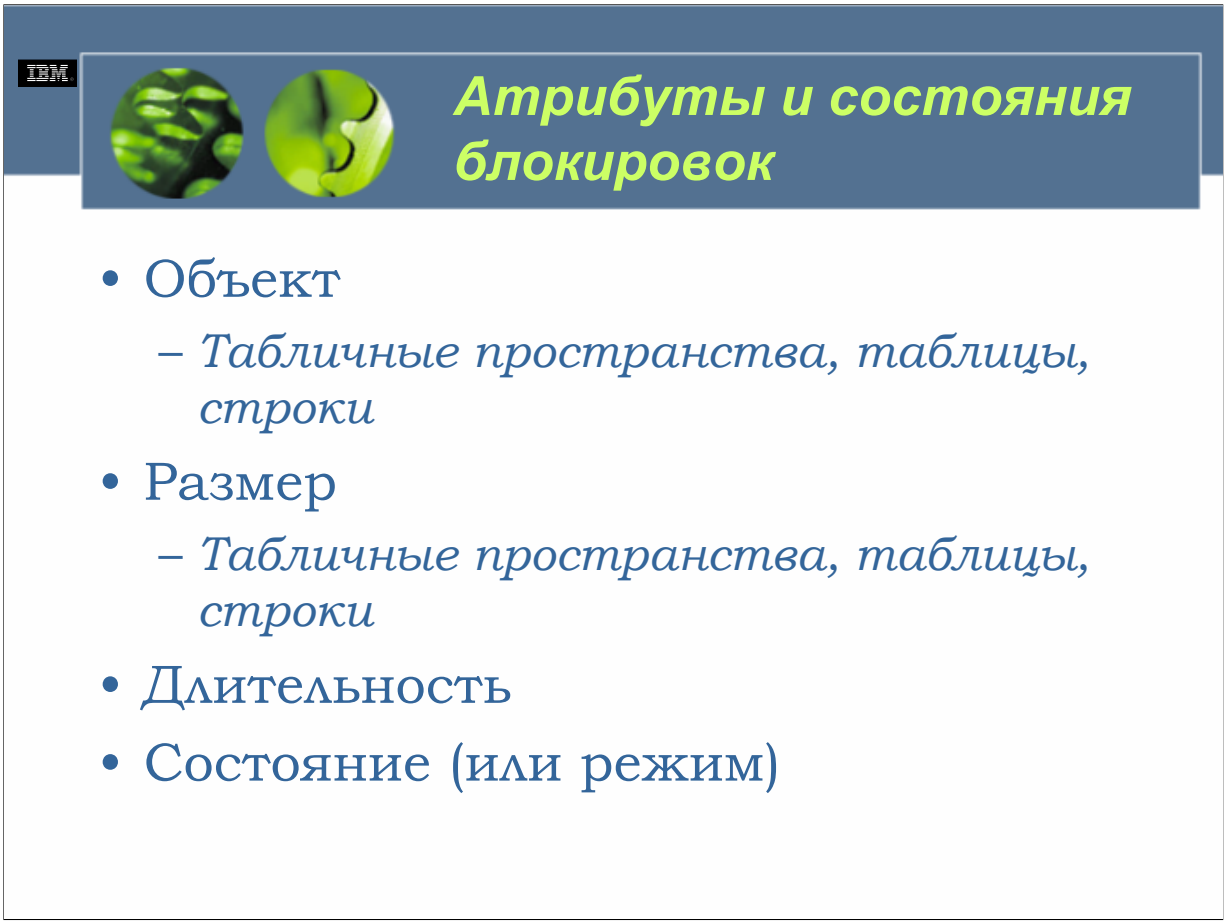
Когда уровень изоляции для приложения не установлен явным образом с использованием одного из этих методов, по умолчанию используется уровень изоляции стабильности на уровне указателя. Это справедливо для команд, операторов SQL и сценариев, выполняемых из процессора командной строки, а также для приложений встроенного SQL, ODBC/CLI, JDBC и SQLJ. Следовательно, можно также определить уровень изоляции для использования с любой транзакцией, которая должна быть выполнена процессором командной строки. В этом случае уровень изоляции устанавливается путем выполнения команды CHANGE ISOLATION перед установлением соединения с базой данных.



WCF03.0

Одно общее, что есть у каждого из четырех уровней изоляции, это то, что они контролируют доступ к данным одновременных транзакций посредством использования блокировок. Так что же такое блокировка? *Блокировка* является механизмом, использующимся для связывания ресурса данных с одной транзакцией, с единственной целью контролирования того, как другие транзакции взаимодействуют с этим ресурсом, когда к нему осуществляет доступ транзакция, которая его заблокировала. (Говорят, что транзакция, связанная с ресурсом данных, «удерживает» или «владеет» блокировкой.) В сущности блокировки («замки»[1]) в среде баз данных служат тем же самым целям, что и замки в домах и автомобилях: они определяют, кто может, а кто не может получить доступ к определенному ресурсу – в случае ресурса данных это одно или несколько табличных пространств, таблиц и/или строк. Менеджер баз данных DB2 налагает блокировки, чтобы помешать «владеющим» транзакциям получить доступ к не принятым данным, которые были записаны другими транзакциями, и чтобы не допустить изменений данных другими транзакциями, которые могли бы неблагоприятно повлиять на владеющую транзакцию. Когда владеющая транзакция завершается (путем принятия или отката), все изменения, сделанные для заблокированного ресурса, делаются постоянными или удаляются, а все блокировки для ресурса, которые были получены владеющей транзакцией, освобождаются. После разблокирования ресурс может быть заблокирован снова и изменен другой активной транзакцией.

[1] Lock (англ.) – блокировка, замок – *Примеч. пер.*



WCF03.0

У всех блокировок DB2 Universal Database есть следующие основные атрибуты.

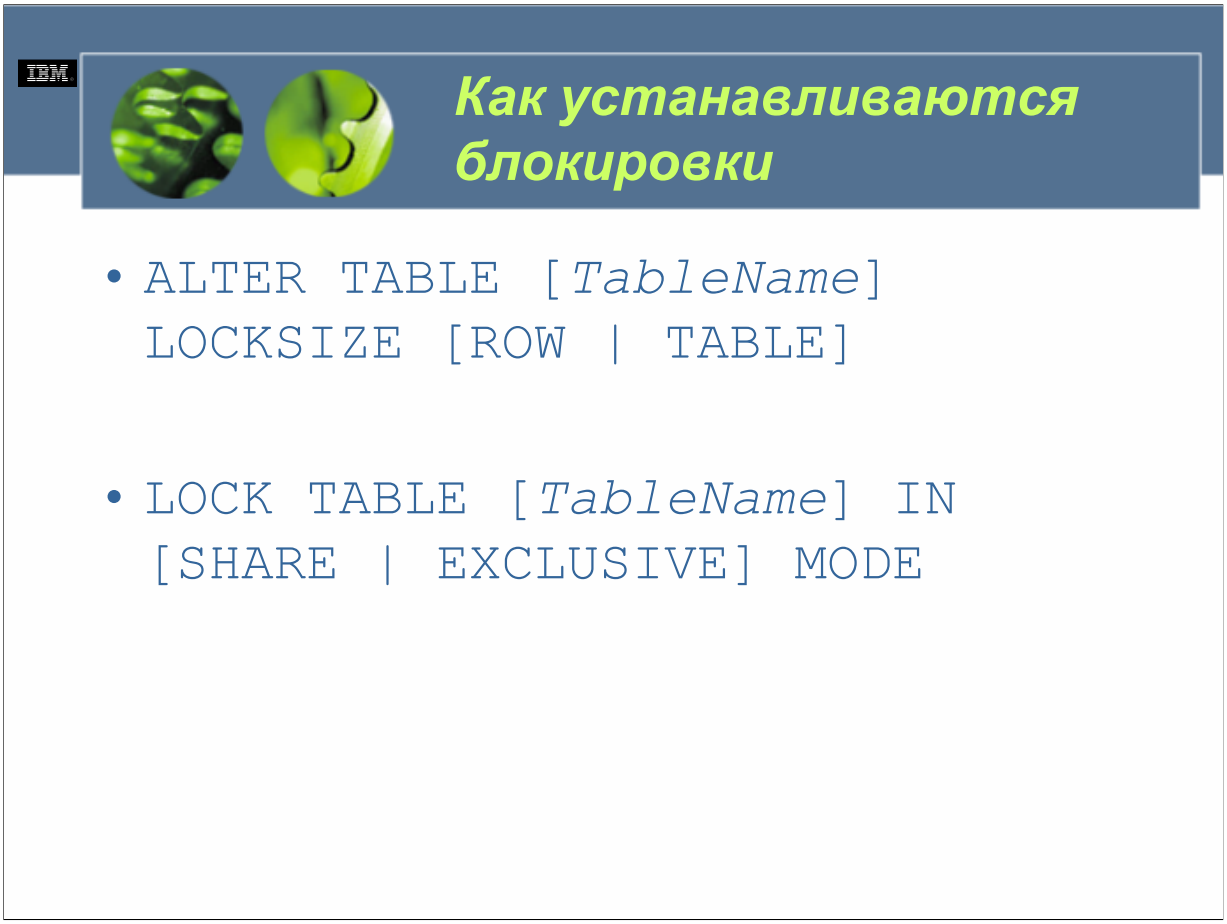
Объект. Этот атрибут идентифицирует ресурс данных, который блокируется. Менеджер баз данных DB2 неявно устанавливает блокировки для ресурсов данных (а именно, для табличных пространств, таблиц и строк) каждый раз, когда они нужны.

Размер. Этот атрибут определяет физический размер части ресурса данных, которая блокируется. Блокировка не всегда должна контролировать весь ресурс данных. Например, вместо того чтобы предоставить приложению исключительный контроль над всей таблицей, менеджер баз данных DB2 может решить дать приложению исключительный контроль над одной или несколькими определенными строками в таблице.

Длительность. Этот атрибут определят продолжительность времени, в течении которого удерживается блокировка. Использованный уровень изоляции имеет значительное влияние на длительность блокировки. (Например, блокировка, установленная для транзакции многократного чтения, которая получает доступ к 500 строкам, вероятно, будет иметь большую длительность, если должны быть изменены все 500 строк; с другой стороны, блокировка, полученная для транзакции стабильности на уровне указателя, вероятно, будет иметь значительно меньшую длительность).

Состояние (или режим). Этот атрибут определяет вид доступа, разрешенного для владельца блокировки, а также вид доступа, разрешенного для одновременно работающих пользователей для заблокированного ресурса данных.

WCF03.0



WCF03.0

За исключением случаев, когда используется уровень изоляции чтения непринятого, транзакция никогда не должна запрашивать блокировку явным образом. Это происходит вследствие того, что менеджер баз данных DB2 устанавливает блокировки неявным образом, когда они нужны (и после получения эти блокировки остаются под контролем менеджера баз данных DB2 до тех пор, пока они будут больше не нужны). По умолчанию менеджер баз данных DB2 всегда пытается установить блокировки на уровне строк. Однако можно контролировать то, должен ли менеджер баз данных DB2 всегда устанавливать блокировки на уровне строк или на уровне таблиц для определенного ресурса таблицы, выполнив специальную форму оператора SQL ALTER TABLE. Синтаксис данной формы оператора ALTER TABLE следующий.

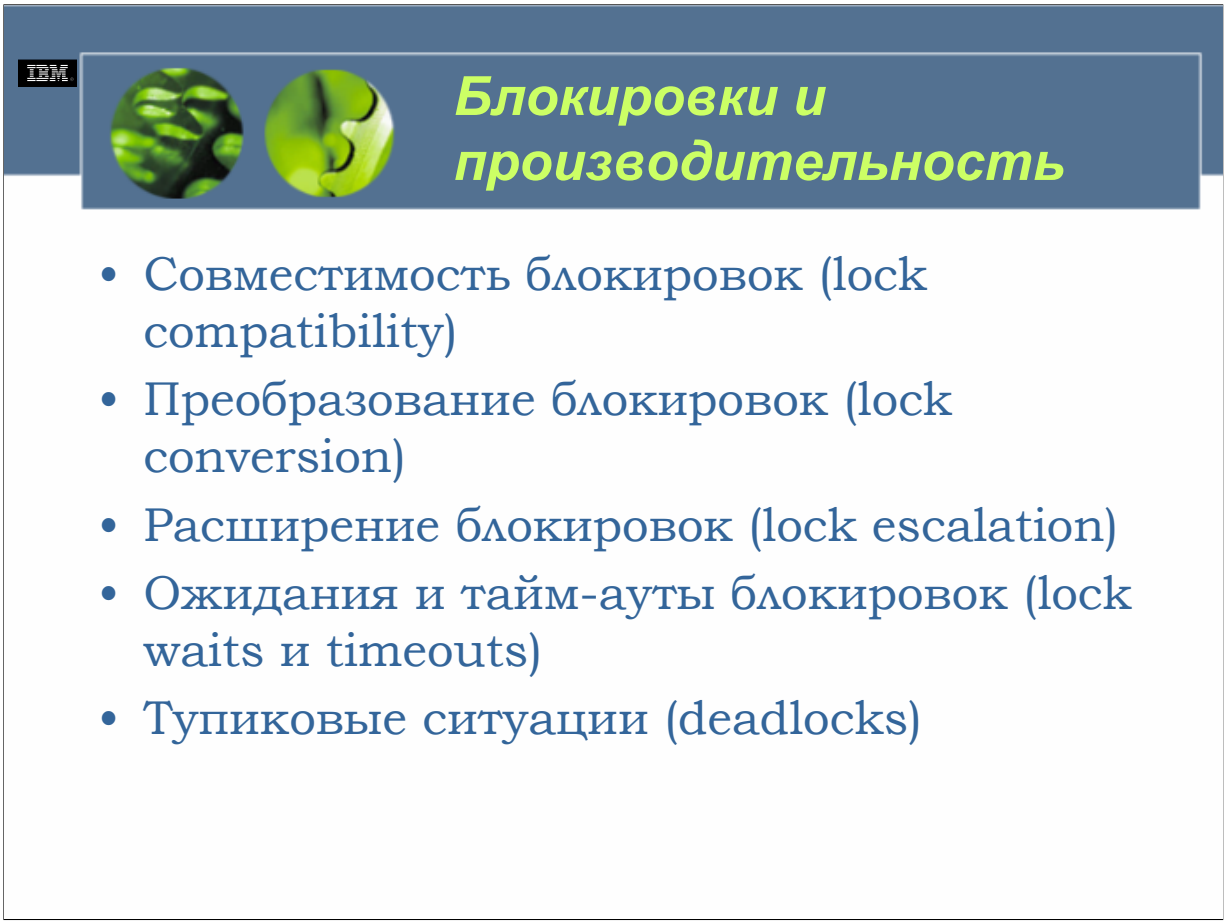
ALTER TABLE [*TableName*] LOCKSIZE [ROW | TABLE]

TableName определяет имя существующей таблицы, для которой должен быть указан уровень блокировки, который используют все транзакции при доступе к ней.

Но что, если вы не хотите, чтобы каждая транзакция, которая работает с определенной таблицей, получала блокировки на уровне таблицы? Что, если вы вместо этого хотите, чтобы только одна определенная транзакция получила блокировки на уровне таблицы, а все другие транзакции получали блокировки на уровне строк, работая с этой таблицей? В таком случае вы не трогаете поведение блокировки по умолчанию (когда используется блокировка на уровне строк), а для получения блокировки на уровне таблицы используете оператор SQL LOCK TABLE для соответствующей конкретной транзакции. Синтаксис для оператора LOCK TABLE следующий.

LOCK TABLE [*TableName*] IN [SHARE | EXCLUSIVE] MODE

TableName определяет имя существующей таблицы, которая должна быть заблокирована. Как вы можете видеть, оператор LOCK TABLE позволяет получить транзакции блокировку на уровне таблиц для определенной таблицы в одном из двух режимов: режиме SHARE и режиме EXCLUSIVE. Если таблица заблокирована с использованием режима SHARE, от имени запрашивающей транзакции устанавливается блокировка Share (S) на уровне таблицы, а другим одновременным транзакциям разрешается читать, но не изменять данные, хранящиеся в заблокированной таблице. С другой стороны. если таблица заблокирована с использованием режима EXCLUSIVE, устанавливается блокировка Exclusive (X) на уровне таблицы, а другие одновременные транзакции не могут ни получить доступ, ни изменить данные, хранящиеся в заблокированной таблице.



WCF03.0

Хотя менеджер баз данных DB2 устанавливает блокировки неявно, когда они необходимы, и, за исключением использования операторов SQL ALTER TABLE и LOCK TABLE для того, чтобы заставить менеджер баз данных DB2 установить блокировки на уровне таблиц, блокировки находятся вне вашего контроля, имеется несколько факторов, которые могут повлиять на блокировки, о которых следует знать. Эти факторы включают в себя.

совместимость блокировок (lock compatibility);

преобразование блокировок (lock conversion);

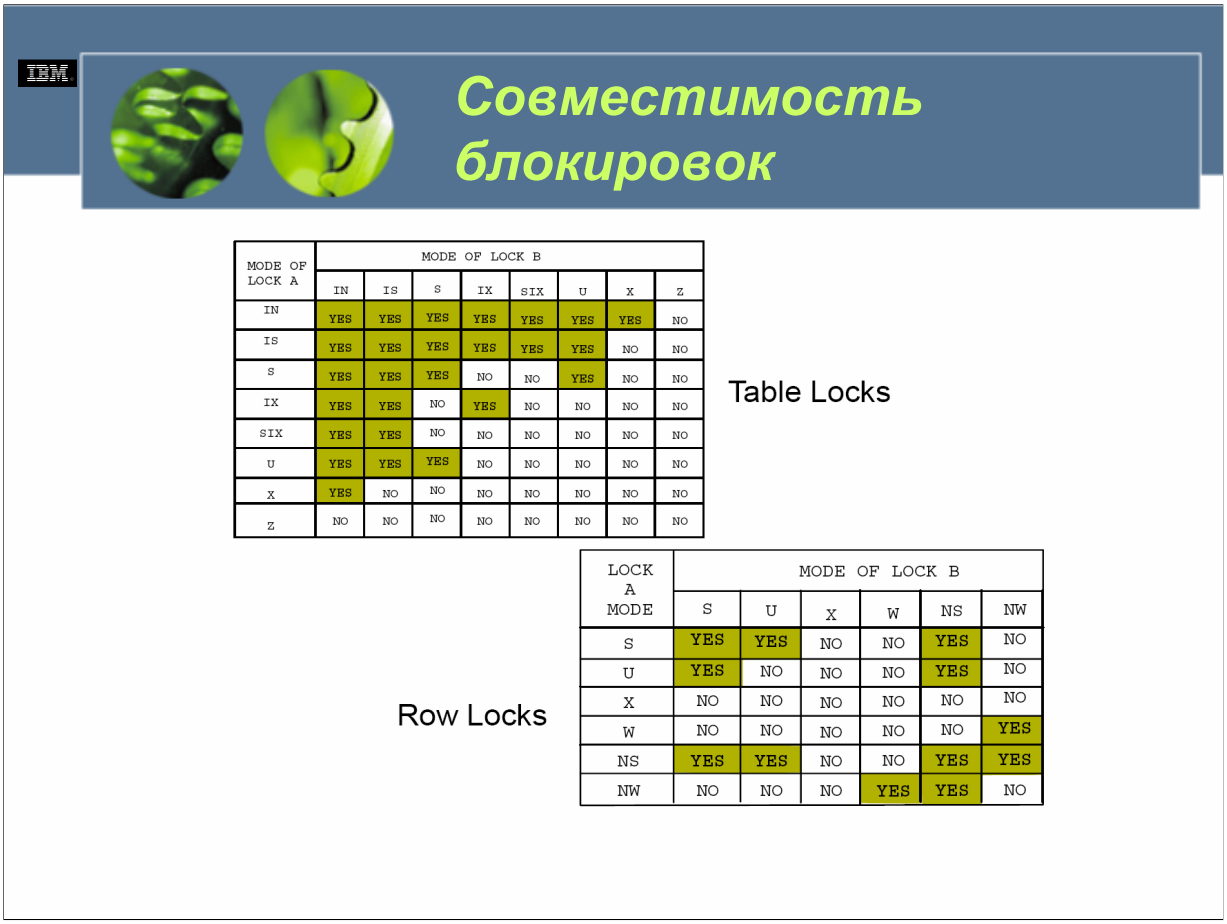
расширение блокировок (lock escalation);

ожидания и тайм-ауты блокировок (lock waits и timeouts);

тупиковые ситуации (deadlocks);

одновременность и размер (concurrency и granularity).

Знание того, что представляют собой эти факторы, и понимание того, как они влияют на общую производительность, может помочь вам в проектировании приложений баз данных, которые хорошо работают в многопользовательских средах баз данных и опосредованно предоставить вам больший контроль над тем, как используются блокировки.

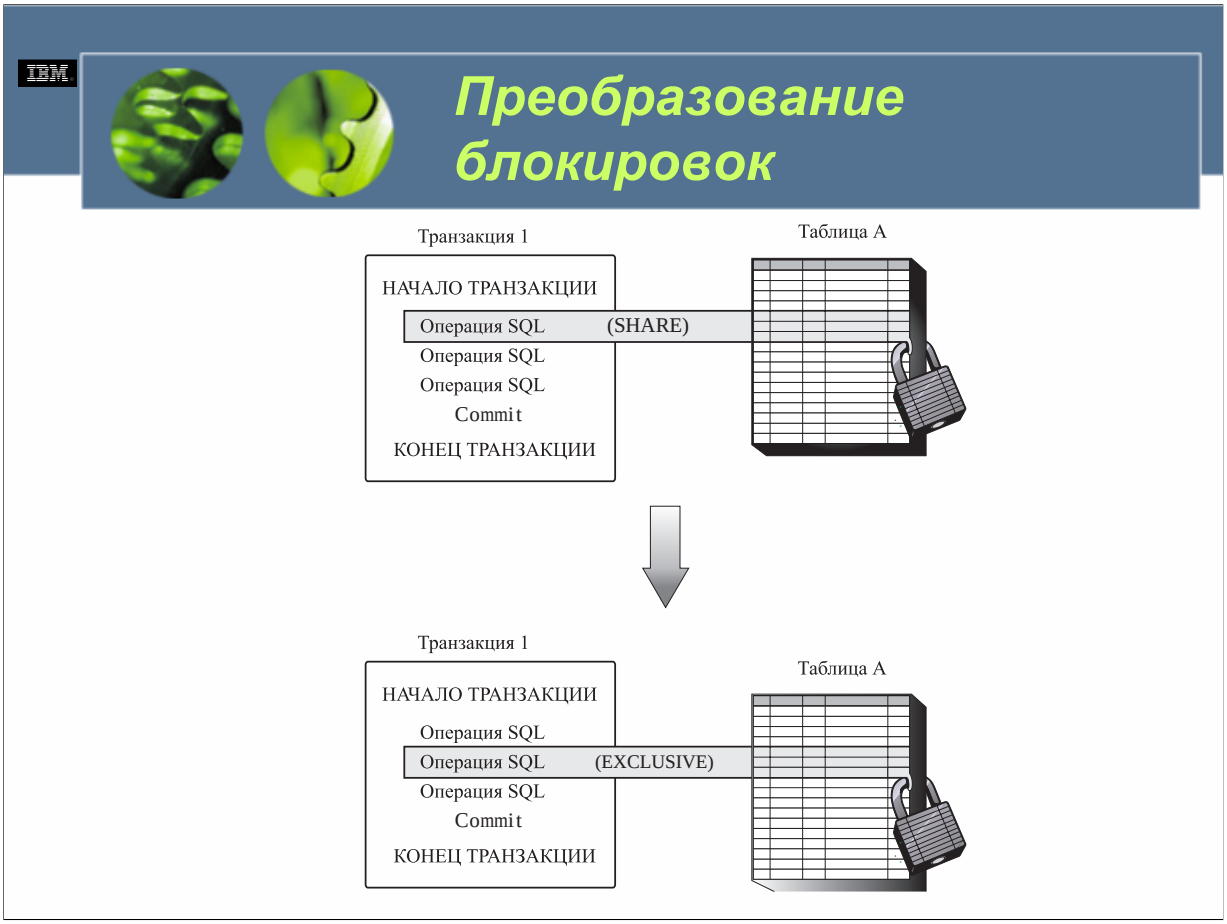


MODE OF LOCK A	MODE OF LOCK B							
	IN	IS	S	IX	SIX	U	X	Z
IN	YES	YES	YES	YES	YES	YES	YES	NO
IS	YES	YES	YES	YES	YES	YES	NO	NO
S	YES	YES	YES	NO	NO	YES	NO	NO
IX	YES	YES	NO	YES	NO	NO	NO	NO
SIX	YES	YES	NO	NO	NO	NO	NO	NO
U	YES	YES	YES	NO	NO	NO	NO	NO
X	YES	NO	NO	NO	NO	NO	NO	NO
Z	NO	NO	NO	NO	NO	NO	NO	NO

LOCK A MODE	MODE OF LOCK B					
	S	U	X	W	NS	NW
S	YES	YES	NO	NO	YES	NO
U	YES	NO	NO	NO	YES	NO
X	NO	NO	NO	NO	NO	NO
W	NO	NO	NO	NO	NO	YES
NS	YES	YES	NO	NO	YES	YES
NW	NO	NO	NO	YES	YES	NO

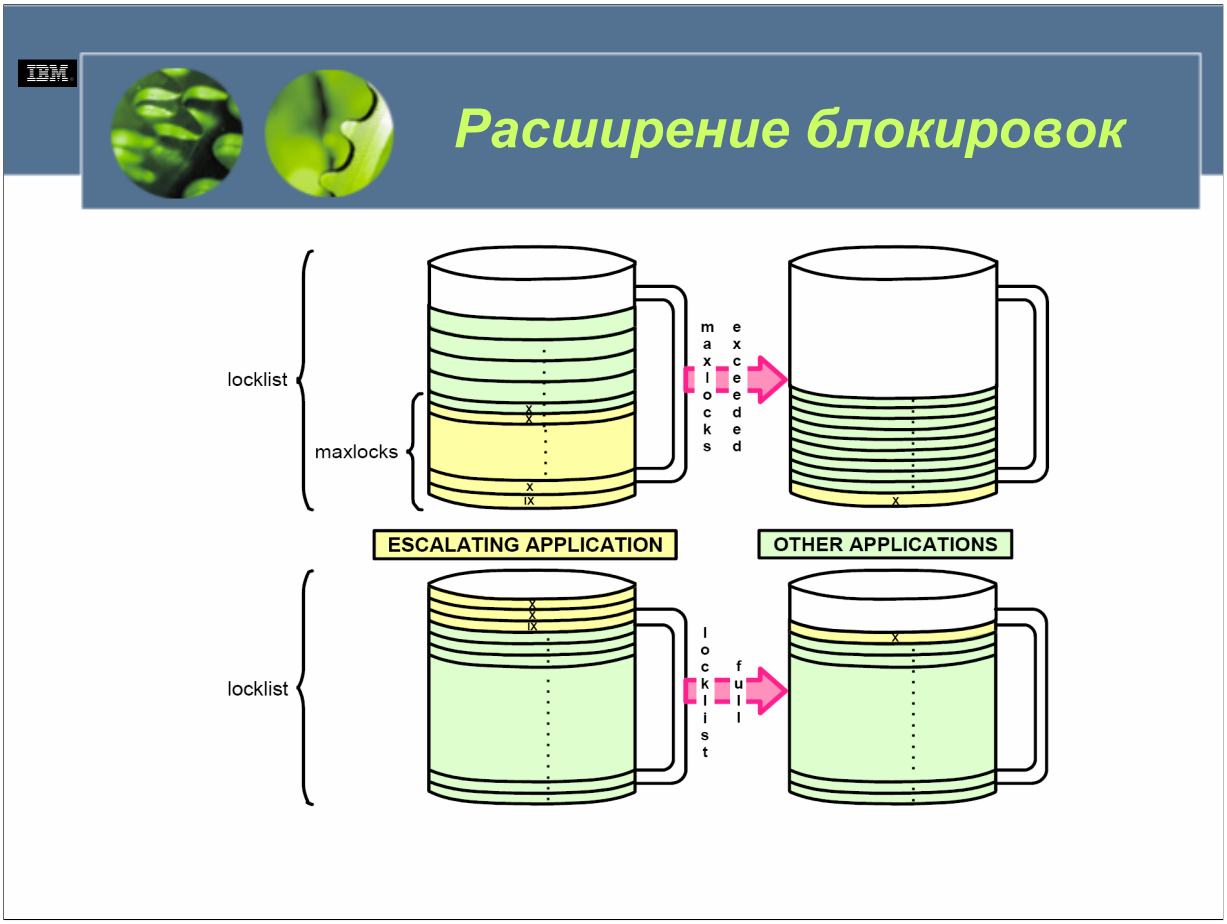
WCF03.0

Если состояние блокировки, наложенной на ресурс данных одной транзакцией, такое, что на тот же самый ресурс другой транзакцией может быть помещена другая блокировка до того, как первая полученная блокировка освобождена, говорят, что эти две блокировки (или состояния блокировок) *совместимы*. Каждый раз, когда одна транзакция удерживает блокировку для ресурса данных, а другая транзакция пытается получить блокировку на тот же самый ресурс, менеджер баз данных DB2 проверяет состояния этих двух блокировок для определения того, являются ли они совместимыми. Если две блокировки совместимы, от имени запрашивающей транзакции устанавливается вторая блокировка. С другой стороны, если блокировки несовместимы, запрашивающая транзакция должна ждать, пока владеющая транзакция освободит блокировку, которую она удерживает, прежде чем можно будет получить доступ к ресурсу и получить нужную блокировку. Это известно как *событие ожидания блокировки*. Когда происходит событие ожидания блокировки, транзакция, пытающаяся получить доступ к заблокированному ресурсу, просто приостанавливает выполнение до тех пор, пока владеющая транзакция не завершится (и освободит несовместимую блокировку), или до тех пор, пока не возникнет *событие тайм-аута блокировки* (мы вскоре рассмотрим события тайм-аутов блокировок).



WCF03.0

В большинстве случаев преобразование блокировки осуществляется для блокировок на уровне строк, и процесс преобразования довольно прост. Например, если удерживается блокировка обновления (U), а нужна монопольная (X) блокировка, блокировка обновления (U) будет преобразована в монопольную (X). Однако совместно используемая блокировка (S) и блокировка намерения монопольного (IX) являются особыми случаями, поскольку ни одна из них не считается более ограничивающей, чем другая; если удерживается одна из этих блокировок и запрошена другая, удерживаемая блокировка преобразуется в совместно используемую блокировку с намерением монопольного (SIX). Со всеми другими преобразованиями состояние текущей блокировки изменяется на запрашиваемое состояние блокировки при условии, – что запрашиваемое состояние блокировки является более ограничивающим. (Преобразование блокировки происходит лишь в том случае, если удерживаемая блокировка увеличивает ограничение.) После преобразования блокировки она остается на высшем достигнутом уровне до тех пор, пока транзакция, удерживающая блокировку, не завершится, и блокировка не будет освобождена.

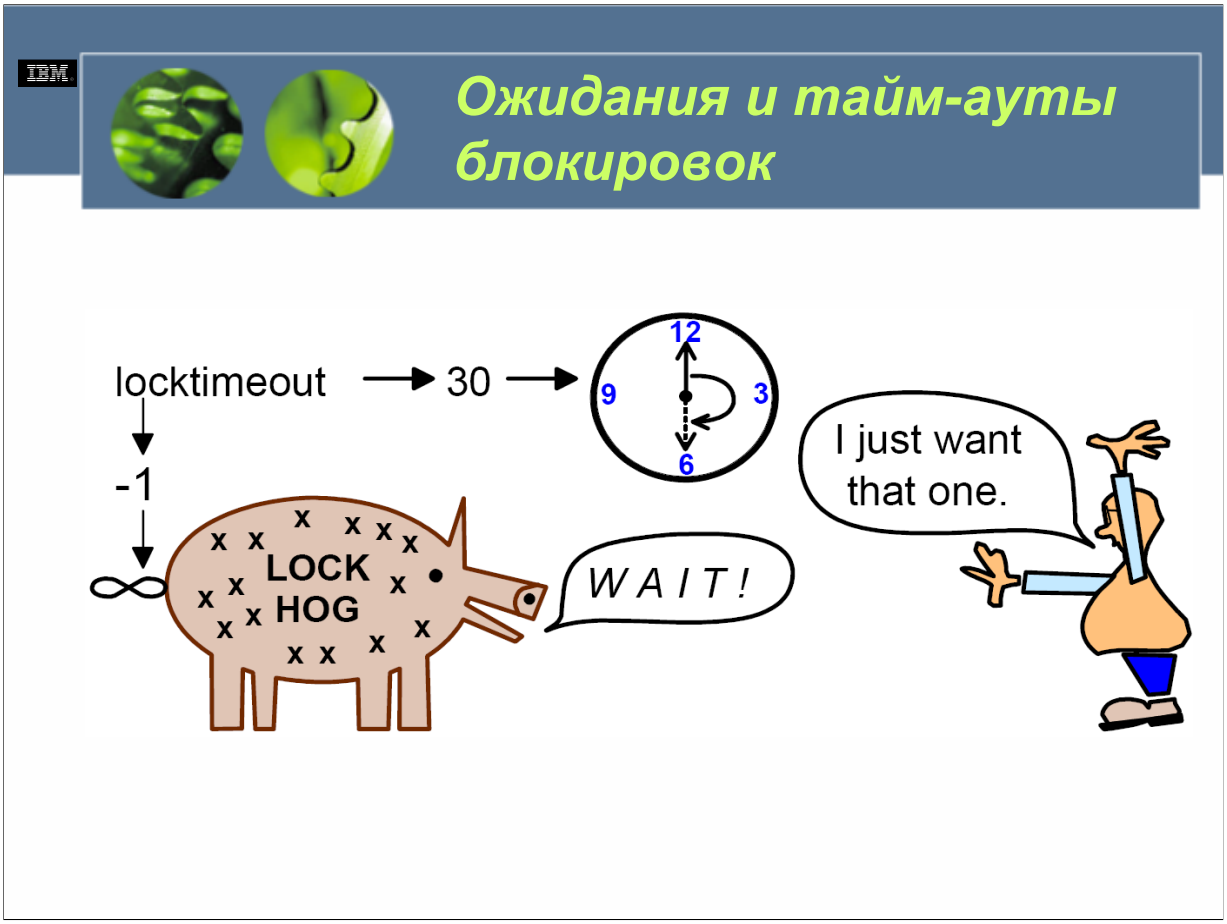


WCF03.0

Когда впервые устанавливается соединение с базой данных, резервируется определенный объем памяти для хранения структуры, которую DB2 UDB использует для управления блокировками. Эта структура называется *списком блокировок*, и именно здесь хранятся после получения блокировки, удерживаемые всеми приложениями, одновременно подключенными к базе данных. (Действительный объем памяти, резервируемый для списка блокировок, определяется параметром конфигурации базы данных *locklist.*)

Поскольку доступно ограниченное количество памяти и поскольку эта память должна совместно использоваться всеми, менеджер баз данных DB2 накладывает ограничение на объем пространства в списке блокировок, который каждая транзакция может использовать для своих собственных блокировок (что определяется параметром конфигурации базы данных *maxlocks*). Чтобы предотвратить определенным агентом базы данных превышение ограничения своего размера пространства, каждый раз, когда от имени одной транзакции было получено слишком много блокировок (независимо от их типа), выполняется процесс, известный как *расширение блокировок*. В ходе расширения блокировок пространство в списке блокировок освобождается за счет преобразования нескольких блокировок на уровне строк в одну блокировку на уровне таблицы.

Так как же работает расширение блокировок? Когда транзакция запрашивает блокировку, а список блокировок базы данных полон, выбирается одна из таблиц, связанная с транзакцией, запрашивается блокировка на уровне таблицы, все блокировки на уровне строк для этой таблицы освобождаются для образования свободного пространства, и полученная блокировка на уровне таблицы добавляется в список блокировок. Если этот процесс не освобождает пространство памяти, необходимое для получения запрошенной блокировки, выбирается другая таблица, и процесс повторяется до тех пор, пока не будет получено достаточно свободной памяти. Только после этого будет получена запрошенная блокировка, и транзакции будет разрешено продолжить выполнение. Однако если после расширения всех блокировок транзакции на уровне строк необходимое пространство списка блокировок по-прежнему недоступно, генерируется код ошибки SQL, все изменения, которые были сделаны с базой данных с момента инициирования транзакции, откатываются, и транзакция постепенно завершается. Если база данных сконфигурирована должным образом, события расширения блокировок возникают редко.

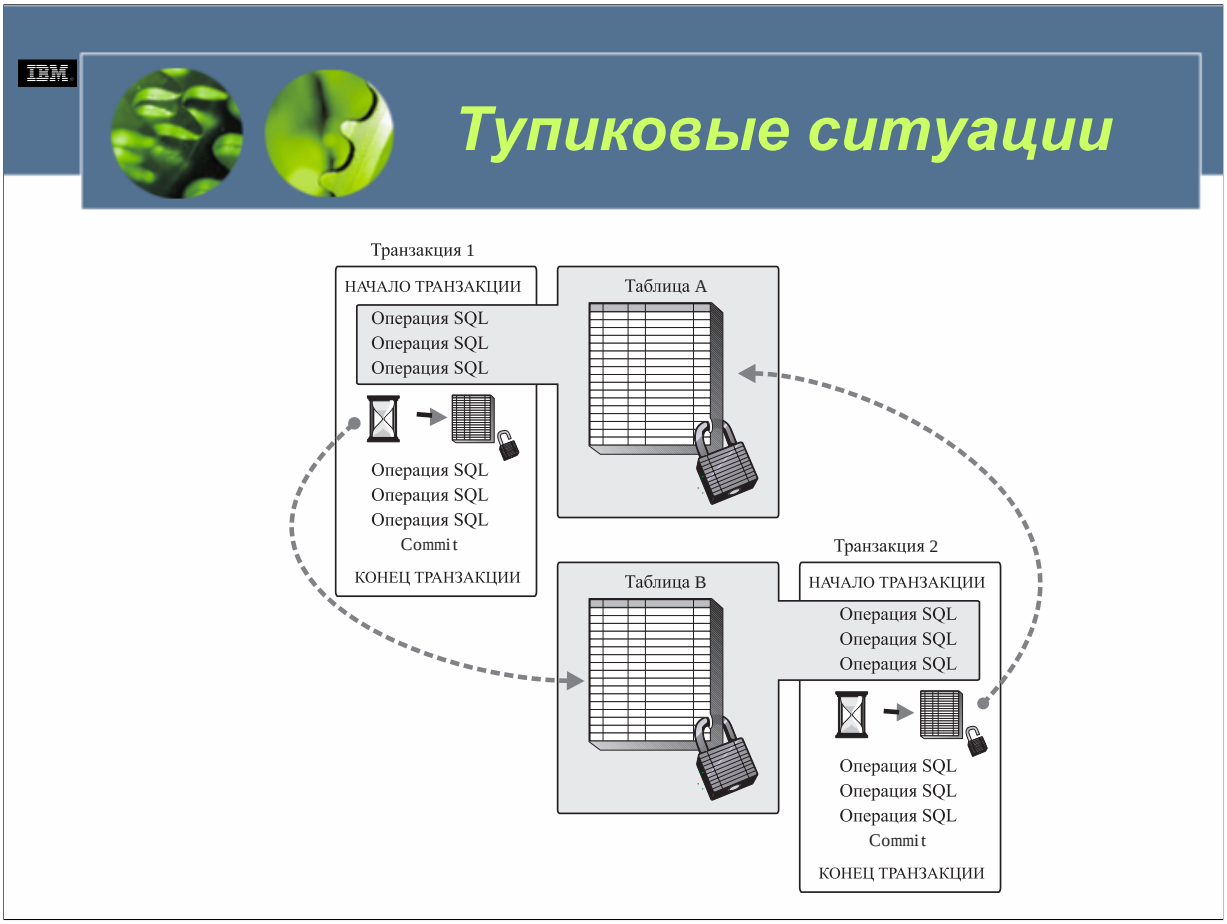


WCF03.0

Каждый раз, когда транзакция удерживает блокировку для определенного ресурса (табличного пространства, таблицы или строки), другим транзакциям может быть отказано в доступе к этому ресурсу до тех пор, пока владеющая транзакция не завершится и не освободит все полученные блокировки. Из-за такого поведения без определенного механизма обнаружения тайм-аута транзакция может ждать освобождения блокировки неопределенно долго. Например, если транзакция в одном приложении пользователя ожидала освобождения блокировки, удерживаемой транзакцией в приложении другого пользователя, а этот другой пользователь оставил свою рабочую станцию, не сделав какого-либо действия, которое позволило бы его приложению завершить владеющую транзакцию и освободить все удерживаемые блокировки, приложение, ожидающее освобождения блокировки, было бы неспособно продолжить работу в течение неопределенного промежутка времени. К сожалению, было бы также невозможно завершить приложение, ожидающее освобождения блокировки, не нарушив согласованности данных.

Чтобы предотвратить возникновение подобных ситуаций, в менеджер баз данных DB2 была включена возможность *определения тайм-аута блокировки*. При использовании эта возможность предотвращает неопределенное ожидание приложениями освобождения блокировки в ненормальных ситуациях. Присвоив значение параметру конфигурации *locktimeout* в файле конфигурации соответствующей базы данных, вы можете контролировать, когда произойдет определение тайм-аута блокировки. Этот параметр контролирует промежуток времени, в течение которого любая транзакция будет ждать получения запрошенной блокировки; если запрошенная блокировка не получена до истечения интервала времени, указанного в параметре конфигурации *locktimeout*, ожидающее приложение получает сообщение об ошибке, и транзакция, запросившая блокировку, откатывается. После отката транзакции, запрашивающей блокировку, ожидающее приложение по умолчанию будет завершено. (Такое поведение предотвращает возникновение несогласованности данных.)

По умолчанию конфигурационный параметр *locktimeout* установлен в –1, что означает, что приложения будут ждать получения нужной блокировки бесконечно. Во многих случаях это значение следует заменить на какое-либо другое, отличное от значения по умолчанию. Кроме того, приложения должны быть написаны таким образом, чтобы они фиксировали любой код SQL тайм-аута (или тупиковой ситуации), возвращаемый менеджером баз данных DB2, и реагировали соответствующим образом.



WCF03.0

Мы только что видели, как одна транзакция может заставить другую ждать в течение неопределенно долгого периода времени, пока блокировка будет освобождена, и как эту ситуацию можно разрешить, установив тайм-ауты блокировок. К сожалению, есть ситуация, когда борьбу за получение блокировки со стороны двух или более транзакций невозможно разрешить посредством тайм-аута блокировки. Эта ситуация известна как *тупиковая ситуация (deadlock)*, и лучшим способом продемонстрировать то, как она может возникнуть, является пример: предположим, что транзакция 1 получает монопольную (X) блокировку для таблицы A, а транзакция 2 получает монопольную (X) блокировку для таблицы B. Теперь предположим, что транзакция 1 пытается получить монопольную (X) блокировку для таблицы B, а транзакция 2 пытается получить монопольную (X) блокировку для таблицы A. Мы уже видели, что обработка обеих транзакций будет приостановлена до тех пор, пока не будет удовлетворен их второй запрос блокировки. Однако, поскольку ни один запрос не может быть удовлетворен до тех пор, пока одна из владеющих транзакций не освободит удерживаемую в настоящее время блокировку (выполнив операцию принятия или отката), и поскольку ни одна транзакция не может выполнить операцию принятия или отката из-за того, что обе они приостановлены (и ожидают блокировки), возникла тупиковая ситуация.

Когда возникает тупиковый цикл, все вовлеченные транзакции будут ждать в течении неопределенного времени, пока будет освобождена блокировка, пока не вмешается какой-нибудь внешний фактор и не прервет тупиковый цикл. В DB2 Universal Database таким фактором является фоновый процесс, который называется *детектором тупиковых ситуаций (deadlock detector)*, единственным назначением которого является нахождение и разрешение всех тупиковых ситуаций, найденных в подсистеме блокирования.

У каждой базы данных есть свой собственный детектор тупиковых ситуаций, который активируется как часть процесса инициализации базы данных. После активации детектор тупиковых ситуаций остается в течение большей части времени «спящим», но «просыпается» через заданные заранее интервалы времени и проверяет подсистему блокирования для определения того, не существует ли тупиковая ситуация. Обычно детектор тупиковых ситуаций просыпается, видит, что в подсистеме блокирования тупиковых ситуаций нет, и засыпает снова. Однако если детектор тупиковых ситуаций обнаруживает в подсистеме блокирования тупиковый цикл, он случайным образом выбирает одну транзакцию в цикле, чтобы откатить ее и завершить. Выбранная транзакция получает код ошибки SQL, и каждая полученная ею блокировка освобождается. Затем оставшиеся транзакции могут продолжить, поскольку тупиковый цикл был разорван.