**Модуль 1. Введение в теорию баз данных (2 пары)**

**Цель занятия:** изучить историю возникновения баз данных, познакомиться с терминологией в данной области, изучить модели баз данных, рассмотреть реляционную модель данных, 12 правил Кодда.

**План занятия:**

1. Введение в теорию баз данных:

* история и этапы развития;
* понятия база данных и система управления базами данных;
* сравнение существующих моделей баз данных:
* файловая модель;
* сетевая модель;
* иерархическая модель;
* реляционная модель;
* объектно-ориентированная модель;
* понятие реляционной модели баз данных;
* двенадцать правил Кодда;
* сравнительный анализ СУБД Microsoft SQL Server с существующими системами управления базами данных.

2. Основы взаимодействия с Microsoft SQL Server:

* версии и редакции Microsoft SQL Server;
* инсталляция Microsoft SQL Server;
* инструменты управления и утилиты MS SQL Server;
* управление базой данных:
* создание базы данных;
* настройка параметров базы данных;
* изменение размера базы данных;
* переименование базы данных;
* управление группами файлов;
* удаление базы данных.

1. **Введение в теорию баз данных.**
   1. **История и этапы развития.**

Современный мир информационных технологий трудно представить себе без использования баз данных. Практически все системы в той или иной степени связаны с функциями долговременного хранения и обработки информации. Фактически информация становится фактором, определяющим эффективность любой сферы деятельности.

С самого начала развития вычислительной техники образовались *два основных направления* ее использования.

Первое направление — *применение вычислительной техники для выполнения численных расчетов*, которые слишком долго или вообще невозможно производить вручную.

Второе направление — это *использование средств вычислительной техники в автоматических или автоматизированных информационных системах* (автоматизированная система - это система, часть функций которой выполняется автоматически, а часть оператором (организатором)).

**Информационная система** представляет собой программно-аппаратный комплекс, обеспечивающий выполнение следующих функций:

• надежное хранение информации в памяти компьютера;

• выполнение специфических для данного приложения преобразований информации и вычислений;

• предоставление пользователям удобного и легко осваиваемого интерфейса.

Обычно такие системы имеют дело с большими объемами информации, имеющей достаточно сложную структуру. Классическими примерами информационных систем являются банковские системы, автоматизированные системы управления предприятиями, системы резервирования авиационных или железнодорожных билетов, мест в гостиницах и т. д.

*Системы баз данных* сегодня являются основой построения большинства информационных систем и используются при автоматизации практически всех сфер человеческой деятельности. Например, доступ к базе данных необходим при работе с библиотечной информационной системой, содержащей сведения обо всех книгах, имеющихся в библиотеке, ее читателях, заявках на бронирование книг и т.д. В ней обычно содержатся средства, позволяющие читателям находить нужную им книгу по названию, фамилиям авторов или указанной тематике. С помощью такого рода систем организуется учет движения книг, другие операции, необходимые в библиотечной деятельности.

В ВУЗе могут существовать базы данных с информацией о студентах, профессорско-преподавательском составе, факультетах и кафедрах, др. данные, необходимые для функционирования так называемых комплексных информационно-аналитических систем и их подсистем (учета кадров, бухгалтерской, документооборота, информационного обеспечения учебной деятельности и т.п.).

**История развития баз данных** представляет собой историю развития систем управления данными во внешней памяти ЭВМ. На первых электронно-вычислительных машинах существовало 2 вида внешних устройств – *магнитные ленты и магнитные барабаны*.

*Магнитные ленты* обладали достаточно большой емкостью, но основным их недостатком было то, что для чтения информации, находящейся в середине или конце ленты необходимо было прочитать весь предыдущий участок.

*Магнитные барабаны* давали возможность произвольного доступа к памяти, но объем хранимой на них информации был ограничен. В тот момент говорить о какой-либо системе управления данными во внешней памяти не приходилось. Каждая прикладная программа, которой требовалось хранить данные во внешней памяти, сама определяла расположение данных на магнитной ленте или барабане. Функции информационного обмена между оперативной и внешней памятью, именование и структуризацию данных так же выполняла прикладная программа.

Можно предположить, что именно требования нечисловых приложений вызвали появление **съемных магнитных дисков с подвижными головками**, что явилось революцией в истории вычислительной техники. Магнитные диски впервые были реализованы в **1956 году** в исследовательской лаборатории корпорации IBM, расположенной в Сан-Хосе (Калифорния), где был выпущен **серийный дисковый накопитель IBM 350** — первое устройство с подвижной головкой для чтения и записи. Вот как выглядел первый жесткий диск размером 5 мегабайт более 60 лет тому назад. Без автопогрузчика его было не поднять. Железо предназначалось для первого «SUPER» компьютера с жестким диском 305 RAMAC. Весила система около тонны, — получается по 0,2 грамма за байт (или 5 килобайт в 1 кг) и состояла из 50-ти дисков диаметром в 24 дюйма (610 мм). Каждый диск соответственно 100 килобайт. Частота вращения барабана — 1200 оборотов в минуту. Время произвольного доступа — 600 миллисекунд.

Важным шагом в развитии баз данных явился переход к использованию централизованных систем управления файлами, или, используя общепринятый в данный момент термин, **файловым системам**.

С точки зрения прикладной программы **файл** — это именованная область внешней памяти, в которую можно записывать и из которой можно считывать данные.

**Файловая система —** порядок, определяющий формат содержимого и способ физического хранения информации, которую принято группировать в виде файлов. Конкретная файловая система определяет размер имен файлов и (каталогов), максимальный возможный размер файла и раздела, набор атрибутов файла.

Первая развитая файловая система была разработана фирмой **IBM** для ее серии **System/360** в **1964** году.

Пользователи видят файл как линейную последовательность записей и могут выполнять над ними ряд стандартных операций:

• Создать файл (требуемого типа и размера);

• Открыть ранее созданный файл;

• Прочитать из файла некоторую запись (текущую, следующую, предыдущую, первую, последнюю);

• Записать в файл на место текущей записи новую, добавить новую запись в конец файла.

В разных файловых системах эти операции могли несколько отличаться, но общий смысл их был именно таким.

Несмотря на относительную простоту организации, файловые системы имеют ряд недостатков:

**1. Избыточность данных.** Файловые системы характеризуются значительной избыточностью, поскольку нередко для решения различных задач управления используются одни и одни и те же данные, размещенные в разных файлах. Из-за дублирования данных в разных файлах память на внешних запоминающих устройствах используется неэкономно, информация одного и одного и того же объекта управления распределяется между многими файлами. При этом довольно тяжело представить общую информационную модель предметной области.

**2. Несогласованность данных.** Учитывая, что одна и одна и та же информация может размещаться в разных файлах, технологически тяжело проследить за внесением изменений одновременно во все файлы. Из-за этого может возникнуть несогласованность данных, когда одно и одно и то же поле в разных файлах может иметь разные значения.

**3. Зависимость структур данных и прикладных программ.** При файловой организации логическая и физическая структуры файла должны соответствовать их описанию в прикладной программе. Прикладная программа должна быть модифицирована при любом изменении логической или физической структуры файла. Поскольку изменения в одной программе часто требуют внесения изменений в другие информационно-связанные программы, то иногда проще создать новую программу, чем вносить изменения в старую. Поэтому этот недостаток файловых систем приводит к значительному увеличению стоимости сопровождения программных средств. Иногда стоимость сопровождения программных средств может достигать близко 70 % стоимости их разработки.

Эти недостатки послужили тем толчком, который заставил разработчиков информационных систем предложить новый подход к управлению информацией. Этот подход был реализован в рамках новых программных систем, названных впоследствии *системами управления базами данных* (СУБД), а сами хранилища информации, которые работали под управлением данных систем, назвали *базами или банками данных (БД и БнД)*.

В истории развития СУБД и БД можно выделить ***4 основных этапа***. Однако необходимо заметить, что все же нет жестких временных ограничений этих этапов: они плавно переходят один в другой и даже сосуществуют параллельно. Тем не менее, выделение этих этапов позволит более четко охарактеризовать отдельные стадии развития технологии баз данных, подчеркнуть особенности, характерные для конкретного этапа.

**История баз данных насчитывает около 50 лет.**

**В 1968 году** была введена в эксплуатацию первая промышленная СУБД система **IMS (Information Management System) фирмы IBM**. Главным архитектором СУБД был Верн Уоттс. Начав работу в IBM в 1956 году, он непрерывно работал над IMS начиная от времени её первоначального проектирования вплоть до своей кончины 4 апреля 2009 года. В задачу IMS входила обработка спецификации изделия для ракеты Сатурн-5 и кораблей Аполлон.

В качестве носителя информации использовалась магнитная лента, а в качестве структуры данных – иерархическая модель.

В **1975** году появился первый стандарт ассоциации по языкам систем обработки данных — **Conference of Data System Languages (CODASYL)**, который определил ряд фундаментальных понятий в теории систем баз данных, которые и до сих пор являются основополагающими для сетевой модели данных.

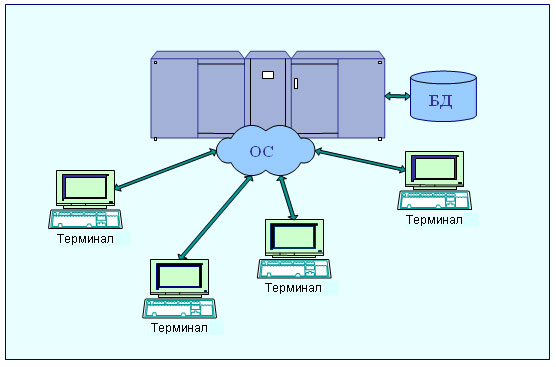
В дальнейшее развитие теории баз данных большой вклад был сделан британским ученым **Эдгаром Франком Коддом**, который является *создателем реляционной модели данных*. В **1981** году Э. Ф. Кодд получил за создание реляционной модели и реляционной алгебры престижную премию Тьюринга Американской ассоциации по вычислительной технике.

**Первый этап — базы данных на больших ЭВМ.**

На сегодняшний день история развития СУБД насчитывает около 50 лет.

**Первый этап** развития СУБД связан с **организацией баз данных на больших машинах** типа IBM 360/370, ЕС-ЭВМ и мини-ЭВМ типа PDP11 (фирмы Digital Equipment Corporation — DEC), разных моделях HP (фирмы Hewlett Packard).

Базы данных хранились во внешней памяти центральной ЭВМ, пользователями этих баз данных были задачи, запускаемые в основном в пакетном режиме. Интерактивный режим доступа обеспечивался с помощью консольных терминалов, которые не обладали собственными вычислительными ресурсами (процессором, внешней памятью) и служили только устройствами ввода-вывода для центральной ЭВМ. Программы доступа к БД писались на различных языках и запускались как обычные числовые программы. Мощные операционные системы обеспечивали возможность условно-параллельного выполнения всего множества задач. Эти системы можно было отнести к системам распределенного доступа, потому что *база данных была централизованной*, хранилась на устройствах внешней памяти одной центральной ЭВМ, а доступ к ней поддерживался от многих пользователей-задач (рис. 1.1).



###### Рис. 1.1. Мэйнфреймовая архитектура

Все СУБД базировались на мощных мультипрограммных операционных системах (MVS, SVM, RTE. OSRV, RSX, UNIX), поэтому в основном поддерживается работа с централизованной базой данных в режиме распределенного доступа (база данных была централизованной, хранилась на устройствах внешней памяти одной центральной ЭВМ, а доступ к ней поддерживался от многих пользователей-задач).

Функции управления распределением ресурсов в основном осуществляются операционной системой (ОС).

Значительная роль отводится администрированию данных.

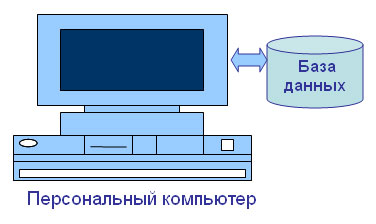
Поддерживаются языки низкого уровня манипулирования данными, ориентированные на навигационные методы доступа к данным.

**Второй этап. Эпоха персональных компьютеров**

Звание первого персонального компьютера принадлежит модели **5100** производства фирмы **IBM**, выпущенной в **1975** году. Он был более компактным, чем мэйнфреймы, имел встроенные монитор, клавиатуру и накопитель на магнитной ленте, и предназначался для решения научно-инженерных задач.

Первым же **массовым персональным компьютером** производства фирмы **IBM**, выпущенным в **1981** году, стал **IBM PC** модели **5150**, положивший начало семейству наиболее распространённых современных персональных компьютеров.

С появлением персональных компьютеров изменились условия использования вычислительной техники в организациях и фирмах. Так как техника стала доступна, компьютеры превратились в удобный инструмент для ведения документации и учетных функций фирм. В это время появляются так называемые ***настольные СУБД***, позволяющие, с одной стороны, хранить в упорядоченном виде большие объемы информации, и, с другой стороны, имеющие удобный интерфейс для заполнения данных и генерации различных отчетов.



###### Рис. 1.2. БД на персональных компьютерах

Наличие на рынке большого числа СУБД, выполняющих сходные функции, потребовало разработки методов экспорта, импорта и открытых форматов хранения данных. Так появились *первые коммерческие СУБД с реляционной моделью данных*.

Основанные на реляционном подходе СУБД для персональных компьютеров принято считать **системами второго поколения**.

В 80-х годах были созданы различные коммерческие реляционные СУБД - например, DB2 или SQL/DS корпорации IBM, Oracle и др. Большинство СУБД имели развитый и удобный пользовательский интерфейс, предлагающий интерактивный режим работы с БД, как в рамках описания БД, так и в рамках проектирования запросов.

*Особенности второго этапа* следующие:

* Главное ограничение при работе с настольными СУБД накладывалось **монопольным доступом**, поскольку первое время персональные компьютеры не были подключены к вычислительным сетям. Базы данных на них создавались для работы одного пользователя.
* Большинство СУБД имели развитый и удобный пользовательский интерфейс.
* Большинство СУБД предлагали развитый и удобный инструментарий для разработки готовых приложений без программирования.
* Во всех настольных СУБД поддерживался только внешний уровень представления реляционной модели, то есть только внешний табличный вид структур данных.
* В настольных СУБД отсутствовали средства поддержки ссылочной и структурной целостности базы данных.
* При наличии высокоуровневых языков манипулиропання данными типа реляционной алгебры и SQL в настольных СУБД поддерживались низкоуровневые языки манипулирования данными на уровне отдельных строк таблиц.
* Наличие монопольного режима работы фактически привело к вырождению функций администрирования БД и в связи с этим - к отсутствию инструментальных средств администрирования БД
* Сравнительно скромные требования к аппаратному обеспечению со стороны настольных СУБД. Вполне работоспособные приложения, разработанные, например, на Clipper, работали на PC 286,

Яркие представители этого семейства - Dbase, FoxPro, Clipper, Paradox.

***Третий этап - распределенные базы данных.***

После процесса «персонализации» начался обратный процесс - **интеграция**. С увеличением количества локальных сетей, все больше информации стало передаваться между компьютерами. Остро встал вопрос согласованности данных, хранящихся и обрабатывающихся в разных местах, но логически друг с другом связанных. Успешное решение этих задач приводит к появлению **распределенных баз данных**, сохраняющих все преимущества настольных СУБД и в тоже время позволяющих организовывать параллельную обработку информации и поддержку целостности баз данных.

*Особенности третьего этапа:*

* поддержка полной реляционной модели, а именно:
  + *структурной целостности* — допустимыми являются только данные, представленные в виде отношений реляционной модели;
  + *языковой целостности*, то есть языков манипулирования данными высокого уровня (в основном SQL);
  + *ссылочной целостности*, контроля за соблюдением ссылочной целостности в течение всего времени функционирования системы, и гарантий невозможности со стороны СУБД нарушить эти ограничения.
* Большинство современных СУБД рассчитаны на многоплатформенную архитектуру (компьютеры с разной архитектурой, разной ОС, при этом для пользователя доступ к данным, управляемым СУБД на различных платформах практически неразличим)
* необходимость поддержки многопользовательской работы с базой данных и возможность децентрализованного хранения данных с реализацией общей концепции средств защиты данных и как следствие развитие средств администрирования БД

Представителями СУБД, относящимся к этому этапу, можно считать MS Access 2007 и все современные серверы баз данных Oracle 11g, MS SQL Server, System 11, Informix и др.

***IV этап***

Этот этап характеризуется появлением новой технологии доступа к данным — **интранет**.

Интранет (англ. Intranet, также употребляется термин интрасеть) — в отличие от Интернета, это внутренняя частная сеть организации. Как правило, интранет — это Интернет в миниатюре, который построен на использовании протокола IP для обмена и совместного использования некоторой части информации внутри этой организации. Это могут быть списки сотрудников, списки телефонов партнёров и заказчиков. Чаще всего под этим термином имеют в виду только видимую часть интранет — внутренний веб-сайт организации. Основанный на базовых протоколах HTTP и HTTPS и организованный по принципу клиент-се́рвер, интранет-сайт доступен с любого компьютера через браузер. Таким образом, интранет — это «частный» Интернет, ограниченный виртуальным пространством отдельно взятой организации. Intranet допускает использование публичных каналов связи, входящих в Internet, (VPN), но при этом обеспечивается защита передаваемых данных и меры по пресечению проникновения извне на корпоративные узлы.

Приложения в intranet основаны на применении Internet-технологий и в особенности Web-технологии: гипертекст в формате HTML, протокол передачи гипертекста HTTP и интерфейс се́рверных приложений CGI. Составными частями Intranet являются Web-се́рверы для статической или динамической публикации информации и браузеры для просмотра и интерпретации гипертекста.Основное отличие этого подхода от технологии клиент-сервер состоит в том, что отпадает необходимость использования специализированного клиентского программного обеспечения. Для работы с удаленной базой данных используется стандартный браузер Интернета, например Microsoft Internet Explorer или Netscape Navigator, и для конечного пользователя процесс обращения к данным происходит аналогично работе во Всемирной Паутине (см. рис. 1.4). При этом встроенный в загружаемые пользователем HTML-страницы код, написанный обычно на языке Java, Java-script, Perl и других, отслеживает все действия пользователя и транслирует их в низкоуровневые SQL-запросы к базе данных, выполняя, таким образом, ту работу, которой в технологии клиент-сервер занимается клиентская программа. Удобство данного подхода привело к тому, что он стал использоваться не только для удаленного доступа к базам данных, но и для пользователей локальной сети предприятия. Простые задачи обработки данных, не связанные со сложными алгоритмами, требующими согласованного изменения данных во многих взаимосвязанных объектах, достаточно просто и эффективно могут быть построены по данной архитектуре. В этом случае для подключения нового пользователя к возможности использовать данную задачу не требуется установка дополнительного клиентского программного обеспечения. Однако алгоритмически сложные задачи рекомендуется реализовывать в архитектуре «клиент-сервер» с разработкой специального клиентского программного обеспечения.



Рис. 1.4. Стандартный алгоритм взаимодействия Web-клиента и Web-сервера

Представитель этого семейства: MySQL.

* 1. **Понятия «база данных» и «система управления базами данных»**

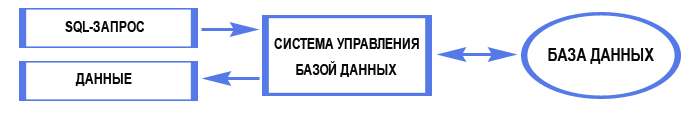
Давайте разберемся с терминами и понятиями баз данных:

*База данных* - набор сведений, хранящихся некоторым упорядоченным способом. База данных(БД) — именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области. Можно сравнить базу данных со шкафом, в котором хранятся документы. Иными словами, база данных - это хранилище данных. Сами по себе базы данных не представляли бы интереса, если бы не было систем управления базами данных (СУБД).

*Система управления базами данных* - это совокупность языковых и программных средств, которая осуществляет доступ к данным, позволяет их создавать, менять и удалять, обеспечивает безопасность данных и т.д. В общем СУБД - это система, позволяющая создавать базы данных и манипулировать сведениями из них. А осуществляет этот доступ к данным СУБД посредством специального языка - SQL.

*SQL* - язык структурированных запросов, основной задачей которого является предоставление простого способа считывания и записи информации в базу данных.

Итак, простейшая схема работы с базой данных выглядит примерно так:



БД должна соответствовать следующей системе требований:

* многократное использование;
* быстрый поиск и получение информации по запросам;
* простота обновления данных;
* уменьшение излишней избыточности данных;
* отсутствие дублирования данных;
* защита от несанкционированного доступа;
* защита от уничтожения данных;
* максимальная независимость от прикладных программ.

Программы, с помощью которых пользователи работают с базой данных, называются приложениями. В общем случае с одной базой данных могут работать множество различных приложений. Например, если база данных моделирует некоторое предприятие, то для работы с ней может быть создано приложение, которое обслуживает подсистему учета кадров, другое приложение может быть посвящено работе подсистемы расчета заработной платы сотрудников, третье приложение работает как подсистема складского учета, четвертое приложение посвящено планированию производственного процесса. При рассмотрении приложений, работающих с одной базой данных, предполагается, что они могут работать параллельно и независимо друг от друга, и именно СУБД призвана обеспечить работу множества приложений с единой базой данных таким образом, чтобы каждое из них выполнялось корректно, но учитывало все изменения в базе данных, вносимые другими приложениями

* 1. **Сравнение существующих моделей баз данных**

Одними из основополагающих в концепции баз данных являются обобщенные категории «данные» и «модель данных».

Понятие **«данные»** в концепции баз данных — это набор конкретных значений, параметров, характеризующих объект, условие, ситуацию или любые другие факторы. Примеры данных: Петров Николай Степанович, $30 и т. д. Данные не обладают определенной структурой, данные становятся информацией тогда, когда пользователь задает им определенную структуру, т. е. осознает их смысловое содержание. Ядром любой базы данных является **модель данных**. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

**Модель данных** - это совокупность структур данных и операций их обработки.

**1. Файловая модель** была первой моделью, используемой при разработке информационных систем. Точнее модель, как таковая, отсутствовала. Можно сказать, что файловая модель – это модель без СУБД. В файловых системах реализуется модель, при которой внутримашинная база представляет собой совокупность независимых, не связанных между собой файлов из однотипных записей с линейной (одноуровневой) структурой. Файл в этих системах является множеством одинаковых по структуре экземпляров записей со значениями в отдельных полях. Перечислим основные недостатки файловой модели:

* *дублирование данных* в различных отделах из-за децентрализованного хранения;
* *жесткая связь данных и прикладных программ.* Поскольку алгоритм управления такой базой данных был полностью заложен в программном обеспечении информационной системы, то при необходимости изменить структуру данных каждый раз приходилось вносить изменения и в программное обеспечение.
* *необходимость разработки структуры базы данных* каждый раз при разработке информационной системы, что требовало определенных усилий и времени.
* *постоянная необходимость преобразования данных из одного формата в другой*. Часто даже в одной фирме создавалось несколько информационных систем (для каждого отдела, а зачастую для каждого рода деятельности), каждая из которых оперировала своими структурами данных. Бесконечной головной болью был перенос данных из одной ИС в другую, ведь структуры данных еще и периодически менялись. Так что часть программистов непрерывно писали программы преобразования данных из одного формата в другой.
* *сложности совместимости форматов файлов*, т.к. разные фирмы часто пользовались различными языками программирования, отличающимися друг от друга структурой создаваемых ими файлов.

Эти недостатки способствовали появлению специальных моделей БД. Такие модели являются совокупностью взаимосвязанных объектов. Связь двух объектов отражает их подчиненность.

### 2. Иерархическая модель базы данных

*Иерархические базы данных* - самая ранняя модель представления сложной структуры данных. *Информация* в иерархической базе организована по принципу древовидной структуры, в виде отношений "предок-*потомок*". Каждая *запись* может иметь не более одной родительской записи и несколько подчиненных. Связи записей реализуются в виде физических указателей с одной записи на другую.

Основными понятиями иерархической модели данных являются: *уровень*, *узел* (или *элемент*) и *связь*.

*Иерархические базы данных* графически могут быть представлены как перевернутое *дерево*, состоящее из объектов различных уровней. ***Дерево*** представляет собой иерархию элементов, называемых ***узлами***. Под ***элементами*** понимается совокупность атрибутов, описывающих объекты. Верхний уровень (*корень дерева*) занимает один *объект*, второй - объекты второго уровня и так далее.

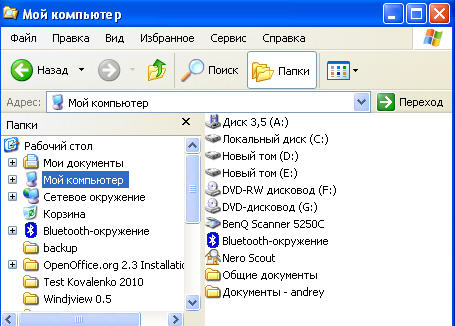
Между объектами существуют связи, каждый *объект* может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (*объект*, более близкий к корню) к потомку (*объект* более низкого уровня), при этом *объект*-предок может не иметь потомков или иметь их несколько, тогда как *объект*-*потомок* обязательно имеет только одного предка. Объекты, имеющие общего предка, называются близнецами.

Такая модель данных обладает следующими свойствами:

* каждый узел связан только с одним вышестоящим узлом, кроме вершины;
* иерархическая модель данных имеет только одну вершину, узел не подчинен более никаким узлам;
* от каждого узла существует единственный путь к вершине;
* связь не может быть установлена между объектами, находящимися через уровень;
* связь между узлами первого уровня не определяется.

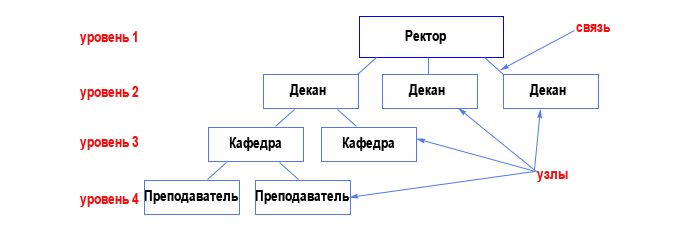
***Примеры***.

1) Иерархической базой данных является Каталог папок *Windows*, с которым можно работать, запустив Проводник. Верхний уровень занимает *папка* *Рабочий стол*. На втором уровне находятся папки Мой *компьютер*, Мои документы, Сетевое окружение и *Корзина*, которые являются потомками папки *Рабочий стол*, а между собой является близнецами. В свою *очередь*, *папка* Мой *компьютер* является предком по отношению к папкам третьего уровня -папкам дисков (*Диск* 3,5(А:), (С:), (D:), (Е:), (F:)) и системным папкам (*сканер*, *bluetooth* и.т.д.) - на [рис. 4.1](http://www.intuit.ru/studies/courses/3439/681/lecture/14023?page=1#image.4.1).



**Рис. 4.1.** Иерархическая база данных Каталог папок Windows

2) Иерархическая структура БД «Университет»



**Иерархическая структура БД «Университет»**

3) Структура организации (директор, заместитель, руководители отделов, сотрудники) (рис.1.3).

Директор

Руководитель отдела продаж

Руководитель PR-отдела

Руководитель отдела обслуживания

Сотрудник

Сотрудник

Сотрудник

Сотрудник

Сотрудник

Рис. 1.3. Иерархическая структура данных

4) реестр Windows.

**Преимущества:**

1. Простота.
2. Минимальный расход памяти.

**Недостатки:**

1. Отсутствие универсальности – не всякую информацию можно выразить в иерархической модели данных.
2. Исключительно навигационный принцип доступа к данным.
3. Доступ к данным только через корневой элемент.

Типичным представителем (наиболее известным и распространенным) является **Information Management System (IMS)** фирмы IBM. Первая версия появилась в **1968** г. До сих пор существуют базы, которые поддерживаются этой СУБД.

**3 Сетевая модель базы данных**

На разработку этого стандарта большое влияние оказал американский ученый Чарльз Бахман. Основные принципы *сетевой модели данных* были разработаны в середине 60-х годов, эталонный вариант *сетевой модели данных* описан в отчетах рабочей группы по языкам баз данных (*COnference* on *DAta SYstem* Languages) **CODASYL** (**1971** г.).

Элементами этой модели являются: *уровень, узел, связь*. Отличия от иерархической модели в том, что элемент одного уровня может быть связан с любым количеством элементов соседнего уровня, и не существует подчиненности уровней друг другу.

Графическое изображение структуры связей сегментов такого типа моделей представляет собой сеть. Сегменты данных в сетевых БД могут иметь множественные связи с сегментами старшего уровня. При этом направление и характер связи в сетевых БД не являются столь очевидными, как в случае иерархических БД. Поэтому имена и направление связей должны идентифицироваться при описании БД.

Таким образом, под ***сетевой СУБД*** понимается система, поддерживающая сетевую организацию: любая запись, называемая записью старшего уровня, может содержать данные, которые относятся к набору других записей, называемых записями подчиненного уровня. Возможно обращение ко всем записям в наборе, начиная с записи старшего уровня. Обращение к набору записей реализуется по указателям.

Типичным представителем является **Integrated Database Management System** (IDMS) появилась в 70-х годах.

Сетевой подход к организации данных является расширением иерархического.

Сетевая БД состоит из набора записей и набора связей между этими записями. На формирование связи особых ограничений не накладывается. В иерархических структурах запись-потомок должна иметь в точности одного предка, а в сетевой структуре данных потомок может иметь любое число предков.

***Пример***. Рассмотрим работу над проектами: можно выделить три вида объектов – сотрудники, проекты, заказчики (рис.1.4).

Сотрудник 1

Сотрудник 2

Сотрудник 3

Сотрудник 4

Проект 1

Проект 2

Проект 3

Заказчик 2

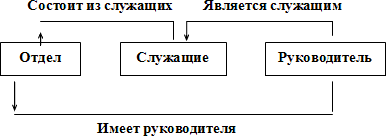
Заказчик 3

Заказчик 4 4

Заказчик 1

Рис. 1.4. Сетевая структура данных

Пример сетевой базы данных



На рисунке показан простой пример схемы сетевой БД.

На этом рисунке показаны три типа записи: Отдел, Служащие и Руководитель и три типа связи: Состоит из служащих, Имеет руководителя и Является служащим.

В типе связи Состоит из служащих типом записи-предком является Отдел, а типом записи-потомком – Служащие (экземпляр этого типа связи связывает экземпляр типа записи Отдел со многими экземплярами типа записи Служащие, соответствующими всем служащим данного отдела).

В типе связи Имеет руководителя типом записи-предком является Отдел, а типом записи-потомком – Руководитель (экземпляр этого типа связи связывает экземпляр типа записи Отдел с одним экземпляром типа записи Руководитель, соответствующим руководителю данного отдела).

Наконец, в типе связи Является служащим типом записи-предком является Руководитель, а типом записи-потомком – Служащие (экземпляр этого типа связи связывает экземпляр типа записи Руководитель с одним экземпляром типа записи Служащие, соответствующим тому служащему, которым является данный руководитель).

К *основным преимуществам* сетевых СУБД относятся следующие:

* обработка больших объемов информации (возможность построения на основе таких СУБД «хранилищ данных»);
* поддержка аналитической обработки данных;
* эффективная реализация обработки данных по показателям затрат памяти и оперативности.

*Недостатки:*

* пользователи сетевых СУБД ограничены связями, определенными для них разработчиками БД-приложений. Подобно иерархическим, сетевые СУБД предполагают разработку БД приложений опытными программистами и системными аналитиками.
* высокая сложность и жесткость схемы БД, построенной на ее основе,
* сложность для понимания и выполнения обработки информации в БД обычным пользователем;
* ослабление контроля целостности связей вследствие допустимости установления произвольных связей между записями.

**4 Реляционная модель данных.**

**Реляционная модель** представления данных была разработана сотрудником фирмы IBM Э. Коддом. Его модель основывается на понятии **«отношения»** (relation). Простейшим примером отношения служит двумерная таблица.

Достоинствами реляционной модели представления данных (по сравнению с иерархической и сетевой моделями) являются ее понятность, простота и удобство практической реализации реляционных баз данных на ЭВМ.

К недостаткам реляционной модели представления данных относятся:

1) отсутствие стандартных средств идентификации отдельных записей;

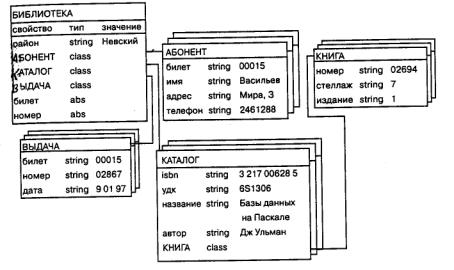
2) сложность описания иерархических и сетевых связей.

Примеры СУБД, построенных на основе реляционной модели данных (Oracle Database, IBM DB2 и Microsoft SQL Server).

**5 Объектно-ориентированная модель данных**

**Объектно-ориентированная модель** представляет структуру, которую можно изобразить графически в виде дерева, узлами которого являются объекты. Каждый объект характеризуется уникальным идентификатором, состоянием и поведением. Состояние объекта определяется множеством значений его атрибутов. Поведение объекта описывают методы, называемые *процедурами*. То есть, составной частью описания объекта являются процедуры, способные производить действия над атрибутами объекта в случае наступления тех или иных событий.

Объекты, схожие по поведению и другим свойствам, объединяются в классы.



Для выполнения действий над объектами применяются объектно-ориентированные механизмы – *наследование, инкапсуляция, полиморфизм*.

Суть **наследования** состоит в том, что на основе существующего класса можно образовать новый класс объектов, который будет наследовать свойства родительского класса.

Доступ к данным осуществляется только лишь в соответствии с правилами поведения объекта, описываемыми методами (инкапсуляция). Инкапсуляция ограничивает область видимости имени свойства пределами того объекта, в котором оно определено.

**Полиморфизм** – способность объектов по-разному реагировать на одно и тоже событие в окружающем мире. Полиморфизм используется для унификации обработки разнородных объектов. Например, метод «Печать результата» может быть определен для многих классов объектов, но работать по-разному, в зависимости от класса, к которому он применяется.

Принцип поиска данных в объектно-ориентированной базе заключается в установлении сходства между объектом, задаваемым пользователем, и объектами, хранящимися в базе данных. Определяемый пользователем объект, называемый объектом-целью, является некоторым подмножеством данных, хранимых в базе.

*Основным достоинством* объектно-ориентированной модели данных в сравнении с реляционной является возможность отображения информации о сложных взаимосвязях объектов. Объектно-ориентированная модель данных позволяет идентифицировать отдельную запись базы данных и определять функции их обработки.

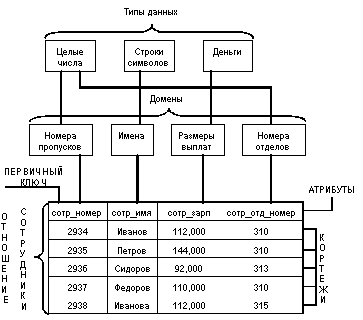
*Недостатками* объектно-ориентированной модели являются высокая понятийная сложность, неудобство обработки данных и низкая скорость выполнения запросов.

К объектно-ориентированным СУБД относятся POET, Jasmine, Versant, O 2, ODB — Jupiter, Iris , Orion, Postgres .

* 1. **Понятие реляционной модели данных**

В основе реляционных систем лежит ***реляционная*** [***модель данных***](https://moodle.vsu.ru/mod/glossary/showentry.php?eid=74&displayformat=dictionary). Принципы реляционной модели были заложены в **1969–1970 гг**. британским ученым **Э.Ф. Коддом** (E. F. Codd), в то время работавшим в корпорации IBM. Будучи математиком по образованию, он привнес в область управления базами данных строгие математические принципы и точность, которых не хватало ранним системам. Хотя реляционный подход утвердился не сразу, можно отметить, что почти все созданные с конца 70-х гг. продукты баз данных основаны именно на реляционном подходе. Подавляющее большинство научных исследований в области баз данных в течение последних 35 лет также проводилось именно в этом направлении.

Основной структурой данных в реляционной модели являются *таблицы*, называемые в реляционной теории *отношениями*. Собственно от термина *отношение* (по-английски relation) и произошло само название модели – *реляционная*. Основными понятиями реляционных баз данных являются *тип данных, домен, атрибут, кортеж, ключ, отношение, схема отношения*.



*Отношение «сотрудники»*

Понятие **тип данных** в реляционной модели данных полностью адекватно понятию типа данных в языках программирования.

**Тип данных** (встречается также термин вид данных) — фундаментальное понятие теории программирования. Тип данных определяет 1) множество значений, 2) набор операций, которые можно применять к таким значениям и, возможно, 3) способ реализации хранения значений и выполнения операций.

В современных реляционных БД допускается хранение символьных, числовых данных, битовых строк, специализированных числовых данных (таких как "деньги"), а также специальных "темпоральных" данных (дата, время, временной интервал). В нашем примере мы имеем дело с данными трех типов: строки символов, целые числа и "деньги".

**Домен** - это семантическое понятие, которое можно рассматривать как подмножество значений некоторого базового типа данных, имеющих определенный смысл. Домен определяется заданием некоторого базового типа данных и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат "истина", то элемент данных является элементом домена.

Домен характеризуется следующими свойствами:

• Домен имеет уникальное имя (в пределах базы данных).

• Домен определен на некотором простом типе данных или на другом домене.

• Домен может иметь некоторое логическое условие, позволяющее описать подмножество данных, допустимых для данного домена.

• Домен несет определенную смысловую нагрузку.

Например, домен D, имеющий смысл "возраст сотрудника" можно описать как следующее подмножество множества натуральных чисел:

Отличие домена от понятия подмножества состоит именно в том, что домен отражает семантику, определенную предметной областью. Может быть несколько доменов, совпадающих как подмножества, но несущие различный смысл. Например, домены "Вес детали" и "Имеющееся количество" можно одинаково описать как множество неотрицательных целых чисел, но смысл этих доменов будет различным, и это будут различные домены.

Основное значение доменов состоит в том, что домены ограничивают сравнения. Некорректно, с логической точки зрения, сравнивать значения из различных доменов, даже если они имеют одинаковый тип.

В основе реляционной модели данных лежит понятие *отношения*, которое задается списком своих элементов и перечислением их значений. Рассмотрим пример, представленный на рисунке отношение «Сотрудники». Налицо определенная структура. Каждый сотрудник имеет свой номер, характеризуется фамилией, размером зарплаты и отделов, в котором работает. Эти сведения легко представляются в виде таблицы. Заголовки колонок таблицы носят название *атрибутов*. Список их имен носит названия *схемы отношения*. Каждый атрибут определяет тип представляемых им данных, который вместе с областью его значений называется *доменом*. Вся таблица целиком называется *отношением*, а каждая строка таблицы носит название *кортежа отношения*. Таким образом, отношение можно представить в виде двумерной таблицы.

Подходы к определению понятия отношения могут быть различными.

Математически *отношение* может быть определено как множество кортежей, являющейся подмножеством декартова произведения фиксированного числа областей (доменов).

В результате получаем, что в каждом кортеже должно быть одинаковое число компонент (атрибутов) и значение каждого из них выбирается из некоторого определенного домена.

Отношение содержит две части: заголовок и тело.

**Заголовок отношения** содержит фиксированное количество атрибутов отношения:

Имена атрибутов должны быть уникальны в пределах отношения. Часто имена атрибутов отношения совпадают с именами соответствующих доменов.

**Тело отношения** содержит множество кортежей отношения.

**Ключ отношения** – это атрибут или набор атрибутов отношения такие, что в любой момент времени в отношении не существует строк, для которых значение или комбинация значений ключевых атрибутов являются одинаковыми. Ключ, таким образом, является уникальным идентификатором кортежей отношения.

**Кортеж**, соответствующий данной схеме отношения, - это множество пар {имя атрибута, значение атрибута}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения. "Значение" является допустимым значением домена данного атрибута (или типа данных, если понятие домена не поддерживается).

(<A1:Val1>, <A2:Val2>,... <An:Valn>)

таких что значение Vali атрибута Ai принадлежит домену Di

Отношение обычно записывается в виде: R(<A1:D1>, <A2:D2>,... <An:Dn>), или короче R(A1, A2, . . ., An), или просто R

Число атрибутов в отношении называют **степенью (или - арностью) отношения**.

Число кортежей отношения называют **мощностью отношения**.

**Схема отношения** - это именованное множество пар {имя атрибута, имя домена (или типа, если понятие домена не поддерживается)}.

**Схема реляционной БД** (в структурном смысле) - это набор именованных схем отношений.

**Реляционной базой данных** называется набор отношений.

Термины, которыми оперирует реляционная модель данных, имеют соответствующие "табличные" синонимы (см. таблица 1):

Таблица 1

|  |  |
| --- | --- |
| Реляционный термин | Соответствующий "табличный" термин |
| База данных | Набор таблиц |
| Схема базы данных | Набор заголовков таблиц |
| Отношение | Таблица |
| Заголовок отношения | Заголовок таблицы |
| Тело отношения | Тело таблицы |
| Атрибут отношения | Наименование столбца таблицы |
| Кортеж отношения | Строка таблицы |
| Степень (-арность) отношения | Количество столбцов таблицы |
| Мощность отношения | Количество строк таблицы |
| Домены и типы данных | Типы данные в ячейках таблицы |

* 1. **12 правил Кодда**

В **1985** году в двух статьях в журнале **Computer World Э.Ф. Кодд** сформулировал правила, которым должны соответствовать настоящие реляционные базы данных. Всего правил было двенадцать. Несколько позже Кодд сформулировал 13-е правило и присвоил ему номер 0.

В действительности правила столь строго императивны, что все популярные, так называемые “реляционные” СУБД не соответствуют многим критериям, а сами критерии не являются строго логичными.

Начнем рассмотрение правил с этого последнего или нулевого.

**Правило 0. Основное правило (Foundation Rule)**

Реляционная СУБД должна быть способна полностью управлять базой данных, используя связи между данными. Чтобы быть реляционной системой управления базами данных (СУБД), система должна использовать исключительно свои реляционные возможности для управления базой данных.

**Правило 1. Явное представление данных (The Information Rule)**

Информация должна быть представлена в виде данных, хранящихся в ячейках. Данные, хранящиеся в ячейках, должны быть атомарны. Порядок строк в реляционной таблице не должен влиять на смысл данных.

**Правило 2. Гарантированный доступ к данным (Guaranteed Access Rule)**

Логический доступ ко всем и каждому элементу данных в реляционной базе данных обеспечивается комбинацией имени таблицы, имени столбца и значением первичного ключа. Это требование предполагает:

o Уникальность имени таблицы в базе данных.

o Уникальность имени столбца в таблице.

o Уникальность первичного ключа, по крайней мере, в пределах одной таблицы.

**Правило 3. Полная обработка неизвестных значений (Systematic Treatment of Null Values):**

Неизвестные значения NULL, отличные от любого известного значения, должны поддерживаться для всех типов данных при выполнении любых операций. Например, для числовых данных неизвестные значения не должны рассматриваться как нули, а для символьных данных — как пустые строки.

**Правило 4. Доступ к словарю данных в терминах реляционной модели (Active On-Line Catalog Based on the Relational Model)**

**Словарь данных** (СД) представляет собой подсистему БнД, предназначенную для централизованного хранения информации о структурах данных, взаимосвязях файлов БД друг с другом, типах данных и форматах их представления, принадлежности данных пользователям, кодах защиты и разграничения доступа и т.п.

Словарь данных должен сохраняться в форме реляционных таблиц, и СУБД должна поддерживать доступ к нему при помощи стандартных языковых средств, тех же самых, которые используются для работы с реляционными таблицами, содержащими пользовательские данные.

**Правило 5. Полнота подмножества языка (Comprehensive Data Sublanguage Rule)**

Реляционная система может поддерживать несколько языков. Однако должен существовать по крайней мере один язык, который позволял бы выражать следующие конструкции:

1) определение данных;

2) определение представлений;

3) операторы манипулирования данными;

4) ограничения целостности;

5) авторизация пользователей;

6) поэтапная организация транзакций.

**Правило 6. Возможность модификации представлений (View Updating Rule)**

Каждое представление должно поддерживать все операции манипулирования данными, которые поддерживают реляционные таблицы: операции выборки, вставки, модификации и удаления данных.

**Правило 7. Наличие высокоуровневых операций управления данными (High-Level Insert, Update, and Delete)**

Операции вставки, модификации и удаления данных должны поддерживаться не только по отношению к одной строке реляционной таблицы, но по отношению к любому множеству строк.

**Правило 8. Физическая независимость данных (Physical Data Independence)**

Приложения не должны зависеть от используемых способов хранения данных на носителях, от аппаратного обеспечения компьютеров, на которых находится реляционная база данных.

**Правило 9. Логическая независимость данных (Logical Data Independence)**

Представление данных в приложении не должно зависеть от структуры реляционных таблиц. Если в процессе нормализации одна реляционная таблица разделяется на две, представление должно обеспечить объединение этих данных, чтобы изменение структуры реляционных таблиц не сказывалось на работе приложений.

**Правило 10. Независимость ограничений целостности (Integrity Independence)**

Ограничения целостности данных должны определяться на языке реляционных данных и храниться в базе данных, а не в прикладных программах.

**Правило 11. Независимость от распределения данных (Distribution Independence)**

База данных может быть распределённой, может находиться на нескольких компьютерах, и это не должно оказывать влияние на приложения. Перенос базы данных на другой компьютер не должен оказывать влияния на приложения.

**Правило 12. Согласование языковых уровней (The Nonsubversion Rule)**

Не должно быть иного средства доступа к данным, отличного от стандартного языка для работы с данными. Если используется низкоуровневый язык доступа к данным, он не должен игнорировать правила безопасности и правила целостности, которые поддерживаются языком более высокого уровня.

**2. Основы взаимодействия с Microsoft SQL Server**

**2.1 Версии и редакции Microsoft SQL Server**

**Microsoft SQL Server** — система управления реляционными базами данных (РСУБД), разработанная корпорацией **Microsoft**. Основной используемый язык запросов — **Transact-SQL**, создан совместно Microsoft и Sybase. **Transact-SQL** является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Исходный код MS SQL Server (до версии 7.0) основывался на коде Sybase SQL Server, и это позволило Microsoft выйти на рынок баз данных для предприятий, где конкурировали Oracle, IBM, и, позже, сама Sybase. **Microsoft, Sybase и Ashton-Tate** первоначально объединились для создания и выпуска на рынок первой версии программы, получившей название **SQL Server 1.0** для OS/2 (около 1989 года), которая фактически была эквивалентом Sybase SQL Server 3.0 для Unix, VMS и др. Microsoft SQL Server 4.2 был выпущен в 1992 году и входил в состав операционной системы Microsoft OS/2 версии 1.3. Официальный релиз Microsoft SQL Server версии 4.21 для ОС Windows NT состоялся одновременно с релизом самой Windows NT (версии 3.1). Microsoft SQL Server 6.0 был первой версией SQL Server, созданной исключительно для архитектуры NT и без участия в процессе разработки Sybase.

К тому времени, как вышла на рынок ОС Windows NT, Sybase и Microsoft разошлись и следовали собственным моделям программного продукта и маркетинговым схемам. Microsoft добивалась исключительных прав на все версии SQL Server для Windows. Позже Sybase изменила название своего продукта на Adaptive Server Enterprise во избежание путаницы с Microsoft SQL Server. До 1994 года Microsoft получила от Sybase три уведомления об авторских правах как намёк на происхождение Microsoft SQL Server.

После разделения компании сделали несколько самостоятельных релизов программ. SQL Server 7.0 был первым сервером баз данных с настоящим пользовательским графическим интерфейсом администрирования. Для устранения претензий со стороны Sybase в нарушении авторских прав, весь наследуемый код в седьмой версии был переписан.

|  |
| --- |
| **История выпусков SQL Server** |
| **Версия** | **Год** | **Название** | **Кодовое имя** |
| 1.0 ([OS/2](https://ru.wikipedia.org/wiki/OS/2)) | 1989 | SQL Server 1.0 (16 bit) | Ashton-Tate / MS SQL Server |
| 1.1 ([OS/2](https://ru.wikipedia.org/wiki/OS/2)) | 1991 | SQL Server 1.1 (16 bit) | - |
| 4.21 ([WinNT](https://ru.wikipedia.org/wiki/Windows_NT" \o "Windows NT)) | 1993 | SQL Server 4.21 | SQLNT |
| 6.0 | 1995 | SQL Server 6.0 | SQL95 |
| 6.5 | 1996 | SQL Server 6.5 | Hydra |
| 7.0 | 1998 | SQL Server 7.0 | Sphinx |
| - | 1999 | SQL Server 7.0 [OLAP](https://ru.wikipedia.org/wiki/OLAP) Tools | Palato mania |
| 8.0 | 2000 | SQL Server 2000 | Shiloh |
| 8.0 | 2003 | SQL Server 2000 64-bit | Liberty |
| 9.0 | 2005 | SQL Server 2005 | Yukon |
| 10.0 | 2008 | SQL Server 2008 | Katmai |
| 10.25 | 2010 | Azure SQL DB | Cloud Database or CloudDB |
| 10.50 | 2010 | SQL Server 2008 R2 | Kilimanjaro (aka KJ) |
| 11.0 | 2012 | SQL Server 2012 | Denali |
| 12.0 | 2014 | SQL Server In-Memory OLTP | Hekaton |
| 13.0 | 2016 | SQL Server 2016 |  |

Сравнение версий SQL Server

| **Функции** |  | **SQL Server 2016** | **SQL Server 2014** | **SQL Server 2012** | **SQL Server 2008 R2** |
| --- | --- | --- | --- | --- | --- |
| **Производительность** | Выполняющаяся в памяти OLTP\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |
|  | Хранение столбцов в памяти\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |
|  | Операционная аналитика в реальном времени\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
|  | Регулятор ресурсов\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |
|  | Хранение запросов | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
| **Доступность** | AlwaysOn\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |
|  | Расширенная поддержка визуализации и динамическая миграция | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |
| **Безопасность** | Постоянное шифрование\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
|  | Прозрачное шифрование данных\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |
|  | Безопасность на уровне строк | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
|  | Динамическая маскировка данных | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
|  | Поддержка шифрования резервного копирования | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |
|  | Детальный аудит | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |
|  | Разделение обязанностей | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |
| **Программируемость** | Поддержка JSON | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
|  | Запросы PolyBase для данных Hadoop\*\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
|  | Temporal | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
| **Готовность к использованию в облаке** | База данных Stretch SQL Server | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
|  | Архивирование в Azure | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |
|  | Аварийное восстановление в Azure\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |
|  | Оптимизированные образы виртуальных машин в коллекции Azure | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |
| **Управление** | Распределенное воспроизведение | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |
|  | Управление на основе политик | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |
| **Бизнес-аналитика** | Усовершенствованные отчеты | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
|  | Мобильная бизнес-аналитика\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
|  | Сервисы интеграции, управляемые в качестве сервера | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |
|  | Закрепление отчетов в Power BI | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
|  | Многомерные семантические модели | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |
|  | Усовершенствованные табличные семантические модели бизнес-аналитики\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
|  | Сервисы основных данных\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |
|  | Сервисы качества данных\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |
| **Расширенная аналитика** | Расширенная аналитика в базе данных с помощью служб R Services | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |
|  | Многопоточная обработка запросов R и потоковая обработка в памяти\* | https://www.microsoft.com/ru-ru/server-cloud/Images/shared/TableIcon.png |  |  |  |

Существуют следующие основные выпуски MS SQL Server:

|  |  |
| --- | --- |
| Enterprise Edition  (64-разрядная и 32-разрядная версия) | Выпуск SQL Server Enterprise Edition является предложением высшего класса, обеспечивающим полный набор возможностей ЦОД с исключительно высокой производительностью, неограниченными возможностями виртуализации и исчерпывающими средствами бизнес-аналитики, что позволяет добиться высокого уровня обслуживания важнейших рабочих нагрузок и предоставить конечным пользователям доступ к анализу данных.  Это серверная редакция, которая устанавливается на крупных предприятиях. Обеспечивает поддержку многопроцессорной конфигурации и различных платформ аппаратного обеспечения. Этот вариант наилучшим образом подходит для реализации Internet-решений. |
| Business Intelligence (64-разрядная версия и 32-разрядная версия) | Выпуск SQL Server Business Intelligence представляет собой комплексную платформу, которая дает организациям возможность строить и развертывать надежные, масштабируемые и управляемые решения бизнес-аналитики. В него включены функции просмотра и представления данных в браузере, мощные возможности комбинирования данных и расширенное управление интеграцией. |
| Standard Edition  (64-разрядная и 32-разрядная версия) | Выпуск SQL Server Standard обеспечивает основные функции управления данными и предоставляет базу данных бизнес-аналитики для приложений, работающих в отделах и небольших организациях. Поддерживаются распространенные средства разработки в локальных системах и вычислительных облаках, что делает возможным эффективное управление базами данных с минимальными затратами ИТ-ресурсов. |

Специализированные выпуски SQL Server предназначены для типичных видов рабочей нагрузки. Специальные выпуски SQL Server:

|  |  |
| --- | --- |
| Web Edition  (64-разрядная и 32-разрядная версия) | Выпуск SQL Server Web Edition — это вариант с низкой совокупной стоимостью владения, предназначенный для размещения веб-сайтов и дополнительных веб-услуг, который по доступной цене обеспечивает масштабируемость и функции управления для небольших и крупномасштабных веб-проектов. |

Дополнительные выпуски SQL Server предназначены для специальных задач и предлагаются бесплатно или за минимальную плату. В следующей таблице приведено описание дополнительных выпусков SQL Server:

|  |  |
| --- | --- |
| Developer Edition (64-разрядная и 32-разрядная версия) | Выпуск SQL Server Developer Edition позволяет разработчикам создавать приложения любого типа на базе SQL Server. Он включает все функциональные возможности выпуска Enterprise Edition, однако лицензируется как система для разработки и тестирования, а не для применения в качестве рабочего сервера. Выпуск SQL Server Developer Edition является идеальным выбором для тех, кто создает и тестирует приложения. |
| Выпуски Express (64-разрядная и 32-разрядная версия) | Выпуск SQL Server Express является бесплатной базой данных начального уровня и идеально подходит для обучения, а также для создания управляемых данными приложений, работающих на рабочих станциях и небольших серверах. Этот выпуск — лучший выбор для независимых поставщиков программного обеспечения, непрофессиональных разработчиков и любителей, создающих клиентские приложения. Если необходимы дополнительные функции базы данных, выпуск SQL Server Express можно легко обновить до версий SQL Server более высокого класса. Новой возможностью версии SQL Server 2012 является SQL Server Express LocalDB. Это облегченная версия Express, которая имеет все программные функции, запускается в пользовательском режиме, быстро устанавливается, не требует настройки и имеет низкие системные требования. |

* 1. **Инсталляция Microsoft SQL Server**

Произвести установку Microsoft SQL Server дома.

* 1. **Инструменты управления и утилиты MS SQL Server**

**Утилита SQL Server Management Studio**

Подавляющую массу задач администрирования SQL Server можно выполнить в графической утилите **SQL Server Management Studio**. В ней можно создавать базы данных и все ассоциированные с ними объекты (таблицы, представления, хранимые процедуры и др.). Здесь вы можете выполнить последовательности инструкций Transact-SQL (запросы). В этой утилите можно выполнить типовые задачи обслуживания баз данных, такие как резервирование и восстановление. Здесь можно настраивать систему безопасности базы данных и сервера, просматривать журнал ошибок и многое другое.

Для запуска Management Studio в меню «Пуск» операционной системы выберите пункт «Microsoft SQL Server 2012\Среда SQL Server Management Studio». Когда откроется окно программы, вас попросят подключиться к какому либо серверу баз данных SQL Server (рисунок 1.1).

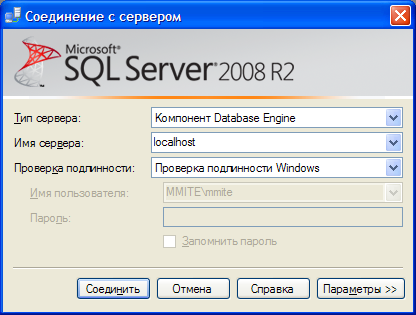


Рисунок 1.1 – Окно подключения к серверу БД MS SQL Server

**Подключение к серверу**

В окне «Соединение с сервером» необходимо указать следующую информацию:

* *Тип сервера*. Здесь следует выбрать, к какой именно службе необходимо подключится. Оставьте вариант «Компонент Database Engine».
* *Имя сервера*. Позволяет указать, к какому серверу будет осуществляться подключение. По умолчанию имя SQL Server совпадает с именем компьютера. Выберите ваш локальный компьютер.
* *Проверка подлинности*. Способ аутентификации, можно выбрать «Проверка подлинности Windows» или «Проверка подлинности SQL Server». Первый способ использует учетную запись, под которой текущий пользователь осуществил вход в Windows. Вариант SQL Server использует свою собственную систему безопасности. Оставьте вариант проверки подлинности Windows.

После подключения экземпляр сервера будет отображаться на панели «Обозреватель объектов» (рисунок 1.2).

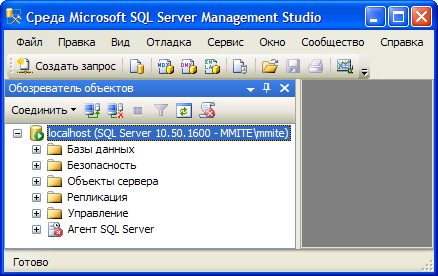


Рисунок 1.2 – Панель «Обозреватель объектов»

Окно Management Studio имеет следующую структуру:

* *Оконное меню* – содержит полный набор команд для управления сервером и выполнения различных операций.
* *Панель инструментов* – содержит кнопки для выполнения наиболее часто производимых операций. Внешний вид данной панели зависит от выполняемой операции.
* *Панель «Обозреватель объектов»*. Это панель с древовидной структурой, отображающая все объекты сервера, а также позволяющая производить различные операции, как с самим сервером, так и с его базами данных и их объектами. Обозреватель объектов является основным инструментом для разработки.
* *Рабочая область*. В рабочей области производятся все действия с базой данных, а также отображается её содержимое.

**Структура базы данных MS SQL Server**

Каждая база данных SQL Server состоит из набора файлов операционной системы. Эти файлы могут группироваться в группы файлов, что облегчает их администрирование, помогает в размещении данных и повышает производительность. Файл базы данных может быть либо файлом данных, либо файлом журнала. Файлы данных служат для хранения данных и объектов, таких как таблицы, индексы, представления, триггеры и хранимые процедуры. Имеются два типа файлов данных: первичные и вторичные. Файлы журналов служат только для хранения информации из журналов транзакций. Место на диске, отводимое для файлов журналов всегда должно администрироваться отдельно от места, отводимого для данных, и никогда не должно быть частью файла данных.

Каждая база данных должна создаваться хотя бы с одним файлов данных и с одним файлом журнала; файлы не могут быть использованы более чем в одной базе данных – т.е., базы данных не могут разделять файлы (или использовать их совместно).

*Первичные файлы данных* содержат всю информацию для запуска базы данных и ее системных таблиц и объектов. Они указывают на другие файлы, созданные в базе данных. Они могут также содержать таблицы и объекты, задаваемые пользователем, хотя это и необязательно. Каждая база данных может иметь только один первичный файл. Для этих файлов рекомендуется применять расширение .mdf.

*Вторичные файлы данных* не являются обязательными. Они могут хранить данные и объекты, которые отсутствуют в первичном файле. База данных может вообще не иметь ни одного вторичного файла (если все ее данные хранятся в первичном файле). Можно иметь ноль, один или несколько вторичных файлов. Для некоторых баз данных требуется иметь несколько вторичных файлов, чтобы размещать данные по нескольким отдельным дискам. Для этих файлов рекомендуется применять расширение .ndf.

*Файлы журналов транзакций* хранят всю информацию из журнала транзакций, служащую для восстановления базы данных. Каждая база данных должна меть хотя бы один файл журнала, а может иметь несколько файлов журналов. Для этих файлов рекомендуется применять расширение .ldf.

**База данных SQL Server** – это контейнер объектов, предназначенных для хранения, записи и извлечения данных в безопасном режиме – содержит следующие компоненты:

• *Таблицы*. Это объекты, внутри которых хранятся данные БД. Информация хранится в таблицах в форме строк и столбцов. Столбцы таблицы определяют простейшие элементы информации, определяют назначение и тип данных, которые в нем хранятся и объем элемента этих данных. Таблица должна состоять минимум из одного столбца. Строка образуется одним элементом информации от каждого столбца. Таблица может содержат любое число строк – оно ограничивается местом свободного пространства на диске. Строка определяет блок информации, состоящий из элементов, хранящихся в столбцах. Другое название строки – запись.

• *Хранимые процедуры*. Программный код, состоящий из одного или нескольких операторов T-SQL, которые уже откомпилированы, хранится в БД и может быть запущена в любой момент.

• *Индексы* можно рассматривать как некие списки с информацией о расположении и упорядочении данных. Используются для ускорения извлечения данных из таблиц БД.

• *Представления*. Особый способ просмотра данных, хранящихся в БД.

В SQL Server есть несколько БД, создаваемых в процессе его установки. К системным БД относят: master (главная), tempdb (временная), model (модель) и msdb.

Каждая из этих системных баз данных имеет свои собственные первичный файл данных и файл журнала. Системные базы данных хранятся в папке для хранения системных файлов.

* *master*. Главная служебная база данных всего сервера. В ней хранится общая служебная информация сервера: настройки его работы, список баз данных на сервере с информацией о настройках каждой базы данных и ее файлах, информация об учетных записях пользователей, серверных ролях и т.п.
* *msdb*. Эта база данных в основном используется для хранения информации службы SQL Server Agent (пакетных заданий, предупреждений и т.п.), но в нее записывается и другая служебная информация (например, история резервного копирования).
* *model*. Эта база данных является шаблоном для создания новых баз данных в SQL Server. Если внести в нее изменения, например, создать набор таблиц, то эти таблицы будут присутствовать во всех создаваемых базах данных.
* *tempdb*. Эта база данных предназначена для временных таблиц и хранимых процедур, создаваемых пользователями и самим SQL Server. Эта база данных создается заново при каждом запуске SQL Server.

**Создание пользовательских баз данных**

Существует два способа создания баз данных:

* графически с помощью SQL Server Management Studio
* посредством кода Transact-SQL

**Создание баз данных в SQL Server Management Studio**

Использование данной утилиты является самым простым способом создания базы данных. Создадим базу данных Sales (Продажи), которую позже заполним таблицами, представлениями и другими объектами, предназначенными для отдела продаж.

1. В окне «Обозреватель объектов» найдите и раскройте папку «Базы данных». Щелкните на ней правой кнопкой мыши и выберите команду «Создать базу данных…».
2. В открывшемся диалоговом окне «Создание базы данных» на странице «Общие» введите следующую информацию (рисунок 1.3):

Имя базы данных: Sales

Владелец: sa

В таблице «Файлы базы данных» измените путь к файлам данных и журнала на ваш каталог.

Для всех остальных параметров оставьте значения по умолчанию.

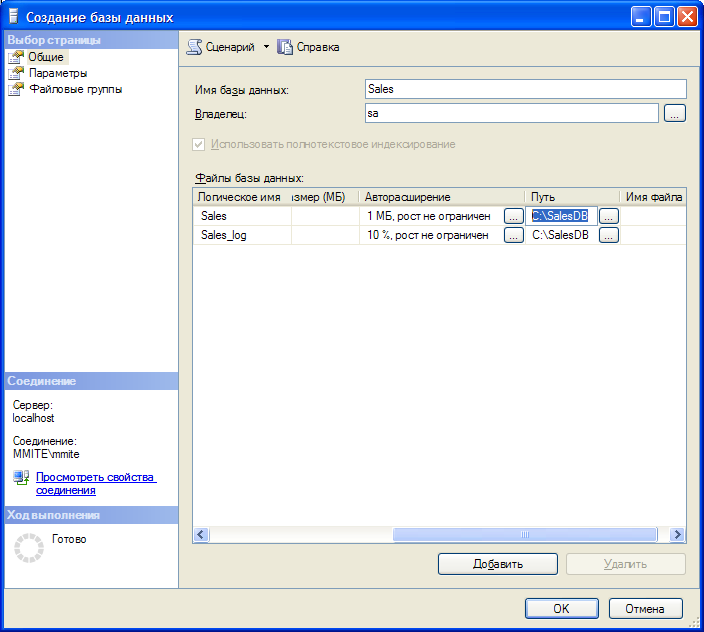


Рисунок 1.3 – Создание БД средствами SQL Server Management Studio

1. Для создания базы данных щелкните «OK». Вы должны увидеть свою новую базу данных в окне «Обозреватель объектов».

**Создание баз данных с помощью Transact-SQL**

Для программного создания базы данных (например, в программе установки приложения) используется инструкция CREATE DATABASE языка T-SQL (сокращенная форма от Transact-SQL). Данная инструкция может включать в себя множество опций, определяющих различные параметры новой базы данных.

Сценарий создания новой базы данных может быть сгенерирован на основе уже существующей базы данных. Для этого в SQL Server Management Studio в контекстном меню узла «Sales» выберите команду «Создать сценарий для базы данных – Используя CREATE – Буфер обмена». В результате в буфер обмена будет сохранен текст запроса на создание новой базы данных с параметрами, указанными при создании базы данных Sales в Management Studio.

Для проверки работоспособности сгенерированного запроса на создание базы данных удалим базу данных Sales. В контекстном меню базы данных выберите команду «Удалить» и в появившемся диалоговом окне нажмите кнопку «OK». База данных со всеми файлами должна исчезнуть.

Чтобы воспользоваться сгенерированным заранее запросом на создание базы данных выполните следующие шаги:

1. В контекстном меню базы Sales выберите команду «Создать запрос» или щелкните соответствующую кнопку на панели инструментов .
2. В открывшемся окне редактора SQL вставьте из буфера обмена сгенерированный запрос.
3. Для запуска запроса на выполнение щелкните кнопку  на панели инструментов или нажмите клавишу F5.
4. Обновите содержимое дерева обозревателя объектов командой «Обновить» из контекстного меню узла «Базы данных». База данных Sales должна вновь появиться в списке доступных.

При необходимости перед выполнением текст запроса может быть изменен в этом же окне.

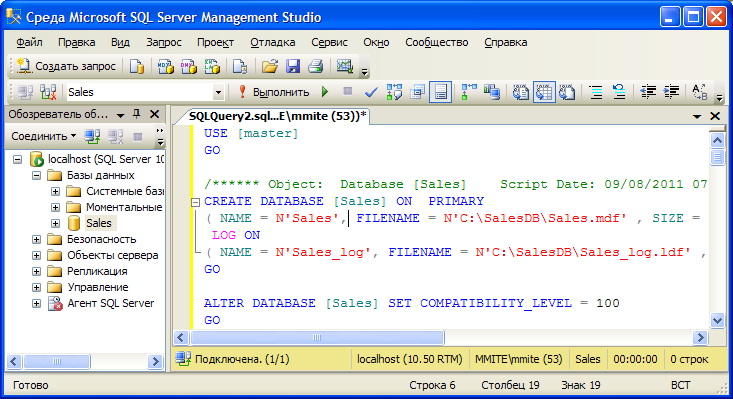


Рисунок 1.4 – Пример создания БД с помощью сгенерированного запроса

Создавать или изменять свои базы данных можно при помощи операторов T-SQL, не пользуясь графическим пользовательским интерфейсом (GUI). Можно создавать свои собственные сценарии создания баз данных.

В приведенном ниже тексте дан пример создания базы данных с именем MyDB, содержащей: первичный файл данных (MyDB\_root); один вторичный файл данных (MyDB\_data1), который по умолчанию останется в первичной группе файлов и один файл журнала транзакций (Log\_data1). Наберите данный текст в панели запросов и запустите этот сценарий нажатием клавиш *F5* или *Ctrl-E*

CREATE DATABASE MyDB

ON

(NAME="MyDBroot", -- Первичный файл данных

FILENAME="C:\Databases\MyDB\mydbroot.mdf",

SIZE=8MB,

MAXSIZE=100MB,

FILEGROWTH=1MB),

(NAME="MyDBdata1", -- Вторичный файл данных

FILENAME="C:\Databases\MyDB\mydbdata1.ndf",

SIZE=10MB,

MAXSIZE=150MB,

FILEGROWTH=1MB)

LOG ON

(NAME="Logdata1", -- Файл журнала

FILENAME="B:\Databases\MyDB\logdata1.ldf",

SIZE=10MB,

MAXSIZE=150MB,

FILEGROWTH=1MB)

Описание параметров оператора CREATE DATABASE:

* ON – ключевое слово, указывает службе SQL Server, что в команде должны быть заданы расположение файлов данных, их имена, объем и величина объема приращения;
* NAME – логическое имя файла, по которому происходит обращение к этому файлу со стороны SQL Server;
* FILENAME – физическое имя файла с указанием полного пути с обязательным указанием расширения файла;
* SIZE – исходный объем в мегабайтах. Параметр не обязателен, его можно опустить;
* FILEGROWTH – приращение объема файла после его заполнения. Приращение можно указать в мегабайтах или процентах от текущего объема;
* LOG ON – ключевое слово, указывает службе SQL Server, что в команде должны быть заданы расположение файлов журнала, их имена, объем и величина объема приращения;
* COLLETE – необязательный параметр, указывает схему сортировки, применяемой в БД.

## *Удаление баз данных*

Базы данных можно удалять как при помощи SQL Server Management Studio, так и командами T-SQL.

Применение SQL Server Management Studio

В Обозревателе объектов разверните Базы данных, щелкните правой кнопкой мыши удаляемую базу данных и затем нажмите кнопку Удалить.

Подтвердите, что выбрана верная база данных, а затем нажмите кнопку ОК.

**Примечание.** Создайте резервную копию базы данных master после удаления любой базы данных. Если база данных master должна быть восстановлена, любая база данных, удаленная с момента создания последней резервной копии базы данных master, останется в представлениях системного каталога и может вызвать сообщения об ошибках.

**Переименование базы данных**

Для переименования базы данных применяется T-SQL команда ALTER DATABASE.

USE master;

GO

ALTER DATABASE MyDB

Modify Name = MyDataBase;

GO

Применение команд SQL

Помните, что удаление базы данных является неотменяемым действием. Для удаления базы данных применяется T-SQL команда DROP DATABASE. Ниже приведены примеры команд, которые удалят базу данных MyDB и все ее файлы.

Use master

--для запуска команды DROP DATABASE вы должны применять базу данных master

GO

DROP DATABASE MyDataBase

--Единственным параметром этой команды является имя удаляемой базы

GO

После удаления базы данных следует обновить резервную копию базы данных master, чтобы в ней содержалась свежая информация о пользовательских базах данных и не содержалась информация о только что удаленной базе данных. Также обратите внимание, что база данных не может быть удалена, когда к ней имеют доступ пользователи. Перед удалением базы данных нужно отсоединить от нее всех пользователей.

**Отсоединение и присоединение базы данных**

Для переноса базы данных на другой сервер необходимо отсоединить ее от текущего сервера. Для этого в контекстном меню базы данных Sales выберите команду «Задачи - Отсоединить…». В диалоговом окне «Отсоединение базы данных» нажмите кнопку «OK» и убедитесь, что Sales исчезла из списка баз данных в дереве обозревателя объектов. Теперь файлы базы данных могут быть перенесены на другой сервер.

Для присоединения базы данных к серверу выберите в контекстном меню узла «Базы данных» команду «Присоединить…». В диалоговом окне «Присоединение базы данных» с помощью кнопки «Добавить…» выберите созданный на предыдущих этапах файл Sales.mdf (ldf файл будет определен системой автоматически), измените владельца на sa и нажмите кнопку «OK» (рисунок 1.5). База данных Sales должна появиться в списке дерева обозревателя объектов.

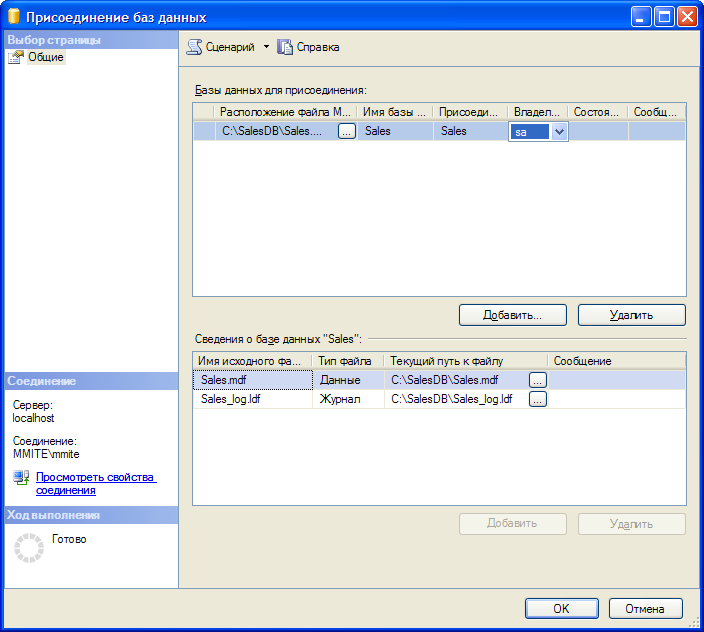


Рисунок 1.5 – Присоединение баз данных

CREATE DATABASE MyDB

ON

(NAME="MyDBroot", -- Первичный файл данных

FILENAME="C:\Databases\MyDB\mydbroot.mdf",

SIZE=8MB,

MAXSIZE=100MB,

FILEGROWTH=1MB),

(NAME="MyDBdata1", -- Вторичный файл данных

FILENAME="C:\Databases\MyDB\mydbdata1.ndf",

SIZE=10MB,

MAXSIZE=150MB,

FILEGROWTH=1MB)

LOG ON

(NAME="Logdata1", -- Файл журнала

FILENAME="B:\Databases\MyDB\logdata1.ldf",

SIZE=10MB,

MAXSIZE=150MB,

FILEGROWTH=1MB)

Для переименования базы данных применяется T-SQL команда ALTER DATABASE.

USE master;

GO

ALTER DATABASE MyDB

Modify Name = MyDataBase;

GO

Use master

--для запуска команды DROP DATABASE вы должны применять базу данных master

GO

DROP DATABASE MyDataBase

--Единственным параметром этой команды является имя удаляемой базы

GO

**Пример. Создание базы данных с удалением существующей**

**"старой" базы данных**

USE master;

GO

IF DB\_ID('MyDataBase') IS NOT NULL

DROP DATABASE MyDataBase;

GO

CREATE DATABASE MyDataBase;

GO

**Создание базы данных BestDatabase**

USE master;

GO

IF DB\_ID('MyDataBase') IS NOT NULL

DROP DATABASE MyDataBase;

GO

CREATE DATABASE MyDataBase

ON PRIMARY (NAME = MyDataBase\_dat,

FILENAME = 'C:\Databases\Winner.mdf',

SIZE = 5 MB,

MAXSIZE = UNLIMITED,

FILEGROWTH = 1 MB)

LOG ON (NAME = MyDataBase\_log,

FILENAME = 'C:\Databases\Winner.ldf',

SIZE = 2 MB,

MAXSIZE = 30 MB,

FILEGROWTH = 1 MB);

GO

**Создание многофайловой базы данных**

USE master;

GO

IF DB\_ID('Multy') IS NOT NULL

DROP DATABASE Multy;

GO

CREATE DATABASE Multy

ON

PRIMARY

( NAME = Multy1,

FILENAME = 'C:\Databases\Multy\Multy1.mdf'),

( NAME = Multy2,

FILENAME = 'C:\Databases\Multy\Multy2.ndf')

LOG ON

( NAME = MultyL1,

FILENAME = 'C:\Databases\Multy\MultyL1.ldf'),

( NAME = MultyL2,

FILENAME = 'C:\Databases\Multy\MultyL2.ldf');

GO

**Создание базы данных с двумя файловыми группами**

USE master;

GO

IF DB\_ID('MultyGroup') IS NOT NULL

DROP DATABASE MultyGroup;

GO

CREATE DATABASE MultyGroup

ON

PRIMARY

( NAME = MultyGroup1,

FILENAME = 'C:\Databases\Multy\MultyGroup1.mdf'),

( NAME = MultyGroup2,

FILENAME = 'C:\Databases\Multy\MultyGroup2.ndf'),

FILEGROUP MultyGroup2

( NAME = MultyGroup3,

FILENAME = 'C:\Databases\Multy\MultyGroup3.ndf'),

( NAME = MultyGroup4,

FILENAME = 'C:\Databases\Multy\MultyGroup4.ndf')

LOG ON

( NAME = MultyGroupLog1,

FILENAME = 'C:\Databases\Multy\MultyGroupLog1.ldf'),

( NAME = MultyGroupLog2,

FILENAME = 'C:\Databases\Multy\MultyGroupLog2.ldf');

GO

**Изменение размера файла данных**

USE master;

GO

ALTER DATABASE MyDB

MODIFY FILE (NAME = MyDBroot,

SIZE = 30 MB,

MAXSIZE = 100 MB);

GO

**Внесение изменений в файловые группы базы данных (для базы данных MultyGroup переименовывается существующая файловая группа MultyGroup2 в MultyGroup0 и добавляется еще одна с именем только что измененной группы MultyGroup2)**

USE master;

GO

ALTER DATABASE MultyGroup

MODIFY FILEGROUP MultyGroup2

NAME = MultyGroup0;

GO

ALTER DATABASE MultyGroup

ADD FILEGROUP MultyGroup2;

GO