Базы данных

Общие сведения о базах данных

Первые компьютеры (англ. computer – вычислитель), как это ясно из названия, были ориентированы только для решения вычислительных задач (например, в ядерной физике, механике, баллистике). Особенностью этих задач было то, что они имели небольшой объем исходных данных, которые сравнительно редко менялись, и поэтому их можно было хранить внутри программы.

При попытке использовать компьютер для решения экономических и управленческих задач возникла следующая проблема: такие задачи имели большой объем исходных данных, и эти данные часто менялись. Следовательно, хранение данных вместе с программой было нецелесообразным. Кроме того, в различных программах встречались очень похожие фрагменты кода, выполняющие некоторые стандартные действия: открыть-закрыть файл, найти на внешнем машинном носителе информации (магнитной ленте) нужную запись, отсортировать массив данных, добавить-удалить-изменить (данные в файле) и т. д. Поэтому в середине 50-х гг. XX в. была разработана концепция БД.

Основные положения этой концепции следующие:

• централизованное хранение информации;

• хранение данных независимо от программы их обработки;

• возможность использования одних и тех же данных для решения различных задач;

• специальная организация данных для оптимизации времени обращения к ним.

Тогда же и появилось первое упоминание о БД.

Одинаковые фрагменты кода программ, встречающиеся в самых разных задачах, организовали в виде библиотеки подпрограмм. Такую библиотеку подпрограмм назвали СУБД. Ее основные функции: определение данных (описание структуры БД), их обработка и управление ими. В настоящее время существуют различные СУБД – MS SQL Server, MySQL, Interbase, Oracle, DB2, Paradox, FoxPro и множество других, менее известных.

Администрирование баз данных

Пользователями БД могут быть:

1. прикладные программы;
2. программные комплексы;
3. специалисты предметной области, выступающие в роли потребителей или источников данных, называемые конечными пользователями.

Администратор БД— физическое лицо или группа лиц, ответственных за состояние, развитие и использование БД организации или учреждения.

Администратор БД обеспечивает:

1. работоспособность БД,
2. контролирует и поддерживает полноту, достоверность, непротиворечивость и целостность данных,
3. реализует необходимый уровень защиты.

Защита базы данных.

В современных многопользовательских СУБД защитные механизмы реализуются следующими способами:

1. установка *пароля* для открытия БД. После установки пароля при каждом открытии БД будет появляться диалоговое окно, в которое требуется ввести пароль. Только те пользователи, которые введут правильный пароль, смогут открыть БД. Этот способ достаточно надежен, но он действует только при открытии БД.
2. установка шифра. Для шифрования применяются специальные шифры (тайнопись, криптография). При шифровании БД ее файл кодируется и становится недоступным для чтения с помощью служебных программ или текстовых редакторов. Дешифрование БД отменяет результаты операции шифрования.
3. защита на уровне пользователей:
   - защита приложения, работающего с БД, от повреждения из-за неумышленного изменения пользователями таблиц, запросов, форм, отчетов от которых зависит работа;
   - защита конфиденциальных сведений в базе данных.

   При запуске СУБД от пользователя требуется идентифицировать себя и ввёсти пароль. Пользователи идентифицируются как члены группы. СУБД по умолчанию создает две группы: администраторы (группа "Admins") и пользователи (группа "Users"). Допускается также определение других групп.

Группам и пользователям предоставляются *разрешения на* доступ.

Восстановление базы данных.

Важными средствами восстановления БД являются регулярное и частое резервное копирование БД и ведение журнала транзакций.

Резервная копия БД— это копия всех файлов, составляющих БД. Если повреждается или теряется файл, являющийся частью БД, то из резервной копии можно извлечь копию этого файла и восстановить его в базе.

Журнал транзакций — это группа файлов, в которые записываются изменения, внесенные завершенными транзакциями. Эта информация достаточна для повторного выполнения, если надо восстановить БД.

Сжатие базы данных.

В процессе работы с БД приходится удалять ее объекты. Файл БД становится фрагментированным и место на диске используется нерационально. Сжатие БД приводит к созданию ее копии, в которой диск используется более экономно. Работа с этой копией повышает быстродействие запросов, повышает скорость отбора записей.

СУБД

СУБД - Комплекс языков и программ, позволяющий создавать базы данных и управлять их функционированием.

СУБД обрабатывает обращения к базе данных, поступающие от пользователей, прикладных процессов и выдает необходимые им сведения.
СУБД характеризуется используемой моделью и средствами администрирования, разработки прикладных процессов, работы в информационной сети.

Классификация СУБД

Классификация СУБД по организации данных

Локальные СУБД

При работе с настольной СУБД сами базы данных расположены на том же компьютере, что и СУБД, осуществляющая доступ к ним. Пользователь работает с БД монопольно (в однопользовательском режиме). Такая БД называется **локальной.** СУБД ответственна за выполнение запросов и за поддержание целостности БД.

Распределенные СУБД

Для того чтобы сохранить конкурентоспособность, организации разукрупняют свои информационные ресурсы, делают их распределенными.

В распределенной БД не все данные хранятся централизованно. Они распределены по узлам, удаленным географически, но связанным коммуникационными линиями. Каждый узел имеет собственную (локальную) БД. Кроме того, он может обращаться к данным, хранящимся на других узлах. Пользователь распределенной БД не обязан знать, каким образом ее компоненты размещены в узлах сети и представляет себе эту БД как единое целое.

Распределенная база данных (РаБД) — совокупность логически взаимосвязанных баз данных, распределенных в компьютерной сети.

Работу с распределенной БД обеспечивают распределенные СУБД.

Распределенная СУБД (РаСУБД) — это программная система, которая обеспечивает управление распределенной БД и прозрачность ее распределенности для пользователей.

Требования к РаБД:

- локальная автономность;
- никакой конкретный сервис не должен возлагаться на какой-либо специально выделенный центральный узел;
- непрерывность функционирования;
- независимость от местоположения, от фрагментации, от тиражирования;
- распределенная обработка запросов;
- управление распределенными транзакциями;
- независимость от оборудования, от операционных систем, от сети, от СУБД.

РаБД могут быть однородными и неоднородными. *Однородные* РаБД имеют в своей основе одну СУБД, обычно с единственным языком баз данных; *неоднородные* РаБД — две или более существенно различающиеся СУБД.

В РаБД различаются и формы распределения данных. В одних случаях данные фрагментируются, т. е. делятся на порции, распределенные между множеством физических ресурсов. *Фрагментация* есть горизонтальная (деление по географическому или другому характеристическому признаку) и вертикальная (разбивание таблицы по столбцам). Независимо от того, какого вида применяется фрагментация, поддерживается глобальная схема, позволяющая воссоздать из имеющихся фрагментов логически централизованную таблицу или другую структуру БД. Пользователь взаимодействует с РаБД посредством транзакций. Транзакция может вызвать несколько процессов в различных узлах, контролируемых независимыми программными модулями.

В других случаях данные тиражируются. *Тиражирование* — это создание дублирующих копий (репликатор) объектов БД на разных узлах с целью повышения доступности и/или сокращения времени доступа к критически важным данным.

Репликаты — это множество различных физических копий некоторого объекта БД (обычно таблицы), для которых в соответствии с определенными в БД правилами поддерживается синхронизация (идентичность) с некоторой "главной" копией.

Классификация СУБД по способу доступа к данным

Следующим шагом в развитии настольных СУБД было появление их сетевых многопользовательских версий, которые обеспечивали одновременную работу нескольких пользователей *с централизованной БД* — БД, размещаемой на одном компьютере — сервере сети. На сервере сети располагается и СУБД. С компьютера пользователя запускается СУБД с сервера, и в результате на нем создается копия СУБД. По каждому запросу пользователя к БД все данные из базы пересылаются на его компьютер, независимо от того, сколько их нужно реально для выполнения запроса. В результате на компьютере пользователя создается локальная копия БД (время от времени обновляемая из реальной БД на сервере сети). Затем СУБД пользователя выполняет запрос. Данная компьютерная архитектура именуется

архитектурой **файл-сервер**. В ней вся тяжесть выполнения запросов к БД и управления целостностью БД ложится на СУБД пользователя. Это приводит к тому, что сеть серьезно "забивается" и безопасность работы невысока. Секретность и конфиденциальность информации обеспечить также трудно.

Наиболее эффективную работу с централизованной БД обеспечивает архитектура **клиент-сервер.** Централизация хранения и обработки данных является базовым принципом этой компьютерной архитектуры.

На сервере сети размещается БД и устанавливается мощная серверная СУБД — сервер баз данных.

Сервер БД—это программный компонент, обеспечивающий хранение больших объемов информации, ее обработку и предоставление пользователям в сетевом режиме.

На компьютере-клиенте приложение-клиент формирует запрос к БД. Серверная СУБД обеспечивает интерпретацию запроса, его выполнение, формирование результата запроса и пересылку его по сети на клиентский компьютер, который интерпретирует его необходимым образом и предоставляет пользователю. Клиентское приложение может также посылать запрос на обновление БД, и серверная СУБД внесет необходимые изменения в БД.

В архитектуре клиент-сервер функции клиентского приложения и серверной СУБД разделены.

При клиент-серверной обработке уменьшается сетевой трафик, так как через сеть передаются только результаты запросов. Груз файловых операций ложится в основном на сервер, который мощнее клиентов и поэтому способен быстрее обслуживать запросы. Как следствие этого, уменьшается потребность клиентских приложений в оперативной памяти.

Технология клиент-сервер имеет огромный потенциал, способный повлиять на расширение возможностей прикладных программ в бизнесе.

Современные серверные СУБД:

- существуют в нескольких версиях для различных платформ, как правило, для различных коммерческих версий .
- в подавляющем большинстве поставляются с удобными административными утилитами;
- осуществляют резервное копирование данных и журналов транзакций;
- поддерживают несколько сценариев репликаций (копирование информации из одной БД в несколько других).
- позволяют параллельную обработку данных в многопроцессорных системах.
- поддерживают создание хранилищ данных. *Хранилище данных* — это совокупность данных, полученных прямо или косвенно из информационных систем, которые содержат текущую и деловую информацию, а также из некоторых внешних источников.
- выполняют распределенные запросы и транзакции.

- дают возможность использовать различные средства проектирования схем данных — универсальные или ориентированные на конкретную СУБД;
- имеют средства разработки клиентских приложений и генераторы отчетов;

Типовая организация СУБД

- ядро, которое отвечает за управление данными во внешней и оперативной памяти, управление транзакциями и журнализацию. При использовании архитектуры "клиент-сервер" ядро является основной составляющей серверной части системы.
- компилятор языка SQL
- подсистема поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД
- сервисные программы (внешние утилиты), обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы.

База данных

База данных (БД) — именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.

Под **предметной областью** принято понимать часть реального мира, подлежащего изучению для организации управления и в конечном счете автоматизации, например, для ведения счетов, учета материальных ценностей, планирования и т.п. **Структурирование данных** – это введение соглашения о способах представления данных.

Проектирование базы данных

Проектирование БД является очень важным этапом, от которого зависят последующие этапы разработки СУБД. Время, затраченное разработчиком на проектирование БД, обычно окупается высокой скоростью реализации проекта.

Перед созданием базы данных необходимо располагать описанием выбранной предметной области, которое должно охватывать реальные объекты и процессы, иметь всю необходимую информацию для удовлетворения предполагаемых запросов пользователя и определить потребности в обработке данных.

На основе такого описания на этапе проектирования базы данных осуществляется определение состава и структуры данных предметной области, которые должны находиться в базе данных и обеспечивать выполнение необходимых запросов и задач пользователя. Структура данных предметной области может отображаться информационно-логической моделью. На основе этой модели легко создается реляционная база данных.

Под проектированием понимают процесс создания описаний новой системы, которая способна функционировать при постоянном совершенствовании ее технических, программных, информационных составляющих и расширять спектр реализуемых управленческих функций и объектов взаимодействия**.**

Целью **концептуального проектирования** является разработка БД на основе описания предметной области. Это описание должно содержать совокупность документов и данных, необходимых для загрузки в БД, а также сведения об объектах и процессах, характеризующих предметную область. Такое описание охватывает весь класс реальных объектов, процессов и явлений, т.е. сущностей, информация о которых должна содержаться в БД и обеспечивать реализацию возможных запросов к БД и решение задач. Разработка БД начинается с определения состава данных, подлежащих хранению в базе для обеспечения выполнения запросов пользователя. Далее производится их анализ и структурирование.

Целью **логического проектирования** является выбор конкретной СУБД и преобразование концептуальной модели в логическую. Для реляционной БД этот этап состоит в разработке структуры таблиц, связей между ними и определении ключевых реквизитов.

Этап **физического проектирования** дополняет логическую модель характеристиками, которые необходимы для определения способов физического хранения и использования БД, объема памяти и типа устройств для хранения.

Наиболее рациональным считается сочетание перечисленных подходов к проектированию. Это связано с тем, что на начальном этапе, как правило, еще не имеется исчерпывающих сведений о всех задачах и пришлось бы отложить проектирование и создание БД до постановки всех задач. Использование такой технологий удобно потому, что средства создания реляционной БД в СУБД позволяют на любом этапе разработки внести изменения в БД и модифицировать ее структуру без ущерба для введенных ранее данных. Эта технология предполагает использование предварительных сведений о необходимости получения из БД различной информации.

В результате проектирования БД должна быть разработана информационно-логическая модель (ИЛМ) данных, т.е. определен состав реляционных таблиц, их структура и логические связи. Структура реляционной таблицы определяется составом полей, типом и размером каждого поля, а также ключом таблицы.

Информационно-логическая модель отображает данные предметной области в виде совокупности информационных объектов и связей между ними. Эта модель представляет данные, подлежащие хранению в базе данных.

Функции управления БД

- управление данными во внешней памяти – обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей.
- управление буферами оперативной памяти – обычно СУБД работает с БД значительного размера, по крайней мере этот размер обычно существенно больше доступного объема оперативной памяти. Если при обращении к элементу будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройств внешней памяти, поэтому единственный способ увеличить скорость является буферизация данных оперативной памяти, выполняемая самой СУБД, а не ОС.
- управление транзакциями. **Транзакция** – последовательность операций над БД, рассматриваемых как единое целое.
- журнализация. СУБД должна обеспечивать надежность хранения данных.
- поддержка языков БД.

Назначение транзакций

Транзакция - это последовательность операций, которые должны быть или все выполнены или все не выполнены (все или ничего).
Методом контроля за транзакциями является ведение журнала, в котором фиксируются все изменения, совершаемые транзакцией в БД. Если во время обработки транзакции происходит сбой, транзакция откатывается - из журнала восстанавливается состояние БД на момент начала транзакции.
В СУБД различных поставщиков начало транзакции может задаваться явно (например, командой BEGIN TRANSACTION), либо предполагаться неявным (так определено в стандарте SQL), т.е. очередная транзакция открывается автоматически сразу же после удачного или неудачного завершения предыдущей.

Основные понятия о моделях данных

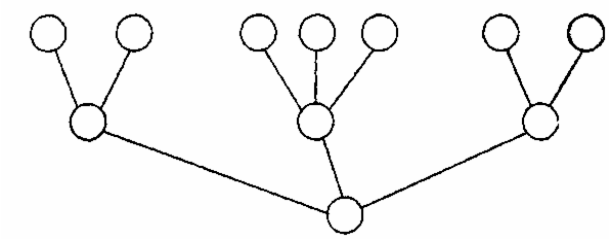
Модель данных - совокупность структур данных и операций их обработки.

На практике наиболее распространены три модели:

- иерархическая
- сетевая
- реляционная

Иерархическая модель данных

Данные представлены в виде *древовидной структуры*. Иерархическая модель представляет собой перевернутое дерево. На самом верхнем уровне только один узел — корень.



Узел – информационная модель элемента, находящегося на данном уровне иерархии.

Свойства иерархической модели данных:

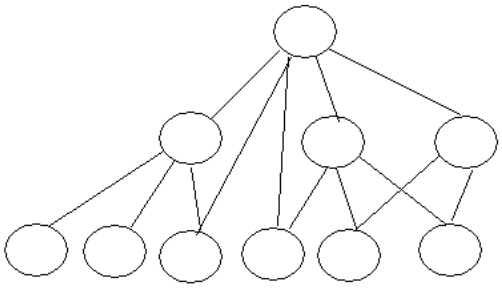
- Несколько узлов низшего уровня связано только с одним узлом высшего уровня.
- Иерархическое дерево имеет только одну вершину (корень), не подчиненную никакой другой вершине.
- Каждый узел имеет свое имя (идентификатор).
- Существует только один путь от корневой записи к более частной записи данных.

Иерархическая модель характеризуется сложностью, неоднородностью, что затрудняет манипулирование данными. Ее применение ограничено, так как не любая предметная область может быть представлена с помощью этой модели.

Сетевая модель данных

Данные представлены в виде *произвольного графа*.

Сетевая модель представляет структуру, у которой один или несколько порожденных элементов имеют более одного исходного элемента. В сетевой структуре любой элемент может быть связан с любым другим элементом.



Иерархическая модель является частным случаем сетевой. Сетевые модели более универсальны. Взаимосвязи большинства предметных областей имеют сетевой характер. Технология работы с сетевыми моделями удобна для пользователя, так как возможен непосредственный доступ к элементам данных.

В качестве примера можно рассмотреть базу данных, хранящую сведения о закреплении учителей - предметников за определенными классами. Один учитель может преподавать в нескольких классах и один и тот же предмет могут вести разные учителя.

Достоинства и недостатки сетевой и иерархической моделей:

Достоинства:

- Компактность данных
- Высокое быстродействие при обработке

Недостатки:

- Не универсальность
- Высокая степень зависимости от конкретных данных (проблемы с обновлением, дополнением, изменением)

Реляционная модель данных

На практике более распространена реляционная модель.

Данные представляют собой *совокупность таблиц*, связанных отношениями

Сотрудник фирмы IBMЭ.Кодд предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение). Он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как отношение – relation.

Реляционная модель БД представляет собой набор связанных между собой таблиц. Каждая из таблиц содержит информацию о каких-либо объектах одной группы. Все записи одной таблицы имеют идентичные, заданные пользователем, структуру и размеры. Например, какой-либо журнал. Каждая строка такой таблицы называется **записью**, а столбец – **полем.**

Имя поля — любое имя длиной до 64 символов и может содержать буквы кириллицы, пробелы и специальные символы, за исключением точек, восклицательных знаков и угловых скобок.

Имя каждому полю дается при создании базы данных. Обычно имя состоит из 3—б символов (до 10). Имена разных полей совпадать не могут.

Длина поля — это максимальное число символов, которые могут быть записаны в поле.

Структура базы данных

Структурой базы данных называют порядок расположения полей записи с указанием имени, типа (формата) и длины.

Таблицы должны обладать следующими свойствами:

- каждый столбец таблицы — это элемент данных (атрибут) и его значения должны быть не расчленяемыми на несколько значений
- все столбцы однородные
- в таблице нет двух одинаковых строк
- столбцы и строки могут просматриваться в любом порядке, безотносительно к их информационному содержанию и смыслу
- число строк не ограничено

Доступ к записям базы данных

Поиск отдельных записей осуществляется по содержанию идентифицирующего поля (идентификатора).

Ключ является идентификатором, уникально определяющим запись об объекте.

Операции реляционной алгебры (объединение, пересечение, разность, произведение)

Объединение – отношение с тем же заголовком, что и у совместимых по типу отношений A и B, и телом, состоящим из кортежей, принадлежащих или A, или B, или обоим отношениям. (A UNION B).
Пересечение - отношение с тем же заголовком, что и у отношений A и B, и телом, состоящим из кортежей, принадлежащих одновременно обоим отношениям A и B. (A INTERSECT B).
Вычитание - отношение с тем же заголовком, что и у совместимых по типу отношений A и B, и телом, состоящим из кортежей, принадлежащих отношению A и не принадлежащих отношению B. (A MINUS B)
Декартово произведение отношение (A1, A2, …, Am, B1, B2, …, Bm), заголовок которого является сцеплением заголовков отношений A(A1, A2, …, Am) и B(B1, B2, …, Bm), а тело состоит из кортежей, являющихся сцеплением кортежей отношений A и B: (a1, a2, …, am, b1, b2, …, bm), таких, что (a1, a2, …, am)∈ A, (b1, b2, …, bm)∈ B. Т.е. каждый кортеж первого отношения объединяется с каждым кортежем второго отношения. (A TIMES B).

Операции реляционной алгебры(выборка, создание проекций, деление)

Ограничение (выборка) - отношение с тем же заголовком, что и у отношения A, и телом, состоящим из кортежей, значения атрибутов которых удовлетворяет некому условию C. C представляет собой логическое выражение, в которое могут входить атрибуты отношения A и/или скалярные выражения. (A WHERE C)
Проекция – отношение, кортежи которого являются соответствующими подмножествами отношения операнда.
A[X, Y, …, Z] или PROJECT A {x, y, …, z}
Соединение – отношение, кортежи которого производятся путем объединения кортежей первого и второго отношения и удовлетворяют некому условию. ((A TIMES B) WHERE C = A JOIN B WHERE C
Реляционное деление - отношение с заголовком (X1, X2, …, Xn) и телом, содержащим множество кортежей (x1, x2, …, xn), таких, что для всех кортежей (y1, y2, …, ym) ∈ B в отношении A(X1, X2, …, Xn, Y1, Y2, …, Ym) найдется кортеж (x1, x2, …, xn, y1, y2, …, ym). (A DIVIDEBY B)

Функции администрирования БД

Администрирование базы данных – это функция управления базой данных (БД). Лицо ответственное за администрирование БД называется “Администратор базы данных” (АБД) или “Database Administrator” (DBA).
организационное и техническое планирование БД, проектирование БД, обеспечение поддержки разработок прикладных программ, подготовка вычислительного комплекса к установке СУБД, участие в установке и приемке СУБД и самой БД с комплексом

прикладных программ, управление эксплуатацией БД, подготовка словарей и другой НСИ - нормативно-справочной информации - к моменту начала испытания БД.

Жизненный цикл БД

1.планирование разработки БД. Рассмотрение с какой целью разрабатывается. Определение объема работ и ресурсов.
2.определение требований к системе, определение диапазон действий, состав пользователей, область применения.
3.сбор и анализ требований пользователей. Анализ документооборота сведений о выполнении транзакций, список требований с указанием приоритетов
4.проектирование БД:
- потенциальное проектирование (создание подробной модели)
- логическое(слияние отдельных моделей в глобальную логическую модель)
- физическое(описание реляционных таблиц и ограничений, разработка ср-в защиты)

5. разработка приложений - проектирование транзакций и пользовательского интерфейса
6. реализация
7. загрузка данных
8. тестирование
9. эксплуатация и сопровождение

Нормализация отношений

Нормализация- это процесс последовательной замены таблицы ее полными декомпозициями до тех пор пока все они не будут находится в 5НФ.

Реляционная база данных представляет собой некоторое множество таблиц, связанных между собой. Число таблиц в одном файле или одной базе данных зависит от многих факторов, основными из которых являются:

1. состав пользователей базы данных,
2. обеспечение целостности информации (особенно важно в многопользовательских информационных системах),
3. обеспечение наименьшего объем требуемой памяти и минимального времени обработки данных.

Учет данных факторов при проектировании реляционных баз данных осуществляется методами нормализации таблиц и установлением связей между ними.
Нормализация таблиц представляет собой способы разделения одной таблицы базы данных на несколько таблиц, в целом отвечающих перечисленным выше требованиям.
Нормализация таблицы представляет собой последовательное изменение структуры таблицы до тех пор, пока она не будет удовлетворять требованиям последней формы нормализации. Всего существует шесть форм нормализации:
• первая нормальная форма (First Normal Form — INF);
• вторая нормальная форма (Second Normal Form — 2NF);
• третья нормальная форма (Third Normal Form — 3NF);
• нормальная форма Бойса—Кодда (Brice—Codd Normal Form — BCNF);
• четвертая нормальная форма (Fourth Normal Form — 4NF);

• пятая нормальная форма, или нормальная форма проекции-соединения (Fifth Normal Form — 5NF, или PJ/NF).

Примеры нормализации

Самым простым примером реляционной БД является БД, состоящая всего из одного отношения – одной таблицы. Однако в таком случае, как правило, не будут выполняться основные требования, предъявляемые к структуре БД из-за возникающих проблем избыточности информации, нарушения целостности данных, сложности редактирования данных, низкой скорости обработки информации и т. п.

Пример. В таблице «Поставки товаров» есть сведения о товарах, их цене, количестве и стоимости, а также о поставщиках, их адресах и расчетных счетах:

Поставки товаров

Товар	Цена	Коли-чество	Стоимость	Поставщик	Адрес	Счет
Стол	12 000	100	1 200 000	Пинскдрев	226000, Брестская обл., г. Пинск	1100022
Стул	6 000	800	4 800 000	Орбита	220111, Минская обл., г. Слуцк	2211003
Кресло	20 000	200	4 000 000	Столиндрев	226100, Брестская обл., г. Столин	3322004
Диван	30 000	80	2 400 000	Пинскдрев	226000, Брестская обл., г. Пинск	1100022

Анализируя структуру таблицы, необходимо, прежде всего, отметить, что в ней имеется повторяющаяся информация о поставщике. Кроме того, стоимость товара является избыточной информацией, т. к. всегда может быть получена на основе цены товара и его количества. Далее, атрибуты «Адрес» и «Счет» характеризуют только поставщика и, вообще говоря, не связаны с поставляемым товаром. Существуют и другие более тонкие недостатки в структуре такой БД.

Таким образом, на первом этапе проектирования реляционной БД важнейшим является вопрос, какую выбрать схему отношений для данной БД из множества альтернативных вариантов, т. е. какую систему таблиц и с каким набором столбцов в каждой таблице выбрать для данной БД. Как правило, БД содержит объекты разных типов, и для каждого типа объектов создается своя таблица с соответствующим набором столбцов-атрибутов объекта.

Процесс создания оптимальной схемы отношений для реляционной БД строго формализован и называется нормализацией БД.

Нормализация – это формализованная процедура, в процессе выполнения которой атрибуты данных группируются в таблицы, а таблицы, в свою очередь, в БД.

Цели нормализации следующие:

• исключить дублирование информации;

• исключить избыточность информации;

• обеспечить возможность проведения непротиворечивых и корректных изменений данных в таблицах;

• упростить и ускорить поиск информации в БД.

Процесс нормализации состоит в приведении таблиц реляционной БД к так называемым нормальным формам. Всего существует пять нормальных форм, которые удовлетворяют соответствующим правилам нормализации. При этом в большинстве случаев оптимальная структура БД достигается при выполнении уже первых трех правил нормализации, которые были сформулированы для реляционных БД Э. Ф. Коддом в 1972 г.

Чтобы таблица, а вместе с ней и БД, соответствовала первой нормальной форме, необходимо, чтобы все значения ее полей были атомарными (неделимыми) и невычисляемыми, а все записи – уникальными (не должно быть полностью совпадающих строк). Выполняя это правило, преобразуем первоначальную таблицу к следующему виду:

Поставки товаров

Товар	Цена	Количество	Поставщик	Индекс	Область	Город	Счет
Стол	12 000	100	Пинскдрев	226000	Брестская	Пинск	1100022
Стул	6 000	800	Орбита	220111	Минская	Слуцк	2211003
Кресло	20 000	200	Столиндрев	226100	Брестская	Столин	3322004
Диван	30 000	80	Пинскдрев	226000	Брестская	Пинск	1100022

Чтобы таблица соответствовала второй нормальной форме, необходимо, чтобы она уже находилась в первой нормальной форме и все неключевые поля полностью зависели от ключевого. В данной таблице на роль ключевого поля может претендовать только поле (атрибут-признак) «Товар», значения которого в таблице не повторяются. Из других полей только поле «Поставщик» непосредственно связано с поставляемым товаром (полем «Товар»), а поля «Индекс», «Область», «Город» и «Счет» характеризуют только самого поставщика. Поэтому, удовлетворяя второму правилу нормализации, необходимо разбить (или разложить) исходную таблицу на две – соответственно «Товары» и «Поставщики».

Товары

Товар	Цена	Количество	Поставщик
Стол	12 000	100	Пинскдрев
Стул	6 000	800	Орбита
Кресло	20 000	200	Столиндрев
Диван	30 000	80	Пинскдрев

Поставщики

Поставщик	Индекс	Область	Город	Счет
Пинскдрев	226000	Брестская	Пинск	1100022
Орбита	220111	Минская	Слуцк	2211003
Столиндрев	226100	Брестская	Столин	3322004

Проведенное преобразование называется разложением, или проектированием, БД и является обратимой операцией. Причем проектирование исходной таблицы привело, с одной стороны, к уменьшению записей (строк) во второй таблице, однако, с другой стороны, для организации связей между отдельными таблицами и обеспечения таким образом целостности БД поле «Поставщик» появилось уже в обеих таблицах, привнося этим некоторую неизбежную избыточность информации. Очевидно, что в больших БД, где реально существуют сотни и тысячи записей, эта избыточность во много раз будет перекрыта уменьшением общего размера таблиц, полученных из исходной таблицы при ее разложении.

Заметим, что на роль ключевого поля таблицы «Поставщики» подходит поле «Поставщик», значения которого в этой таблице уже не повторяются. Можно отметить, что на эту роль вполне подходит и поле «Счет», значения которого также не будут повторяться, а само поле, хотя и выглядит как набор цифр, все же является не количественной, а качественной характеристикой объекта, т. е. является атрибутом-признаком, что необходимо для ключевого поля.

Чтобы теперь перейти к третьей нормальной форме, необходимо, прежде всего, обеспечить, чтобы все таблицы БД находились во второй нормальной форме и все неключевые поля в таблицах зависели только от ключа таблицы и не зависели непосредственно друг от друга.

Анализируя таблицу «Поставщики», можно заметить, что поля «Область» и «Город» являются зависимыми от поля «Индекс», и поэтому эта таблица не находится в третьей нормальной форме.

В связи с этим необходимо разбить таблицу на две: оставить в таблице «Поставщики» только два («Поставщик» и «Счет»), а также поле «Индекс» для обеспечения связи между таблицами, а остальные поля выделить в новую таблицу «Адреса», в которой поле «Индекс», естественно, будет ключевым, т. к. его значения в таблице не повторяются.

Поставщики

Поставщик	Индекс	Счет
Пинскдрев	226000	1100022
Орбита	220111	2211003
Столиндрев	226100	3322004

Адреса

Индекс	Область	Город
226000	Брестская	Пинск
220111	Минская	Слуцк
226100	Брестская	Столин

Приведение БД к четвертой и пятой нормальным формам является необходимой операцией в специальных случаях, когда между элементами БД существуют связи типа «многие-ко-многим» и при этом необходимо обеспечить возможность точного восстановления исходной таблицы из таблиц, на которые она была спроектирована.

Как уже говорилось, этими правилами нормализации при проектировании БД в большинстве случаев можно пренебречь.

Обеспечение целостности данных в БД

Эта характеристика подразумевает наличие средств, позволяющих удостовериться, что информация в базе данных всегда остается корректной и полной. Должны быть установлены правила целостности, и они должны храниться вместе с базой данных и соблюдаться на глобальном уровне. ***Целостность данных*** должна обеспечиваться независимо от того, каким образом данные заносятся в память (в интерактивном режиме, посредством импорта или с помощью специальной программы).

К средствам обеспечения целостности данных на уровне СУБД относятся:

встроенные средства для назначения первичного ключа, в том числе средства для работы с типом полей с автоматическим приращением, когда СУБД самостоятельно присваивает новое уникальное значение;

средства поддержания ссылочной целостности, которые обеспечивают запись информации о связях таблиц и автоматически пресекают любую операцию, приводящую к нарушению ссылочной целостности.

Некоторые СУБД имеют хорошо разработанный процессор СУБД для реализации таких возможностей, как уникальность первичных ключей, ограничение (пресечение) операций и даже каскадное обновление и удаление информации. В таких системах проверка корректности, назначаемая полю или таблице, будет проводиться всегда после изменения данных, а не только во время ввода информации с помощью экранной формы. Это свойство можно настраивать для каждого поля и для записи в целом, что позволяет контролировать не только значения отдельных полей, но и взаимосвязи между несколькими полями данной записи.

Типы данных СУБД Access

Каждое поле базы данных имеет *имя, тип (формат) и длину*.

Тип (формат) поля связан с формой представления информации в этом поле. К основным типам (форматам) полей относят:

Текстовый (символьный) - предназначен для записи алфавитно-цифровой информации;

Числовой - предназначен для записи чисел.

Существуют и другие типы (форматы) полей.

Тип данных	Назначение	Размер
текстовый	- текст, - комбинация текста и чисел, - числа, не требующие вычислений (например, номера телефонов или почтовые индексы)	до 255 символов
поле МЕМО	длинный текст или числа (например, примечания или описания)	до 64 000 символов

числовой	числовые данные (целые или дробные), используемые для вычислений	1, 2, 4 или 8 байт.
дата/время	даты и время	8 байт
денежный	денежные значения. Используется для предотвращения округлений во время вычислений, предполагает до 15 символов в целой части числа и 4 - в дробной	8 байт
счетчик	автоматическая вставка последовательных (увеличивающихся на 1) или случайных чисел при добавлении в таблицу каждой новой записи. Часто используется в качестве ключа. Значения в этих полях не могут обновляться	4 байт
логический	логические значения (да/нет, истина/ложь, вкл/выкл)	1бит
поле объекта OLE	объекты, созданные в других приложениях, использующих протокол OLE (например, таблица MSExcel или документ MSWord)	До1 Гб
гиперссылка	адрес ссылки (путь) на документ или файл, находящийся в Internet, интрасети или локальном компьютере	до 64 000 символов (байт)

Мастер подстановок *- создает поле, в котором предлагается выбор значений из раскрывающегося списка, содержащего набор постоянных значений или значений из другой таблицы.*

Организация поиска информации в БД. Сортировка записей.

Поиск в базе данных — это отбор записей, удовлетворяющих **условиям** поиска, заданным в форме **фильтра или запроса.** Фильтр просто скрывает в исходной таблице записи, не удовлетворяющие условиям поиска. Фильтры и запросы позволяют отбирать записи, которые удовлетворяют условиям поиска. Условия поиска записей создаются с использованием **операторов сравнения** (=, >, < и т. д.). Простые фильтры и запросы содержат условие поиска записей только для одного поля. Составные фильтры и запросы содержат несколько условий поиска для различных полей. В результате применения составного фильтра будут отобраны только те записи, которые удовлетворяют всем условиям одновременно.

Упорядочение записей называется **сортировкой.** Сортировка записей производится по какому-либо полю базы данных. Значения, содержащиеся в этом поле, располагаются **в** порядке **возрастания** или **убывания. В** процессе сортировки целостность записей сохраняется, т. е. **строки** таблицы перемещаются целиком. В базах данных можно проводить **вложенные сортировки,** т. е. сортировать данные последовательно по нескольким полям. При вложенной сортировке строки, имеющие одинаковые значения в ячейках первого поля, будут упорядочены по значениям в ячейках второго поля, а строки, имеющие одинаковые значения во втором поле, будут упорядочены по значениям третьего поля.

Индекс – средство, ускоряющее поиск и сортировку в таблицы за счет использования ключевых значений, которое позволяет обеспечить уникальность строк таблицы. Access автоматически индексирует поле первичного ключа. Для этого нужно выделить в режиме конструктора таблиц поле или поля, которые нужно проиндексировать, щелкнуть правой кнопкой мыши, и выбрать в контекстном меню пункт Ключевое

поле. Кроме того, в том же режиме конструктора можно на вкладке свойств поля выбрать свойство Индексированное поле для любого поля. Потом при желании иметь отсортированную по возрастанию (убыванию) информацию, можно, нажав кнопку Индексы на панели инструментов, выбрать в поле Порядок сортировки соответствующую строку и установить желаемый порядок. Сортировка базы данных в Access представляет собой очень простое действие. При открытии таблицы можно выбрать меню Записи/Сортировка или нажать на панели инструментов кнопки Сортировка по возрастанию (убыванию). Access, выполняя эту команду, отсортирует записи по данному полю так, что текст идет в алфавитном порядке, в порядке возрастания чисел и от более ранней даты к более поздней, – и наоборот.
При индексировании создается отдельный индексный файл, при этом сама база данных остается неизменной. При сортировке же изменяется база данных, порядок следования записей.

Фильтрация данных

Фильтрация - выделение из БД данных, отвечающих некоторому критерию. Критерии бывают двух типов.
Критерии вычисления – это критерии, которые являются результатом вычисления формулы. Например, диапазон критериев =F7=СУММ($B$2:$B$22) выводит на экран строки, имеющие в столбце F значения большие, чем сумма величин в ячейках B2:B22. Формула должна представлять из себя логическое выражение (условие)**.** При фильтрации будут доступные только те строки, значения которых будут придавать формуле значения ИСТИНА**.**
Критерии сравнения – это набор условий для поиска, используемый для извлечения данных при запросах по примеру. Критерий сравнения может быть последовательностью символов (константой) или выражением (например, Цена > 700)**.** Команда **Фильтр** позволяет отыскивать и использовать нужное подмножество данных в списке. В отфильтрованном списке выводятся на экран только те строки, которые содержат определенное значение или отвечают определенным критериям, при этом другие строки скрываются. Для фильтрации данных используются команды **Автофильтр** и **Расширенный фильтр.** Команда **Автофильтр** устанавливает кнопки скрытых списков (кнопки со стрелками) непосредственно в строку с именами столбцов. С их помощью можно выбирать записи базы данных, которые следует вывести на экран. После выделения элемента в открывшемся списке, строки, не содержащие данный элемент, будут скрыты. Команда **Расширенный фильтр** позволяет фильтровать данные с использованием диапазона критериев для вывода только записей, удовлетворяющих определенным критериям.

Логические связи между таблицами

Между таблицами устанавливаются связи. Связи делают их более информативными, чем они являются по отдельности. Они позволяют минимизировать избыточность данных в БД.

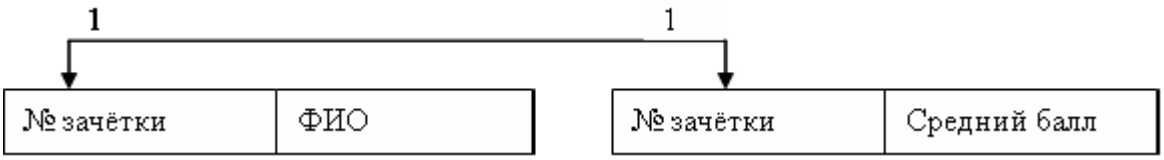
Связь устанавливается посредством связи ключей, содержащих общую информацию для обоих таблиц. Пусть таблица R1 именуется главной, а R2 — подчиненной. Ключ

главной таблицы называется первичным, а подчиненной — вторичным. Особенностью вторичного ключа является то, что его значения могут повторяться.

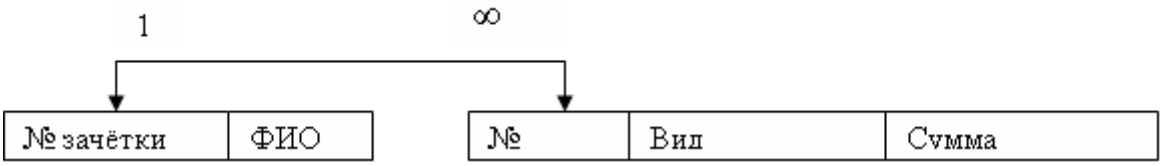
Существует логическая связь четырех типов:

- один к одному (1:1);
- один ко многим (1:∞);
- многие к одному (∞:l)
- многие ко многим (∞:∞)

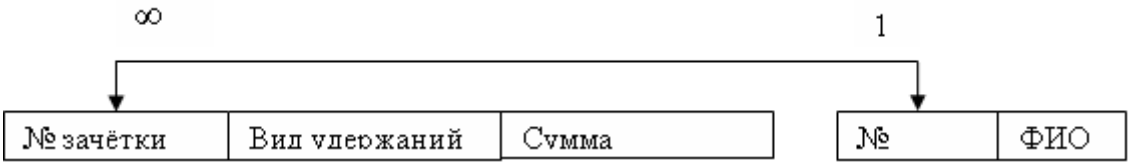
В случае связи 1:1 одному значению первичного ключа соответствует одно и только одно значение вторичного ключа, например:



В случае связи 1:∞ одному значению первичного ключа может соответствовать несколько значений вторичного ключа, например:



В случае связи ∞:1 одному значению вторичного ключа может соответствовать несколько значений первичного, например:



В случае связи ∞:∞ одному значению первичного ключа может соответствовать несколько значений вторичного и одному значению вторичного — несколько значений первичного, например:

Каждый поставщик может поставлять несколько товаров; каждый товар может поставляться несколькими поставщиками.

Объекты СУБД MS Access:

1. Таблица
2. Запрос
3. Форма
4. Отчёт
5. Макрос
6. Страница доступа к данным

7. Модуль

Объект, который позволяет пользователю получить данные из одной или несколько таблиц – **запрос.**

Объект, предназначенный для создания документа, который впоследствии может бать распечатан либо включён в документ другого приложения – **отчёт.**

Объект, предназначенный для просмотра, ввода и редактирования записей базы данных – **форма.**

Страница доступа к данным — диалоговая Web-страница, которая поддерживает динамическую связь с БД и позволяет просматривать, редактировать и вводить данные в базу, работая в окне браузера.

Макрос—этопоследовательность макрокоманд для автоматизации выполнения операций в среде Access без программирования.

Модуль — это программа для работы с БД, написанная на языке Visual Basic for Applications (VBA).

Из всех типов объектов только таблицы предназначены для хранения информации. Остальные используются для просмотра, редактирования, обработки и анализа данных - иначе говоря, для обеспечения эффективного доступа к информации.

Таблицы: Таблицы играют ключевую роль в базах данных, поскольку именно в них хранится информация. База данных может содержать тысячи таблиц, размеры которых ограничиваются только доступным пространством на жестком диске компьютера. Для таблиц обычно используются режим таблицы, предназначенный для ввода данных, и режим конструктора, позволяющий просмотреть и модифицировать структуру таблицы.

Запросы: Запросы предназначены для поиска в базе данных информации, отвечающей определенным критериям. Найденные записи, называемые результатами запроса, можно просматривать, редактировать и анализировать различными способами. Кроме того, результаты запроса могут использоваться в качестве основы для создания других объектов Access. В сущности, запрос представляет собой вопрос, сформулированный в терминах базы данных. Существует различные типы запросов. Наиболее распространенными являются запросы на выборку, параметрические и перекрестные запросы. Для создания простых запросов используется мастер, в менее тривиальных случаях можно создать запрос вручную в режиме конструктора.

Формы: Информация хранится в таблицах в том виде, в котором была введена. Это не имеет особого значения, если никто, кроме вас, не работает с базой данных. Однако если база данных предназначена для пользователей, имеющих смутное представление об Access, работа с таблицами может показаться им чрезмерно сложной, не говоря уже о том, как это отразится на сохранности информации. В таких случаях лучше воспользоваться формами, которые позволяют упростить и сделать более эффективными ввод и обработку содержимого таблиц.В сущности, форма представляет собой окно, куда можно поместить элементы управления, предназначенные для ввода и отображения данных. Access включает панель, которая содержит многие стандартные элементы управления Windows, в том числе поля, надписи, флажки и кнопки выбора. Не требуется особых талантов, чтобы с помощью

этих элементов создать формы, которые выглядят и функционируют примерно так же, как диалоговые окна в приложениях Microsoft Windows.Формы используются для ввода и редактирования записей в таблицах базы данных. Подобно таблицам и запросам, их можно отображать в трех режимах: в режим формы, предназначенном для ввода данных, в режиме таблицы, где данные представлены в табличном формате, и в режиме конструктора, позволяющем изменить внешний вид, содержание и функционирование формы. На следующем рисунке приведен пример формы в режиме конструктора.

Формы – одно из основных средств для работы с базами данных в Access - используются для ввода новых записей (строк таблиц), просмотра и редактирования уже имеющихся данных, задания параметров запросов и вывода ответов на них и др.

Формы представляют собой прямоугольные окна с размещенными в них элементами управления. Существует возможность создания форм динамически при исполнении программы, однако естественным режимом их создания является режим визуального конструирования.

Отчеты: Отчеты используются для отображения информации, содержащейся в таблицах, в отформатированном виде, который легко читается как на экране компьютера, так и на бумаге. Помимо данных, извлеченных из нескольких таблиц и запросов, отчеты могут включать элементы оформления, свойственные печатным документам, как, например, названия, заголовки и колонтитулы. Отчет можно отобразить в трех режимах: в режиме конструктора, позволяющем изменить внешний вид и макет отчета, в режиме просмотра образца, где можно просмотреть все элементы готового отчета, но в сокращенном виде, и в режиме предварительного просмотра, где отчет отображается в том виде, в каком будет напечатан.

Страницы : Чтобы предоставить доступ к информации, хранящейся в базе данных, пользователям Интернета или интранета, можно создать страницы, называемые страницами доступа к данным. Работа с данными на странице доступа в Web осуществляется примерно так же, как в Access - пользователи могут просматривать таблицы, выполнять запросы и заполнять поля форм.Хотя публикация информации из базы данных в Web на первый взгляд кажется сложной, Access включает мастер, которые берет на себя большую часть кропотливой работы по созданию страницы доступа. При желании созданную мастером страницу можно доработать в режиме конструктора.

Макросы: Макросы представляют собой небольшие программы, с помощью которых обеспечивается реакция Access на такие события, как открытие формы, щелчок кнопки или обновление записи. Это особенно удобно, если предполагается передать базу данных неквалифицированным пользователям. Например, можно написать макросы, содержащие последовательность команд, выполняющих рутинные задачи, или связать такие действия, как открытие формы или печать отчета, с кнопками кнопочной формы.

Модули: Модули представляют собой программы на Visual Basic for Applications (VBA), языке программирования высокого уровня, разработанного Microsoft для создания приложений Windows. Помимо стандартного набора команд VBA, каждая программа Microsoft Office имеет собственные команды. В отличие от макросов, позволяющих автоматизировать не более пяти, шести десятков операций, VBA включает сотни команд и может неограниченно расширяться за счет дополнений,

вносимых другими компаниями и частными лицами. Программы VBA используются для решения задач, слишком сложных для макросов, как, например, извлечение определенной информации из рабочих листов Excel.

Назначения и виды запросов

Запросы предназначены для поиска в базе данных информации, отвечающей определенным критериям. Найденные записи, называемые результатами запроса, можно просматривать, редактировать и анализировать различными способами. Кроме того, результаты запроса могут использоваться в качестве основы для создания других объектов Access. В сущности, запрос представляет собой вопрос, сформулированный в терминах базы данных. Существует различные типы запросов.

Наиболее распространенными запросами являются:

- запросы на выборку,
- параметрические запросы,
- перекрестные запросы.

Параметрические запросы

запрос, при выполнении которого в его диалоговом окне пользователю выдается приглашение ввести данные, например условие для возвращения записей или значение, которое требуется вставить в поле.

В параметрическом запросе указывается критерий, который может изменяться по заказу пользователя. Такой запрос выгодно применять как основу для форм и отчетов. Например, в отчете появляется приглашение ввести месяц, за который он составлен. Условие отбора записей задается непосредственно при вызове запроса. При этом для внесения изменений не требуется открывать запрос в окне Конструктора (строка «Условие отбора»).

При создании параметрического запроса прежде всего формируется обычный запрос на выборку. В ячейку строки условие отбора для заданного поля вводится необходимый оператор и имя параметра, заключенное в квадратные скобки. Это имя выводится в диалоговом окне при выполнении запроса, поэтому в качестве имени параметра удобно использовать какую-либо содержательную фразу или приглашение на ввод данных. В одном запросе можно установить несколько параметров, однако каждый параметр должен иметь уникально содержательное имя.

Вычисления в запросах

Запрос на выборку можно использовать не только для того, чтобы выбирать из таблиц базы данных содержащуюся в них информацию, но также чтобыпроизводить вычисления и отображать результаты вычислений в динамической таблице. Следует иметь в виду, что результаты вычислений не сохраняются в таблицах базы данных. При выполнении запроса вычисления выполняются всякий раз заново, поэтому результаты всегда отражают текущее состояние базы данных.

Для выполнения вычислений в запросе необходимо в бланке запроса создатьновое вычисляемое поле. Для этого следует в пустую ячейку строки Поле ввести

вычисляемое выражение, в котором могут использоваться имена полей таблиц, константы и функции, связанные с помощью арифметических операторов. Для изменения порядка вычислений и группировки данных в выражениях используются круглые скобки, а имена полей таблицы должны быть заключены в квадратные скобки. Если в запросе используется несколько таблиц, во избежание ошибок следует указывать полное имя поля, помещая перед именем поля имя таблицы. В этом случае для разделения имен объектов используется **!**.

Таким образом, запись полного имени поля таблицы выглядит следующим образом:
[Имя таблицы]![Имя поля]
Имя вычисляемого поля выводится перед выражением и отделяется от него двоеточием
Имя_Нового_Поля:Правило_вычисления

Групповые операции

Запросы на выборку можно также использовать для группировки записей и вычисления итоговых значений с помощью так называемых групповых функций:
Sum вычисляет сумму всех значений заданного поля в каждой группе;
Avg вычисляет среднее арифметическое всех значений данного поля в каждой группе;
Min (Max) возвращает наименьшее (наибольшее) значение, найденное в этом поле в каждой группе;
Count определяет количество записей в каждой группе и др.

Для создания итогового запроса в бланк запроса по команде Групповые операции на панели инструментов добавляется строка Групповая операция, которая позволяет указать, по какому полю должна быть выполнена группировка записей, и какие вычисления итоговых значений по группам данных необходимо провести.
Допустимо использование вычисляемых полей.

Конструирование перекрестного запроса

Он позволяет создать результирующие таблицы на основе результатов расчётов, полученных при анализе группы таблиц. В перекрёстном запросе отображаются результаты расчетов (таких как суммы, количество записей и среднее значения), выполненных по данным из одного поля таблицы. Эти результаты группируются по двум наборам данных, один из которых расположен в левом столбце таблицы, а второй в верхней строке.

Общая характеристика языка запросов SQL

SQL содержит набор стандартных операторов доступа к данными.
SQL— это язык программирования, предназначенный для выборки и обработки информации, содержащейся в реляционной базе данных. SQL является единственным стандартным языком для работы с реляционными базами данных. SQL является языком реляционных баз данных, его основа реляционная алгебра и реляционное исчисление.
SQL обеспечивает независимость от конкретных СУБД: реляционную базу данных и программы, которые с ней работают, можно перенести с одной СУБД на другую с минимальными доработками и переподготовкой персонала. Все ведущие поставщики

СУБД используют SQL.
Язык SQL является интерпретируемым языком.
Интенсивное развитие технологий БД потребовало разработки стандартного языка, пригодного для создания БД и работы с ними. Таким стал и по прогнозам останется в ближайшее время язык SQL. Поначалу SQL был средством формирования запросов к реляционным БД. С течением времени он превратился в мощное средство для работы с такими БД и стал использоваться многими СУБД. Пользователи, владеющие им, имеют огромные возможности доступа к данным разнообразных баз, применения и интеграции их.

Функции языка SQL предполагают:

- *организацию данных* в таблицах,
- *обновление данных* — добавление в БД новых данных, удаление и изменение уже имеющихся;
- *чтение данных* — SQL дает возможность пользователю или прикладной программе извлекать данные из БД,
- *управление доступом* — с помощью SQL можно ограничить возможности пользователя по чтению и изменению данных и защитить их от несанкционированного доступа;
- *совместное использование данных* — SQL координирует совместное использование данных пользователями, работающими одновременно;
- *целостность данных* — SQL позволяет защитить БД от разрушения из-за несогласованных изменений или отказа компьютерной системы.

Язык SQL можно использовать для доступа к БД в двух режимах: при *интерактивной работе* (командный режим) и в *прикладных программах (*программный режим). Освоив стандарт SQL, пользователь может работать с БД в среде любой реляционной СУБД.

Рассмотрим ограниченное множество команд SQL

Команда	Назначение
Описание данных	
CREATE TABLE	Создает структуру таблицы
Манипулирование данными	
INSERT	Добавляет новые записи в таблицу
DELETE	Удаляет записи из таблицы
UPDATE	Обновляет данные таблицы
Формирование запросов	
SELECT	Извлекает данные из БД

1. Команда SQL начинается с *глагола*, описывающего действие, выполняемое командой. Например, *CREATE* (создать), *INSERT* (добавить), *COMMIT* (завершить).

2. После глагола идет одно или несколько *предложений.* Предложение описывает данные, с которыми работает команда, содержит информацию о действии, выполняемом командой. Каждое предложение начинается с ключевого слова. Например. *WHERE* (где), *FROM* (откуда), *INTO* (куда), *HAVING* (имеющий). Одни предложения являются обязательными, а другие — нет. Предложения содержат имена таблиц, полей БД, ключевые слова, константы и выражения.
Ключевые слова SQL нельзя использовать для идентификации таблиц, полей и пользователей. Имена должны содержать от 1 до 18 символов, начинаться с буквы и не содержать пробелы и специальные символы пунктуации.

Выражение, которое указывает, какие записи необходимо включить в динамическую таблицу при выполнении запроса вводится в строку Условие отбора для поля, по которому это условие необходимо проверить. Для задания условий отбора можно использовать операторы меньше (<), меньше или равно (<=), больше (>), больше или равно (>=), не равно (<>), равно (=), Like (выбор по маске), Between (между), In (в интервале), And (и), Or (или) и другие, а также имена объектов, константы и функции.
Завершение ввода условия выполняется нажатием клавиши Enter или просто переходом к другой ячейке бланка запроса с помощью клавиш управления курсором или мыши. Microsoft Access проводит синтаксический анализ заданного выражения.
Например, если было введено значение текстового поля то это выражение будет представлено в кавычках. Если выражение не содержит никакого оператора, Microsoft Access будет исходить из того, что подразумевается оператор = или Like.
Допускается использование нескольких условий отбора, которые можно задать как для разных полей, так и для одного поля.
Условия отбора, заданные в одной строке, связываются с помощью логической операции "И", заданные в разных строках — с помощью логической операции "ИЛИ". Эти операции могут быть заданы явно в выражении условия отбора с помощью операторов AND и OR соответственно.
Для создания сложных условий выбора данных используются логические операторы And и Or. Если условия отбора связаны оператором And, запись выбирается только в случае выполнения всех условий. Если же условия отбора связаны оператором Or, запись выбирается при выполнении хотя бы одного из всех условий. При определении нескольких условий отбора, связанных оператором And, для различных полей необходимо просто задать условие в строке Условие отбора для каждого из полей, образующих критерий выбора данных. Если же при определении нескольких условий поместить их в различные строки — строку Условие отбора и строку или — Microsoft Acces будет использовать Or-связь. В результате условия, расположенные в одной строке, связываются оператором And, в разных строках — оператором Or.
Т.о., условием отбора является выражение, которое состоит из операторов сравнения и операндов, используемых для сравнения. В качестве операндов выражения могут использоваться: литералы, константы, идентификаторы (ссылки).