БАЗЫ ДАННЫХ

Краткие теоретические сведения

База данных – представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ) (Гражданский кодекс РФ, ст. 1260).

В широком смысле слова база данных – это совокупность сведений о *конкретных объектах* реального мира в *какой-либо предметной области.* Создавая базу данных, пользователь стремится *упорядочить информацию* по *различным признакам* и быстро извлекать нужную информацию с *произвольным сочетанием* признаков.

С понятием базы данных тесно связано понятие системы управления базой данных **(СУБД).** *Это комплекс программных и языковых средств, предназначенных* для создания структуры новой базы, наполнения ее содержимым, редактирования содержимого и организации поиска в них необходимой информации.

Классификация баз данных

Классифицировать базы данных можно по разным признакам:

По технологии обработки данных выделяют централизованные и распределенные БД.

Централизованная(сосредоточенная) база данных хранится в памяти одного компьютера. В сети такой компьютер выполняет роль сервера обслуживающего запросы других машин к базе данных. Такой способ использования баз данных часто применяют в локальных сетях.

Распределенная база данных, составные части которой размещаются в различных узлах компьютерной сети в соответствии с каким-либо критерием. Причем, части базы данных, хранимые в разных узлах могут как пересекаться и дублироваться. Работа с такой базой осуществляется с помощью системы управления распределенной базой данных (СУРБД).

По способу доступа к данным базы данных разделяются на базы данных с *локальным доступом* и базы данных с *удаленным (сетевым) доступом*. Системы централизованных баз данных с сетевым доступом предполагают *различные архитектуры* :

Файл-сервер – архитектура систем БД с сетевым доступом предполагает выделение одной из машин сети в качестве центральной (сервер файлов). На такой машине хранится совместно используемая централизованная БД. Все другие машины сети выполняют функции рабочих станций, с помощью которых поддерживается доступ пользовательской системы к централизованной базе данных. Файлы базы данных в соответствии с пользовательскими запросами передаются на рабочие станции, где в основном и производится обработка. При большой интенсивности доступа к одним и тем же данным производительность информационной системы падает. Пользователи могут создавать также на рабочих станциях локальные БД, которые используются ими монопольно.

Клиент-сервер. В этой концепции подразумевается, что помимо хранения централизованной базы данных центральная машина (сервер базы данных) должна обеспечивать выполнение основного объема обработки данных. Запрос на данные, выдаваемый клиентом порождает поиск и извлечение данных на сервере. Извлеченные данные (но не файлы) транспортируются по сети от сервера к клиенту. Спецификой архитектуры клиент-сервер является использование языка запросов SQL.

По моделям данных базы данных делятся на:

- иерархические
- сетевые
- реляционные

Иерархическая база данных – БД построенная на основе иерархической модели (элементы нижележащего уровня (потомки) имеют только одну связь с элементами вышележащего уровня (родителями) и могут иметь несколько связей с элементами нижележащего уровня).

Объекты, связанные иерархическими отношениями, образуют ориентированный граф (перевернутое дерево) см. рис.1. К основным понятиям иерархической структуры относятся: уровень, элемент (узел), связь. *Узел* - это совокупность атрибутов данных, описывающих некоторый объект. На схеме иерархического дерева узлы представляются вершинами графа.

К каждой записи базы данных существует только один (иерархический) путь от корневой записи. Например, как видно из рис. 1, получить доступ к записи 12 возможно только последовательно пройдя через узлы 1, 2 и 7.

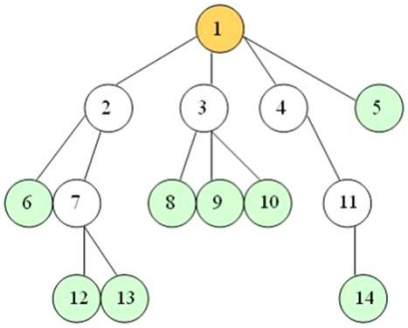

Рис.1.

Сетевая БД – каждый элемент может быть связан с любым другим элементом. Совокупность узлов и линей связей образует неориентированный граф (рис. 2).

Реляционная БД – хранит информацию в виде двумерных таблиц (отношений). Каждая база данных может состоять из одной или нескольких связанных таблиц. Каждая таблица состоит из строк и столбцов, образующих ячейки, содержащие информацию. Данный тип организации данных получил наибольшее распространение в связи с наглядностью и относительной простотой использования хранимой в таблицах данных.

Реляционные БД

Реляционная база данных состоит из одной или нескольких связанных таблиц, структуру которых образуют столбцы и строки.

В реляционных базах данных приняты следующие обозначения:

Отношение – таблица;

Поле– набор однотипных записей для нескольких объектов (столбец);

Кортеж (запись) – строка таблицы, содержащая набор нескольких записей соответствующих одному объекту;

Атрибут – запись в строке одного поля.

Сущность – любой различимый объект, информация о котором хранится в базе данных.

Ключевые поля

Каждое отношение базы данных должно содержать в себе поле (или совокупность нескольких полей), однозначно идентифицирующее каждую запись отношения. Такие поля, позволяют связывать данные нескольких отношений и в конечном счете сформировать единую базу данных. Эти поля называют ключевыми полями.

Различают следующие виды ключей:

Потенциальный ключ – поле, атрибуты которого обеспечивают уникальность записи (в отношении таких полей может быть несколько).

Первичный ключ – один из потенциальных ключей, выбранный в качестве основного (как правило, имеет минимальную длину атрибута).

Внешний (вторичный) ключ – одно или несколько полей отношения, обеспечивающих связь с первичным ключом другого отношения.

В зависимости от количества полей образующих ключ выделяют:

Простой ключ – состоит из единственного атрибута, однозначно определяющего запись (номер зачетной книжки студента).

Составной ключ – состоит из двух и более атрибутов, совокупность которых однозначно определяет запись (серия и номер паспорта человека).

Если в отношении есть уникальное поле, однозначно определяющий каждую запись отношения, то его можно использовать в качестве первичного ключа, но значения его атрибутов должны быть различными для всех записей. Не следует использовать в качестве первичного ключа имена или фамилии людей, т. к. они могут повторятся и в одном отношении могут оказаться люди с одинаковы именем и фамилией. Даже если на данный момент фамилии всех людей зарегистрированных в базе данных разные, поле фамилия не должно использоваться в качестве ключевого, поскольку записи в отношении со временем могут быть изменены в связи с изменением состава людей учтенных в баз данных.

При выборе первичного ключа следует также учитывать, что атрибуты ключевого поля не могут быть пустыми. Если поле допускает пустые значения, то его не следует использовать в качестве первичного ключа.

Также при выборе первичного ключа следует учитывать, что его значения не должны меняться. Если же он меняется, то необходимо обеспечить обновление информации о данном изменении во всех связанных с данным полем отношениях. Применение первичного ключа с постоянным значением позволяет упростить синхронизацию между отношениями в базе данных.

Часто в качестве первичного ключа выбирают искусственно созданное поле, значения атрибутов которого не имеют фактического смысла. Такими полями могут быть ***Код*** или ***Номер***, эти поля содержат только числовое обозначение строки, причем зачастую это обозначение выставляет компьютер при помощи счетчика. Такие коды не подвержены изменениям в отличие от полей содержащих фактические данные, т.к. Фамилия, Номер телефона, Адрес и т.д. могут меняться и повторятся.

В том случае если уникальность записи не может быть обеспечена одним полем применяется составной ключ, образованный двумя или более полями. Примером составного ключа могут являться поля серия и номер паспорта, отдельно серия и номер паспорта не могут гарантировать уникальность записи, т.к. есть паспорта с одинаковой серией, так же как и с одинаковым номером, но одновременное совпадение серии и номера двух паспортов невозможно.

Создание связей между отношениями

После определения списка полей в отношениях и приведения их к нормализованному виду, необходимо установить связи между различными отношениями с целью их объединения в единую базу данных.

Связь между отношениями устанавливается через совпадающие атрибуты ключевых полей (для удобства имена связанных полей делают одинаковыми). В большинстве случаев первичный ключ одного отношения связывается внешним ключом другого отношения.

Существует три типа связей в зависимости от количества записей соответствующих ключу:

- один ко многим
- многие ко многим
- один к одному

Все три типа связей находят применение в проектировании баз данных.

«один ко многим»

Это – самый распространенный тип связи. Каждой записи первичного ключа соответствует несколько записей внешнего ключа.

Рассмотрим пример баз данных состоящей из двух отношений *Клиенты* и *Заказы*.

Клиенты

Код клиента	**ФИО**	**Телефон**	**Адрес**
1	Иванов В.С.	62-62-62	ул. К.Маркса, 38, кв. 5
2	Петров Н.С.	62-45-85	ул. Пирогова, 8, кв.2
3	Васильева Е. Н.	45-98-22	ул. И.Франко, 2, кв. 1

Заказы

Код заказа	**Код клиента**	**Код товара**	**Количество товара**
2161	1	10	2
2165	2	20	1
3369	1	20	1
4166	3	10	2
5864	2	30	2

Очевидно, что в таблице *Клиенты* роль первичного ключа должно выполнять поле *Код клиента*, в то время как в таблице *Заказы* эту роль может выполнять только поле *Код заказа*.

Связь между данными отношениями может быть осуществлена через поля Код клиента. Поле *Код клиента* будет являться первичным ключом для отношения *Клиент* и внешним ключом для отношения *Заказы.*

В таблице *Клиенты*, данное поле содержит уникальные неповторяющиеся записи, в таблице же Заказы записи в поле Код клиента могут повторяться неограниченное число раз, поскольку, один клиент может сделать несколько заказов. Т.е. одной записи таблицы *Клиенты* может соответствовать несколько записей таблицы *Заказы*. Поэтому между двумя этими таблицами устанавливается отношение один ко многим.

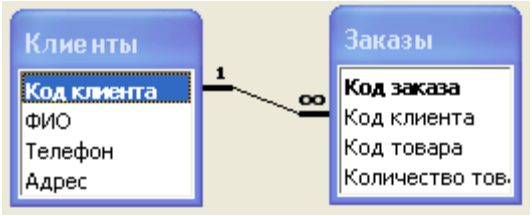

Рис. 2

«многие ко многим» – первичные ключи двух отношений связанные через внешние ключи третей таблицы.

Рассмотрим предыдущий пример баз данных, но добавим туда отношение *Товары*.

Товары

Код товара	**Название товара**	**Цена**

10	Процессор Core i3	5000р.
20	Процессор Amd A10 6800	4500р.
30	Процессор Core i7	9800р.

Имеется три отношения *Клиенты*, *Заказы*, *Товары*. Рассмотрим связь между отношением *Клиенты* и *Товары*. Каждй клиент может заказать несколько товаров и один и тот же товар может быть заказан разными клиентами (рис. 3.). То есть для каждой записи отношения *Клиенты*, можно поставить в соответствие несколько записей в отношении *Товары* и наоборот. Связь с таким соотношением связей называется отношением «многие-ко-многим». Фактические данная связь образована двумя связями «один ко многим» между отношениями *Клиенты – Заказы* и *Товары – Заказы*.

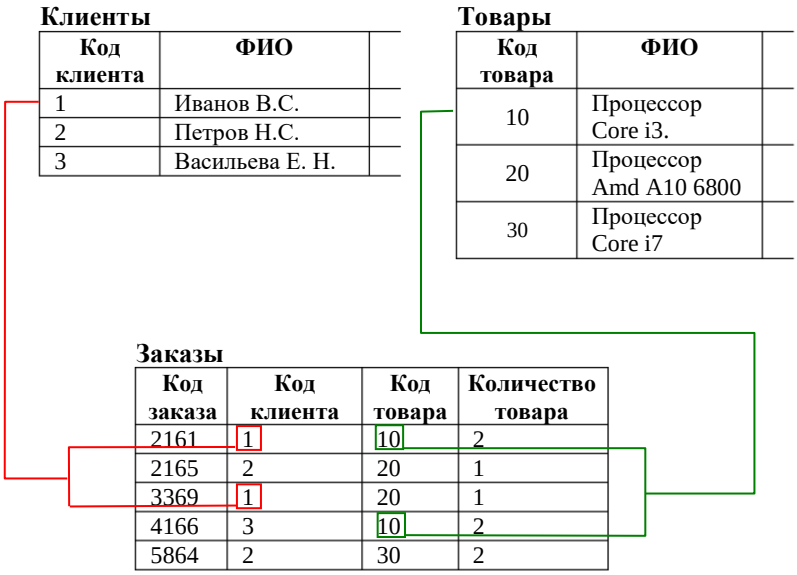

Рис. 3.

Данный тип связи может показаться довольно сложным, попробуем разобраться можно ли упростить данную связь между отношениями *Клиент* и *Товары*. Самым простм способом изменить данную связь в сторону упрощения может показаться вариант с добавлением в отношение *Клиенты* поля *Код товара.* Вариант такого отношения представлен в таблице **Клиенты –код товара.**

Клиенты –код товара

Код клиента	ФИО	Телефон	Адрес	Код Товара
1	Иванов В.С.	62-62-62	ул. К.Маркса, 38, кв. 5	10
1	Иванов В.С.	62-62-62	ул. К.Маркса, 38, кв. 5	20
2	Петров Н.С.	62-45-85	ул. Пирогова, 8, кв.2	20
2	Петров Н.С.	62-45-85	ул. Пирогова, 8, кв.2	30
3	Васильева Е. Н.	45-98-22	ул. И.Франко, 2, кв. 1	10

Очевидно что добавление поля *Код товара* в отношение *Клиенты* приводит к многократному дублированию информации, так информация о клиентах с кодом 1 продублирована в четырех записях и это только при заказе двух разных товаров. Каждая дополнительная запись увеличивает объем должна храниться в базе данных, что приводит к увеличению ее размера, снижению производительности. Следует также учитывать, что в случае изменения информации о клиенте придется обновлять информацию в нескольких кортежах, что в конечном итоге может привести к снижению точности данных. Особенно остро все эти недостатки проявятся в базах данных содержащих тысячи или миллионы кортежей.

Для того, чтобы избавится от всех этих недостатков связь между отношениями обеспечивает третья связующая таблица, развивающая отношение «многие ко многим» на два отношения «один ко многим». Причем поля первичных ключей таблиц *Клиенты* и *Товары* являются вторичными ключами по отношению к таблице *Заказы*. Схема полученной связи представлена на рис. 4.

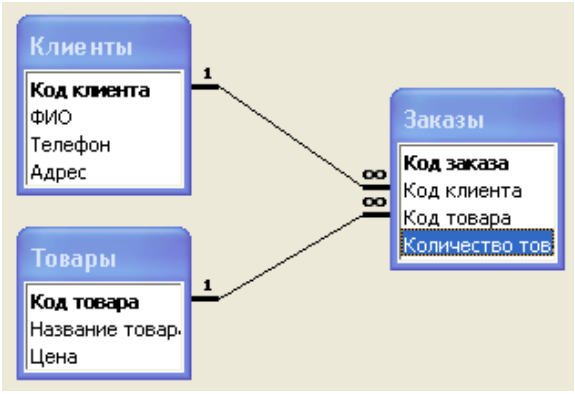

Рис. 4.

«один к одному» – каждой записи первичного ключа (отношение Клиенты) соответствует одна запись внешнего ключа, который также может является первичным в своей таблице (отношение Заказы).

Рис. 5.

Данный тип связи обычно используется в том случае, если в исходной таблице есть кортежи, для которых необходимо добавить дополнительную информацию, несвойственную большинству кортежей данного отношения.

Например, если среди клиентов оказываются оптовые покупатели, то они могут указать помимо своего юридического адреса (поле *Адрес*), адрес склада на который следует доставить товар. Ввиду того, что число обычных клиентов может исчисляться тысячами, а число оптовых клиентов меньше на порядок, добавлять поле с реквизитами склада в отношение *Клиенты* не имеет смысла, т.к. в большинстве кортежей атрибут данного поля останется пустым. Чтобы не создавать пустые записи, информацию о складе выносят в отдельную таблицу *Склад*. В качестве первичного ключа

в отношении склад следует выбрать поле *Код клиента*. Таблица, содержащее дополнительную информацию (*Склад*), и основная таблица (*Клиенты*) связываются между собой отношением «один к одному», через поле *Код клиента*. Данное поле должно являться первичным ключом для обеих таблиц, т.е. каждой записи отношения *Склад* соответствует только одна запись в отношении *Клиенты,* обратное утверждение будет справедливо только для тех кортежей, для которых существует соответствующая запись в поле *Код клиента* в таблице *Склад.*

При установке отношения один к одному следует придерживаться следующих правил:

- Если таблицы связаны тематически, то для связи используется общий первичный ключ.

Если таблицы тематически не связаны и первичные ключи не совпадают, то первичный ключ одной таблицы, вставить во вторую в качестве вторичного ключа.

Объекты базы данных.

Таблицы – основные объекты хранения данных.

Запросы – это формализованное требование на отбор, изменение, добавление или удаление данных из таблиц, а также на создание новых таблиц.

Формы – объект базы данных, предназначенный для ввода, отображения и изменения данных, находящихся в полях таблицы.

Отчеты – обеспечивают вывод информации, хранящейся в таблицах или запросах на печать.

Макросы – объекты, позволяющие автоматизировать операции с данными.

Модули – объекты баз данных, содержащие в себе программный код, написанный на языке Visual Basic for Application (VBA). Модули применяются для автоматизации обработки событий и вычислений.

Страницы – Web-страницы, обеспечивающие доступ к данным удаленных клиентов через сеть Internet

Основным объектом БД является таблица. На базе таблиц осуществляется построение форм, запросов и отчетов. Таблицы используются для хранения информации в базе данных.

Запросы

Запросы применяются для отбора, изменения и анализа записей в базе данных. На основе запросов можно создавать как новые запросы, так и отчеты, формы, страницы доступа к данным.

В Access запросы создаются с помощью языка программирования SQL, причем пользователь может не знать данного языка, поскольку программный код может быть сгенерирован автоматически при заполнении бланков запросов.

В Access выделяют следующие виды запросов:

Запрос на выборку – создает результирующую таблицу, содержащую информацию, взятую из одной или нескольких таблиц или запросов. Результирующая таблица создается каждый раз при выполнении запроса.

Среди запросов на выборку выделяют *итоговые запросы,* которые позволяют проводить вычисления сумм, подсчета количества записей, нахождения средних значений, группировки записей и т.п.Для выполнения этих вычислений используются итоговые функции:

Sum – находит суммарное значение в группе записей;

Avg – находит среднее значение в группе записей;

Min – находит минимальное значение в группе записей;

Max – находит максимальное значение в группе записей;

Count – находит количество записей в группе;

StDev – определяет несмещенную оценку стандартного отклонения для генеральной совокупности, где выборка является подмножеством генеральной совокупности;

Var – возвращают предполагаемое значение дисперсии генеральной совокупности или ее выборки, представленной в виде набора значений, содержащихся в указанном поле запроса;

First – возвращает значение поля из первой записи группы;

Last – возвращает значение поля последней записи группы

Запрос с параметрами – создает результирующую таблицу на основе имеющихся таблиц и запросов, и удовлетворяющий параметру, значение которого вводится через всплывающее окно при каждом выполнении запроса. Эти запросы применяют в том случае, если запрос предполагается выполнять многократно для разных значений одного или нескольких полей, также такие запросы можно применять в качестве основы для создания форм, отчетов и страниц доступа к данным.

Перекрестный запрос – запрос на выборку с измененной структурой результирующей таблицы. Поля в перекрестном запросе распределены по столбцам и строкам, а на их пересечении находится вычисляемый результат. Такая форма представления табличных данных является более наглядной, чем в простых запросах на выборку. В перекрестном запросе в качестве вычисляемых значений можно найти сумму, среднее значение и другие статистические функции.

Запрос на изменение – изменяет или перемещает данные. К этому типу относятся: запрос на удаление записей, запрос на обновление записей, запрос на добавление записей, запрос на создание таблицы.

- *Запрос на удаление записей* – удаляет записи, соответствующие заданному критерию, из одной или нескольких таблиц базы данных.
- *Запрос на обновление записей* – изменяет записи в одной или нескольких таблицах. Изменяются только те записи, значение которых соответствует указанному в запросе критерию.
- *Запрос на добавление записей* – переносит записи из одной или нескольких таблиц в конец другой таблицы.
- *Запрос на создание таблиц* – формирует новую таблицу на основе данных из одной или нескольких таблиц или запросов. Такие запросы бывают полезны при создании «архивных» таблиц, хранящие в себе старые записи.

SQL запросы – запросы созданные при помощи непосредственного ввода инструкций языка SQL (Structured Query Language –англ. структурированный язык запросов). При создании запросов в режиме конструктора SQL код генерируется автоматически, просмотреть и изменить его можно в режиме SQL. В данном режиме могут быть созданы как простейшие запросы на выборку, так и ряд запросов, которые не могут быть выполнены в режиме конструктора. К таким запросам относятся: запрос к серверу, управляющие запросы и запросы на объединение.

Отдельно можно выделить тип запросов с *вычисляемыми полями*. Значения в вычисляемых полях получаются в результате арифметических и логических операций с полями таблиц и запросов базы данных. Полученные в ходе вычислений результаты не хранятся в базе данных, а пересчитываются каждый раз при выполнении запроса.