***Тема 5. Основы проектирования   
баз данных средствами СУБД***

[5.1. Уровни моделей данных](#_5.5.1._Основные_понятия)

[5.2. Основные понятия реляционной модели данных](#_5.5.2._Основные_понятия)

[5.3. Модель «Сущность-Связь» (ER-модель)](#_5.5.3.__Основные)

[5.4. Создание БД средствами СУБД MS Access](file:///E:\сосновиков\Тема-05-05.docx)

5.4.1. Создание таблиц и работа с таблицами

5.4.2. Работа с формами

5.4.3. Запросы-выборки

5.4.4.Язык SQL. Оператор Select

5.5. Задачи для самостоятельного решения

[5.6.](#_5.3.6._Тестовые_задания)  Лабораторная работа

5.7. Контрольные вопросы

### 5.1. Уровни моделей данных

Восприятие реального мира можно сопоставить с последовательностью разных, хотя иногда и взаимосвязанных, явлений. С давних времен люди пытались описать эти явления (даже тогда, когда не могли их понять). Такое описание называют ***данными***.

А как осуществляется фиксация данных? Традиционно фиксация данных осуществляется с помощью конкретных средств общения (например, с помощью естественного языка или изображений) на конкретном носителе. Обычно при естественном общении ***данные*** (факты, явления, события, идеи или предметы) и их ***интерпретация***(семантика) фиксируются совместно, так как естественный язык (язык интерпретации) достаточно гибок для представления того и другого. Примером может служить утверждение "Номер телефона Иванова И.И. – 273-16-38". Здесь "273-16-38" – данное, а " Номер телефона Иванова И.И." – его семантика.

С понятием ***данные*** тесно связано понятие ***информации***.

***Информацию*** обычно определяют как изменение объема и структуры знания о некоторой предметной области (лица, предметы, факты, события, явления, процессы) воспринимающей системой (человек, организационная структура, информационная система) независимо от формы и способа представления знаний. От **информации данные** отличаются конкретной формой представления и являются некоторым ее подмножеством, которое определяется целями и задачами сбора и обработки информации. Поэтому **данные** можно определить как **информацию**, отражающую определенное состояние некоторой предметной области в конкретной форме представления и содержащую лишь наиболее существенные с точки зрения целей и задач сбора и обработки информации элементы отражаемого фрагмента действительности.

Таким образом, **информация** на стадии данных характеризуется **определенной формой** представления и дополнительной характеристикой, выражаемой термином **структура**.

***Структура данных*** связана с понятием **представления информации** и определяется функциональной, логической, технической и т.п. структурой той предметной области, информацию о которой содержат данные. Таким образом, можно выделить **неструктурированную** (простую) и **структурированную** (сложную) **форму** **представления данных**. Мы будем трактовать понятия информации и данных как синонимы.

С понятием информации и интерпретацией данных тесно связано понятие **предметной области**.

***Предметной областью*** называют часть реальной системы (системы реального мира), которая представляет интерес для данного описания или исследования. Следовательно, предметная область позволяет описать систему реального мира с необходимым уровнем детализации.

Для облегчения обработки информации используются **информационные системы (ИС).** Автоматизированными называются ИС, в которых применяются компьютеры.

В широком понимании под определение ИС подпадает любая система обработки информации. Более узкая трактовка понятия ИС – это совокупность аппаратно-программных средств, задействованных для решения некоторой прикладной задачи.

Исторически, одним из первых этапов интеграции данных было объединение данных в **файлы.**

С точки зрения прикладной программы ***файл* -** это именованная область внешней памяти, в которую можно записывать и из которой можно извлечь данные. Правила именования файлов, способ доступа к данным, хранящимся в файле, и структура этих данных зависят от конкретной ***системы управления файлами*** и, возможно, от типа файла.

Отметим здесь лишь, что файловые системы обычно обеспечивают хранение слабоструктурированной информации, оставляя дальнейшую структуризацию прикладным программам.

Следующий этап интеграции данных – это **Базы Данных (БД).**

***База данных* –** поименованная совокупность данных, отражающая состояние объектов (элементов предметной области) и их отношения в рассматриваемой предметной области, организованная по определенным правилам и с определенной целью, и, включающая общие принципы описания, хранения и манипулирования данными.

Уточним элементы этого определения:

* **Совокупность, организованная по определенным правилам** означает, что указанная совокупность включает данные, которые сохра­няются, имеют определенный формат (отформатированы), к ним может быть обеспечен доступ (они доступны), и они могут быть представлены потребителю информации в приемлемом виде.
* **Совокупность, организованная с определенной целью** подчеркивает, что состав данных должен соответ­ствовать некоторой задаче (или классу задач): т.е. с одной стороны, в базе не должны содержаться данные, не имеющие отношения к задаче, а с другой - должна содержаться вся информация, касающаяся задачи.

Прекрасным и понятным примером базы данных может служить телефонный справочник. Он содержит данные, имеющие прямое отношение к цели его существования (в данном случае - имена абонентов), которые открывают доступ к номеру телефона. В справочнике отсутствует избыточная информация, ска­жем цвет аппарата абонента. То есть в этой простейшей базе содержится информация, непосредственно связанная с целью ее создания.

***При использовании ПК*** для решения различных прикладных задач, можно выделить две принципиально разные области их применения:

* **в численных расчетах** – это обычно сложные алгоритмы обработки данных, простые неструктурированные данные или слабоструктурированные данные, сравнительно небольшой объем обрабатываемых данных;
* **в информационных системах** – это обычно относительно простые алгоритмы обработки данных, имеющие сложную структуру и большой объем информации.

**БД** можно классифицировать условно по следующим признакам: по области применения**,** по используемой информации**,** по способу использования,  по цели функционирования, по уровням использования**,** по использованию в сети Internet, по объему хранимых данных.

***Логическую структуру данных*** хранимых в базе данных называют **моделью представления данных**. К основным моделям данных относятся модели: иерархическая, сетевая, реляционная, пост реляционная, многомерная и объектно-ориентированная. Наиболее распространенными являются реляционные модели данных.

Для эффективного управления БД и взаимодействия с пользователями (прикладными программами) между физической базой данных и пользователями информационной системы в современных ИС располагается программное обеспечение – **Система Управления БД (СУБД).**

***СУБД*** - это комплекс языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями. Обычно СУБД различаются по используемым моделям данных. Так, СУБД, основанные на использовании реляционной модели данных, называются реляционными СУБД.

***Приложение (прикладная программа)*** представляет собой программу или комплекс программ, обеспечивающих автоматизацию обработки информации для прикладной задачи. Мы будем рассматривать приложения, использующие БД. Приложения могут создаваться в среде или вне среды СУБД с помощью систем программирования, использующих средства доступа к БД. Например, Delphi, С++ Builder, Visual Basic, С#.

***Пользователей БД*** можно разделить на три группы: прикладные программисты, конечные пользователи, администраторы баз данных (**АБД**). Отметим, что для успешной реализации проекта БД, необходимо ограничить проектируемую систему определенными рамками, в которые будет входить отдельная, четко определенная, совокупность объектов (элементов предметной области) и связей между ними.

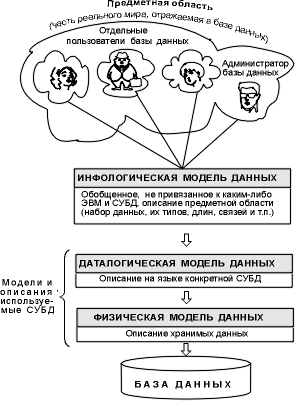
Здесь важно отметить, чтоСУБД должна предоставлять доступ к данным любым пользователям, включая и тех, которые практически не имеют и (или) не хотят иметь представления о физическом размещении в памяти данных и их описаний, о механизмах поиска запрашиваемых данных, о проблемах, возникающих при одновременном запросе одних и тех же данных многими пользователями (прикладными программами), о способах обеспечения защиты данных от некорректных обновлений и (или) несанкционированного доступа, об поддержки баз данных в актуальном состоянии, о множестве других функций СУБД.

При выполнении основных из этих функций СУБД должна использовать различные описания данных.

Естественно, что проект базы данных надо начинать с анализа предметной области и выявления требований к ней отдельных пользователей (сотрудников организации, для которых создается база данных). Объединяя частные представления о содержимом проектируемой базы данных, полученные в результате опроса пользователей, и свои представления о данных, которые могут потребоваться в будущих приложениях, разработчик сначала создает обобщенное неформальное описание создаваемой базы данных. Это описание, выполненное с использованием естественного языка, математических формул, таблиц, графиков и других средств, понятных всем людям, работающих над проектированием базы данных, называют ***инфологической моделью данных*** (рис. 5.1-1).

Остальные модели, показанные на рис. 5.1-1, являются компьютерно-ориентированными. С их помощью СУБД дает возможность программам и пользователям осуществлять доступ к хранимым данным лишь по их именам, не заботясь о физическом расположении этих данных. Нужные данные отыскиваются СУБД на внешних запоминающих устройствах по физической модели данных.

Так как указанный доступ осуществляется с помощью конкретной СУБД, то модели должны быть описаны на языке описания данных этой СУБД. Такое описание, создаваемое по инфологической модели данных, называют ***даталогической моделью данных***.



*Рис. 5.1-1. Уровни моделей данных*

Трехуровневая архитектура (инфологический, даталогический и физический уровни) позволяет обеспечить независимость хранимых данных от использующих их программ.

Как отмечалось ранее, инфологическая модель отображает реальный мир в некоторые понятные человеку концепции, полностью независимые от параметров среды хранения данных. Существует множество подходов к построению таких моделей: модели, основанные на графах, семантические сети, основанные на модели «сущность-связь» и т.д. Наиболее популярной из них оказалась модель «сущность-связь», которая будет рассмотрена ниже.

Инфологическая модель должна быть отображена в компьютерно-ориентированную ***даталогическую модель***, «понятную» СУБД.

Следует различать *создаваемые описания БД* – модели инфологическая, даталогическая и физическая и *модели структур данных*, вид которых определяет и множество различных БД. Такие модели также называют моделями данных. Различают модели иерархическую, сетевую, объектно-ориентированную, реляционную и т.д. Рассмотрим наиболее используемую модель – реляционную.

### 5.2. Основные понятия реляционной модели данных

***Модель данных*** – это некоторая абстракция, в которой отражаются самые важные аспекты функционирования предметной области, а второстепенные – игнорируются.

***Реляционная модель данных*** подразумевает определенный подход при создании данных.

Во-первых, БД представляются на внешнем, не зависящем от структуры ЭВМ уровне в виде совокупности ***двумерных таблиц*** (структурная часть реляционной модели). Такие таблицы в реляционной теории называются ***отношениями (relation)***.

Во-вторых, в реляционной модели подразумевается поддержка данных в ***целостном непротиворечивом состоянии,*** гарантирующем корректность данных (целостная часть реляционной модели).

В-третьих, манипулирование данными реляционной БД осуществляется в соответствии со специально разработанной для этой цели ***реляционными операциями*** (манипуляционная часть реляционной модели).

При проектировании базы данных решаются три основных проблемы. Во-первых, как адекватно отразить предметную область и информационные потребности пользователей в концептуальной модели? Как мы отмечали ранее, эту проблему называют проблемой инфологического проектирования баз данных. Цель инфологического этапа проектирования состоит в получении семантических (смысловых) моделей, отражающих информационное содержание проблемы. Во-вторых, каким образом отобразить объекты предметной области в абстрактные объекты модели данных так, чтобы это отображение не противоречило семантике предметной области, и было, по возможности, наилучшим (эффективным, удобным и т.д.)? Эта проблема известна как проблема логического проектирования баз данных. Цель логического этапа проектирования - организация данных, выделенных на предыдущем этапе, в форму, принятую в выбранной СУБД. И, в-третьих, как обеспечить эффективность выполнения запросов к базе данных, т.е. каким образом, имея в виду особенности конкретной СУБД, расположить данные во внешней памяти, создания каких дополнительных структур (например, индексов) необходимо потребовать и т.д.? Эту проблему называют проблемой физического проектирования баз данных. Цель физического этапа - выбор рациональной структуры хранения данных и методов доступа к ним.

Любые данные, используемые в программировании, имеют свои типы данных. Реляционная модель требует, чтобы *типы используемых данных были простыми*. В реляционной модели данных с понятием тип данных тесно связано понятие домена, которое можно считать уточнением типа данных.

***Домен*** - это семантическое понятие. Домен можно рассматривать как подмножество значений некоторого типа данных, имеющих определенный смысл. 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: .

Если тип данных можно считать множеством всех возможных значений данного типа, то домен напоминает подмножество в этом множестве. Отличие домена от понятия подмножества состоит именно в том, что домен отражает семантику, определенную предметной областью. Может быть несколько доменов, совпадающих как подмножества, но несущих 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 этом проявляется смысловое ограничение доменов.

Фундаментальным понятием реляционной модели данных является понятие ***отношения,*** которое в реляционных БД соответствует понятию ***таблицы****.* Строки таблицы соответствуют ***кортежам*** отношения, столбцы – ***атрибутам***.Причем каждый элемент столбца определяет характеристику (атрибут) некоторой сущности, определяемой соответствующей строкой.

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

***Мощность множества кортежей*** отношения называют ***мощностью*** отношения.

Таким образом, заголовок отношения описывает декартово произведение доменов, на котором задано отношение. Заголовок статичен, он не меняется во время работы с базой данных. Если в отношении изменены, добавлены или удалены атрибуты, то в результате получим уже другое отношение (пусть даже с прежним именем). Тело отношения представляет собой набор кортежей. То есть, тело отношения собственно и является отношением в математическом смысле слова. Тело отношения может изменяться во время работы с базой данных - кортежи могут изменяться, добавляться и удаляться.

**Пример 5.2-1.** **Рассмотрим отношение СОТРУДНИКИ заданное на доменах Номер\_сотрудника, Фамилия, Зарплата, Номер\_отдела.**

Так как все домены различны, то имена атрибутов отношения удобно назвать так же, как и соответствующие домены. Заголовок отношения имеет вид: **СОТРУДНИКИ (Номер\_сотрудника, Фамилия, Зарплата, Номер\_отдела)**.

Пусть в данный момент отношение содержит три кортежа:   
**(1,Иванов, 1000,1), (2, Петров, 2000, 2), (3, Сидоров, 3000, 1).**

Такое отношение естественным образом представляется табл. 5.2-1.

Таблица 5.2-1

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер\_сотрудника** | **Фамилия** | **Зарплата** | **Номер\_отдела** |
| 1 | Иванов | 1000 | 1 |
| 2 | Петров | 2000 | 2 |
| 3 | Сидоров | 3000 | 1 |

***Реляционной базой данных*** называется набор отношений.

***Схемой реляционной базы данных*** называется набор заголовков отношений, входящих в базу данных.

Хотя любое отношение можно изобразить в виде таблицы, нужно четко понимать, что отношения не являются таблицами. Это близкие, но не совпадающие понятия (табл. 5.2-2).

Таблица 5.2-2

|  |  |
| --- | --- |
| **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 данных в ячейках таблицы |
| Первичный ключ | Один или несколько атрибутов |

Рассмотрим основные свойства отношений. В этих свойствах в основном и состоят различия между отношениями и таблицами:

1. В отношении нет одинаковых кортежей. Действительно, тело отношения есть множество кортежей и, как всякое множество, не может содержать неразличимые элементы. Таблицы, в отличие от отношений, могут содержать одинаковые строки.
2. Кортежи не упорядочены (сверху вниз). Действительно, несмотря на то, что мы изобразили отношение **СОТРУДНИКИ** в виде таблицы, нельзя сказать, что сотрудник**Иванов** «предшествует» сотруднику**Петрову**. Причина та же - тело отношения есть множество, а множество не упорядочено. Это вторая причина, по которой нельзя отождествить отношения и таблицы - строки в таблицах упорядочены. Одно и то же отношение может быть изображено разными таблицами, в которых строки идут в различном порядке.
3. Атрибуты не упорядочены (слева направо). Так как каждый атрибут имеет уникальное имя в пределах отношения, то порядок атрибутов не имеет значения. Это свойство несколько отличает отношение от математического определения отношения (компоненты кортежей там упорядочены). Это также третья причина, по которой нельзя отождествить отношения и таблицы - столбцы в таблице упорядочены. Одно и то же отношение может быть изображено разными таблицами, в которых столбцы идут в различном порядке.
4. Все значения атрибутов атомарные. Это следует из того, что лежащие в их основе атрибуты имеют атомарные значения. Это четвертое отличие отношений от таблиц - в ячейки таблиц можно поместить что угодно - массивы, структуры, и даже другие таблицы.

### 5.3. Модель «Сущность-Связь»

Одной из наиболее популярных семантических моделей данных на этапе инфологического проектирования является неформальная модель «**Сущность-Связь**» – **ER**-модель. Моделирование предметной области базируется на использовании графических диаграмм, включающих небольшое число разнородных компонентов. В связи с наглядностью представления концептуальных схем баз данных (и не только их) **ER**-модели получили широкое распространение в **CASE**-системах. **CASE**-система – набор инструментов и методов программной инженерии для проектирования программного обеспечения, который помогает обеспечить высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов. Также под CASE понимают совокупность методов и средств проектирования информационных систем с использованием CASE-инструментов.

Базовыми конструктивными понятиями инфологических моделей в терминах ER**-**модели являются: ***сущность****,* ***экземпляр сущности****,* ***связь*** между сущностями, ***свойства (атрибуты)***сущности, ***домен*** и ***ключ*** сущности**.**

***Сущность*** – это различимый класс однотипных объектов предметной области (класс объектов, который мы можем отличить от другого класса объектов), информация о которых должна быть учтена в модели, а затем храниться в базе данных.

Каждая сущность должна иметь наименование. Сущностью, например, могут быть люди, места, самолеты, рейсы, цвета, студенты, накладные и другие.

***Экземпляр сущности* –** это конкретный представитель данной сущности. Например, представителем сущности **СТУДЕНТ** может быть **Петров И.И*.*** Экземпляры сущностей должны быть различимы, то есть сущности должны иметь некоторые свойства, уникальные для каждого экземпляра этой сущности.

Таким образом, необходимо различать такие понятия, как ***класс (тип)*** сущности и ***экземпляр*** сущности. Понятие тип сущности относится к набору однородных личностей, предметов, событий или идей, выступающих как целое. Экземпляр сущности относится к конкретной вещи в наборе. Например, типом сущности может быть **ГОРОД**, а экземпляром – **Москва, Киев** и т.д.

***Атрибут сущности*** – это именованная характеристика сущности, являющаяся (определяющая) некоторым свойством сущности и принимающая значения из некоторого множества значений. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей. Например, **Цвет**может быть определен для многих сущностей: **СОБАКА, АВТОМОБИЛЬ, ДЫМ** и т.д. Атрибуты используются для определения того, какая информация должна быть собрана о сущности. Примерами атрибутов для сущности **АВТОМОБИЛЬ** являются **Тип, Марка, Номерной знак, Цвет** и т.д. Здесь также существует различие между типом и экземпляром. Тип атрибута ***Цвет*** имеет много экземпляров или значений: *Красный, Синий, Банановый, Белая ночь* и т.д., однако каждому экземпляру сущности присваивается только одно значение атрибута.

***Домен*** – это диапазон допустимых значений, которые может принимать атрибут***.***

***Ключ сущности*** – это не избыточный (минимальный) набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся атрибутам, то есть нарушает его уникальность. Например, для сущности **СОТРУДНИК** ключом является атрибут **Табельныйномер** или набор: **Фамилия*,* Имя*,* Отчество*,*Должность*.*** Необходимо отметить, что сущность может иметь несколько различных ключей.

***Связь*** – это некоторая ассоциация между двумя сущностями, представляющая собой абстракцию набора отношений, которые систематически возникают между различными видами предметов в реальном мире. Одна сущность может быть связана с другой сущностью или сама с собой. Связи позволяют по одной сущности находить другие сущности, связанные с ней.

Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако одно из основных требований к организации базы данных – это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи. А так как в реальных базах данных нередко содержатся сотни или даже тысячи сущностей, то теоретически между ними может быть установлено более миллиона связей. Наличие такого множества связей и определяет сложность инфологических моделей.

Например, связь между сущностями может выражаться следующей фразой – «каждый студент обязан числиться ровно в одной группе».

При построении инфологических моделей в качестве инструмента моделирования будем использовать язык ER-диаграмм (диаграмм «сущность-связь», ER, Entity-Relationship).

Первый вариант модели «сущность-связь» был предложен в середине 1970-х годов Питером Пин-Шен Ченом. К настоящему времени существуют различные способы описания ER-моделей – так называемые НОТАЦИИ.

В данном пособии сущности будем изображать прямоугольниками с наименованиями, атрибуты изображать в пределах прямоугольника, определяющего сущность, причем ключевые атрибуты будем подчеркивать, связи изображаем линией, соединяющей сущности (рис. 5.3-1).

|  |
| --- |
| image335 image336    image337image338 |

*Рис. 5.3-1. Способы описания* ***ER****-моделей*

Каждая связь имеет два конца и одно или два наименования. Наименование обычно выражается в неопределенной глагольной форме: «иметь», «принадлежать» и т.п. Каждое из наименований относится к своему концу связи. Обычно наименования не пишутся ввиду их очевидности.

Каждая связь может иметь один из следующих типов связи   
(рис. 5.3-2).

|  |  |
| --- | --- |
| image339 |  |

*Рис. 5.3-2. Виды связей*

Такой способ изображения связей соответствует нотации Баркера. В нотации IDEF1X мощность связи со стороны «много» изображается черным кружком (рис. 5.3-3).

|  |
| --- |
|  |

*Рис. 5.3-3. Нотация IDEF1X*

***Связь типа один-к-одному*** (**1:1**) означает, что один экземпляр первой сущности (левой) связан с одним экземпляром второй сущности (правой). Связь один-к-одному чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

***Связь типа один-ко-многим*** (**1:M**) означает, что один экземпляр первой сущности (левой) связан с одним или несколькими экземплярами второй сущности (правой).

***Связь типа много-ко-многим*** (**N:M**) означает, что один экземпляр первой сущности (левой) связан с одним или несколькими экземпляром второй сущности (правой), а также, что один экземпляр второй сущности (правой) связан с одним или несколькими экземпляром первой сущности (левой).

При большом объеме, ввод лишь нескольких основных атрибутов в описание значительно усложнит ER-диаграмму. В связи с этим язык ER-диаграмм используется для построения небольших моделей и иллюстрации отдельных фрагментов больших. Чаще же применяется менее наглядный, но более содержательный **язык инфологического моделирования** (**ЯИМ**), в котором сущности и ассоциации представляются предложениями вида:

|  |
| --- |
| **СУЩНОСТЬ (атр\_1, атр\_2, ..., атр\_n)**  **АССОЦИАЦИЯ [СУЩНОСТЬ S1, СУЩНОСТЬ S2,...](атр\_1, атр\_2,..., атр\_n)** |

Атрибуты, входящие в ключ, должны быть отмечены с помощью подчеркивания.

***Инфологическое моделирование*** представляет собой моделирование структуры данных, основываясь на смысле этих данных.

Для выявления связей между сущностями необходимо, как минимум, определить сами сущности. Но это не простая задача, так как в разных предметных областях один и тот же объект может быть сущностью, атрибутом или ассоциацией.

Все связи требуют описания, то есть формализации. Описание должно обеспечивать: идентификацию связи; формулировку имен связи с точки зрения каждой участвующей сущности; вид связи (множественность и условность); формулировку того, как связь была формализована.

Цель формализации связей состоит в том, чтобы позволить установить связь экземпляра одной сущности с экземпляром другой. Это выполняется размещением вспомогательных атрибутов в соответствующих сущностях на модели. Когда это выполнено, говорят, что связь формализована.

Для формализации связи «один к одному» вспомогательные атрибуты могут быть добавлены к любой сущности, но не к обоим.

Для формализации связи «один ко многим» вспомогательные атрибуты должны быть добавлены к сущности на стороне «много».

Для формализации связи «многие ко многим» создают отдельную ассоциативную сущность (сущность-связку), которая содержит ссылки на идентификаторы каждого из участвующих экземпляров. Подобно любой другой, ассоциативная сущность может иметь дополнительные атрибуты и участвовать в связях с другими сущностями

Отметим здесь, что первичный ключ дочерней сущности, как правило, является внешним ключом родительской сущности, а первичный ключ ассоциативной сущности строится из первичных ключей сущностей, связи между которыми она формализует.

**Пример 5.3-1.** **Создать ER-диаграмму (модель) предметной области БИБЛИОТЕКА.**

1. ***Описание предметной области.***

База данных должна содержать данные о книгах в библиотеке. Кроме того, БД должна содержать данные о читателях, а также о факте выдачи книги соответствующему читателю. Одна книга имеет много экземпляров и поэтому может быть выдана многим читателям

1. ***Уточнение задания.***

Данные о книгах: номер книги в библиотеке (код книги), автор, название, количество. Данные о каждом читателе: номер читательского билета (код читателя), ФИО читателя. Данные о факте выдачи: дата сдачи книги в библиотеку.

1. ***Описание сущностей.***

Опишем сущности на языке инфологического моделирования:

**КНИГИ (КодКниги, Автор, Название, Количество)**

**ЧИТАТЕЛИ (КодЧитателя, ФИО)**

1. ***Назначение ключевых атрибутов.***

В сущности,**КНИГИ**ключевой атрибут ***–* КодКниги*,*** в сущности **ЧИТАТЕЛИ** ключевой атрибут – **КодЧитателя**.

1. ***Описание связей.***

Одна книга может быть выдана многим читателям, т.к. она имеет много экземпляров, один читатель может получить много книг. Таким образом, связь между сущностями **КНИГИ** и **ЧИТАТЕЛИ** «много-ко-многим».

1. ***Формализация связей.***

При формализации связи «много-ко-многим» вводится дополнительная сущность-связь. Название такой сущности – **ВЫДАЧА.** Ключи сущностей **КНИГИ** и **ЧИТАТЕЛИ** входят как внешние в сущность **ВЫДАЧА**, и образуют составной ключ. Таким образом, на языке инфологического моделирования описание сущностей выглядит следующим образом:

**КНИГИ (КодКниги, Автор, Название, Количество)**

**ВЫДАЧА [КНИГИ, ЧИТАТЕЛИ](КодКниги, КодЧитателя, ДатаСдачи)**

**ЧИТАТЕЛИ(КодЧитателя, ФИО)**

***Представление ER-модели.***

1. ER-модель разрабатываемой БД представлена на рис. 5.3-4, где PK – это обозначение ключевого параметра.

|  |
| --- |
|  |

*Рис. 5.3-4. ER-модель БД*

Как мы уже обсуждали, процесс проектирования БД представляет собой последовательность переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели.

Этап логического проектирования не заканчивается проектированием схемы отношений, а является лишь его первым этапом, где используются способы «хорошего» или «правильного» проектирования реляционных отношений. Будем считать, что проблема логического проектирования реляционной базы данных состоит в обоснованном принятии решений о том, из каких отношений должна состоять база данных, какие атрибуты должны быть у этих отношений.

Очевидно, что для одной и той же предметной области реляционные отношения можно спроектировать множеством различных способов. Например, можно спроектировать несколько отношений с большим количеством атрибутов, или наоборот, разнести все атрибуты по большому числу мелких отношений. В общем случае проектирование схемы БД может быть выполнено двумя путями:

* путем **декомпозиции** (разбиения), когда исходное множество отношений, входящих в схему БД заменяется другим множеством отношений (число их при этом возрастает), являющихся проекциями исходных отношений;
* путем синтеза, то есть путем компоновки из заданных исходных элементарных зависимостей между объектами предметной области схемы БД.

Классическая технология проектирования реляционных БД связана с теорией нормализации, основанной на анализе функциональных зависимостей между атрибутами отношений. Понятие функциональной зависимости является фундаментальным в теории нормализации реляционных БД и определяет устойчивые отношения между объектами и их свойствами в рассматриваемой предметной области.

### 5.4. Создание БД средствами СУБД Access

Первая версия **Access** была создана в 1993 г.  фирмой Microsoft. **MS Access** - это функционально полная реляционная **СУБД**, работающая в среде **Windows**. **СУБД** **Access** позволяет создавать довольно сложные базы данных, определяет структуры таблиц и связи между ними. СУБД Access обладает также системой создания запросов, отчетов и форм любой сложности. В Access, как любом приложении Windows, можно использовать все возможности обмена данными между приложениями, что позволяет включить в базу данных графическую и (или) звуковую информацию.

В Access база данных включает в себя все объекты, связанные с хранимыми данными (таблицы, формы, отчеты и т.д.). Все объекты Access хранятся в одном файле с расширением. **acсdb (Access 2010).** В таблицах хранятся данные, которые можно просматривать, редактировать, добавлять. Используя формы, можно выводить данные на экран в удобном виде, просматривать и изменять их. Запросы позволяют быстро выбирать необходимую информацию из таблиц. С помощью отчетов можно создавать различные виды документов для вывода на печать, макросы и модули позволяют автоматизировать работу с базой данных.

**В данном пособии используется Access 2010.**

#### 5.4.1. Создание таблиц и работа с таблицами

После запуска **Access** нужно выбрать шаблон   **Новая база данных,** ввести *имя базы* и нажать на кнопке ***Создат*ь.**

Откроется новая база данных, в которой будет создана и открыта в режиме таблицы новая таблица с именем **Таблица1**.

1. ***Создание таблицы в существующей базе данных***

* На вкладке **Файл** наведите указатель мыши на команду ***Открыть*** и выполните одно из указанных ниже действий:
* если нужная база данных есть в списке **Последние базы данных**, выберите ее;
* если базы данных нет в списке **Последние базы данных**, выберите нужный параметр в разделе **Открыть документ.**
* В диалоговом окне **Открыть** выберите нужную базу данных и нажмите кнопку ***Открыть*.**
* На вкладке ***Создание*** в группе **Таблицы**нажмите кнопку ***Таблица***. Новая таблица вставится в базу данных и откроется в режиме таблицы.

Таблицу можно также создать в режиме **Конструктора.** Если выбрать способ создания таблицы – **Конструктор**, то после этого СУБД Access выводит окно **Конструктор таблиц**, в котором задаются имена, типы и свойства полей для создаваемой таблицы.

1. ***Общие сведения о типах полей***

Разобраться в использовании типов данных может быть непросто: например, поля текстового типа могут хранить как текстовые, так и цифровые данные, однако поле с типом данных **Числовой** может содержать только числовые данные. Таким образом, необходимо понимать, какие свойства используются с каждым типом данных. Тип данных поля определяет и другие важные его свойства, например:

* форматы, которые можно использовать в поле;
* максимальный размер значения в поле;
* варианты использования поля в выражениях;
* возможность индексации поля.

Тип данных поля может быть предопределен либо выбираться в зависимости от способа его создания. Например, при создании поля в режиме таблицы имеет место следующее:

* если используется существующее поле из другой таблицы, его тип данных уже определен в шаблоне или другой таблице;
* если данные вводятся в пустой столбец или поле, Access назначает полю тип данных на основе вводимого значения либо тип данных и формат для поля выбирает пользователь;
* если на вкладке ***Изменение полей*** в группе **Поля и столбцы** пользователь выбирает элемент ***Добавление полей***, **Access** отображает список типов данных, в котором можно выбрать нужный вариант.

1. ***Применение тех или иных типов данных***

Тип данных поля можно рассматривать как набор характеристик, применяемых ко всем содержащимся в поле значениям. Например, значения в текстовом поле могут содержать только буквы, цифры и ограниченный набор знаков препинания, а максимальное количество знаков в таком поле – 255.

Иногда фактический тип данных в поле может отличаться от кажущегося типа данных. Например, может показаться, что поле содержит числовые значения, хотя на самом деле они являются текстовыми, такими, например, как номера комнат. Сравнивать значения различных типов и преобразовывать их из одного типа в другой можно с помощью выражений.

В табл. 5.4-1 в приведены форматы, доступные для каждого типа, и описаны эффекты использования параметров форматирования.

Таблица 5.4-1.Основные форматы

|  |  |
| --- | --- |
| **ФОРМАТ** | **СЛУЖИТ ДЛЯ ОТОБРАЖЕНИЯ** |
| **Текст** | Короткие буквенно-цифровые значения, например фамилия или адрес. |
| **Числовой** | Числовые значения, например расстояния. Имейте в виду, что для денежных значений есть отдельный тип. |
| **Денежный** | Денежные значения. |
| **Да/Нет** | Значения, Да и Не, а также поля, содержащие только одно из двух возможных значений. |
| **Дата/время** | Значения даты и времени для годов с 100 по 9999. |
| **Форматированный текст** | Текст или сочетания текста и чисел, которые можно отформатировать с помощью элементов управления цветами и шрифтом. |
| **Вычисляемое**  **поле** | Результаты вычисления. В вычислении должны использоваться поля в той же таблице. Для создания вычислений используется построитель выражений. |
| **Вложение** | Изображения, листы, документы, диаграммы и файлы других поддерживаемых типов, прикрепленные к записям в базе данных (аналогично файлам, вложенным в сообщения электронной почты). |
| **Гиперссылка** | Текст или сочетания текста и цифр, хранящиеся в виде текста и используемые в качестве адреса гиперссылки. |
| **MEMO** | Длинные фрагменты текста. Поле типа MEMO часто используется для хранения подробного описания продукта. |
| **Подстановка** | Список значений, возвращаемых запросом или из таблицы, либо набор значений, указанных при создании поля. Поле подстановки можно создать с помощью мастера подстановок. Такое поле обычно имеет текстовый или числовой тип данных в зависимости от того, какие параметры были выбраны в мастере.  Поля подстановки имеют дополнительный набор свойств, доступных на вкладке **Подстановка** в области **Свойства поля**. |

Числовой тип

|  |  |
| --- | --- |
| **ФОРМАТ** | **СЛУЖИТ ДЛЯ ОТОБРАЖЕНИЯ** |
| **Общий** | Числа без дополнительного форматирования (в том виде, в котором они хранятся). |
| **Денежный** | Денежные значения в общем формате. |
| **Евро** | Общие денежные значения в формате евро. |
| **Фиксированный** | Числовые данные. |
| **Стандартный** | Числовые данные с дробной частью. |
| **Процентный** | Процентные значения. |
| **Экспоненциальный** | Вычисления. |

Дата и время

|  |  |
| --- | --- |
| **ФОРМАТ** | **СЛУЖИТ ДЛЯ ОТОБРАЖЕНИЯ** |
| **Краткий формат даты** | Дата в кратком формате. Зависит от региональных настроек даты и времени (например, 3/14/2001 для США). |
| **Средний формат даты** | Дата в среднем формате (например, 3-Apr-09 для США**).** |
| **Длинный формат даты** | Дата в длинном формате. Зависит от региональных настроек даты и времени (например, Wednesday, March 14, 2001 для США). |
| **Время в формате am/pm** | Время в 12-часовом формате с учетом региональных настроек даты и времени. |
| **Средний формат времени** | Время в формате AM/PM. |
| **Время в 24-часовом формате** | Время в 24-часовом формате с учетом региональных настроек даты и времени. |

Да/Нет

|  |  |
| --- | --- |
| **ТИП ДАННЫХ** | **СЛУЖИТ ДЛЯ ОТОБРАЖЕНИЯ** |
| **Флажок** | Поле флажка. |
| **Да/Нет** | Варианты, Да и Нет. |
| **Истина/ложь** | Варианты Истина" и Ложь. |
| **Вкл/выкл** | Варианты Вкл и Выкл. |

1. ***Дополнительные свойства полей***

После создания поля и выбора его типа данных можно настроить дополнительные свойства поля. Тип данных поля определяет, какие другие свойства можно задать. Например, можно управлять размером текстового поля с помощью его свойства **Размер поля**.

Для числовых и денежных полей свойство **Размер поля** является особенно важным, поскольку определяет диапазон значений поля. Например, одноразрядное числовое поле может содержать целые числа в диапазоне от 0 до 255.

Свойство **Размер поля** также определяет объем места на диске, занимаемого каждым значением числового поля. В зависимости от размера поля число может занимать 1, 2, 4, 8, 12 или 16 байт.

Текстовые поля и поля MEMO имеют переменный размер. Для этих типов данных свойство **Размер поля** задает максимальный объем места, доступный для одного значения.

1. ***Типы данных в связях и соединениях***

***Связь между таблицами — это отношение между общими полями (столбцами) в двух таблицах.*** Существуют следующие разновидности связей: [один-к-одному](javascript:AppendPopup(this,'861468277_3')), [один-ко-многим](javascript:AppendPopup(this,'341285802_4')) и [многие-ко-многим](javascript:AppendPopup(this,'310748866_5')).

Соединение — это операция SQL, объединяющая данные из двух источников в одну запись в [наборе записей](javascript:AppendPopup(this,'804386500_6')) запроса на основе значений указанного общего поля в источниках. Соединение может быть [внутренним](javascript:AppendPopup(this,'183733625_7')), [левым внешним](javascript:AppendPopup(this,'704008842_8')) или [правым внешним](javascript:AppendPopup(this,'302258237_9')).

При создании связи между таблицами или добавлении соединения в запрос связываемые поля должны иметь одинаковые или совместимые типы данных. Например, невозможно создать соединение между числовым и текстовым полями, даже если значения в них совпадают.

В связи или соединении поля с типом данных **Счетчик** совместимы с полями числового типа, если для свойства **Размер поля** последних задано значение **Длинное целое**.

Для поля, используемого в связи между таблицами, невозможно изменить тип данных или свойство **Размер поля**. Чтобы изменить значение свойства **Размер поля**, можно временно удалить связь, однако если изменить тип данных поля, для воссоздания такой связи придется сначала изменить и тип данных связанного поля

1. ***Определение ключевых полей***

При создании многотабличной базы данных важно правильно определить связи между таблицами и задать ключи.

Поля, формирующие связь между таблицами, называются ключами. Ключ обычно состоит из одного поля, однако может включать и несколько. Есть два вида ключей:

* **Первичный ключ**. В таблице может быть только один первичный ключ. Он состоит из одного или нескольких полей, однозначно определяющих каждую запись в этой таблице. Часто в качестве первичного ключа используется уникальный идентификатор, порядковый номер или код. Например, в таблице **Клиенты** каждому клиенту может быть назначен уникальный код клиента. Поле кода клиента является первичным ключом этой таблицы. Если первичный ключ состоит из нескольких полей, он обычно включает уже существующие поля, формирующие в сочетании друг с другом уникальные значения. Например, в таблице с данными о пользователях в качестве первичного ключа можно использовать сочетание фамилии, имени и даты рождения.
* **Внешний ключ**. В таблице может быть один или несколько внешних ключей. Внешний ключ содержит значения, соответствующие значениями первичного ключа другой таблицы. Например, в таблице Заказы каждый заказ может включать код клиента, соответствующий определенной записи в таблице клиентов. Поле кода клиента является внешним ключом таблицы **Заказы**.

Создавая таблицы, Access может автоматически создать первичный ключ, поставив перед первым указанным в структуре полем  поле **Код** и задав для него тип **Счетчик**. Но далеко не всегда такой автоматически определенный ключ бывает корректен. В большинстве случаев ключи должен определять сам разработчик БД.

1. ***Задание или изменение первичного ключа***

* Выберите таблицу, первичный ключ которой требуется задать или изменить;
* На вкладке **Главная** в группе **Режимы** выберите пункт ***Режим*** и команду ***Конструктор***;
* В сетке конструктора таблиц выделите одно или несколько полей, которые необходимо использовать в качестве первичного ключа:
* Чтобы выделить одно поле, щелкните соответствующую [область выделения строки](javascript:AppendPopup(this,'560535107_3'));
* Чтобы выделить несколько полей, щелкните область выделения строки каждого поля, удерживая нажатой клавишу <CTRL>.
* На вкладке **Конструктор** в группе **Сервис** нажмите кнопку ***Первичный ключ***.

1. ***Связи между таблицами***

Хотя в каждой таблице хранятся данные по определенной теме, данные в разных таблицах обычно связаны между собой.

Поскольку данные по разным темам хранятся в отдельных таблицах, их необходимо как-то связать, чтобы можно было легко комбинировать данные из разных таблиц. Для этого используются связи. Связь — это логическое отношение между двумя таблицами, основанное на их общих полях.

1. ***Преимущества использования связей***

Раздельное хранение данных в связанных таблицах обеспечивает указанные ниже преимущества:

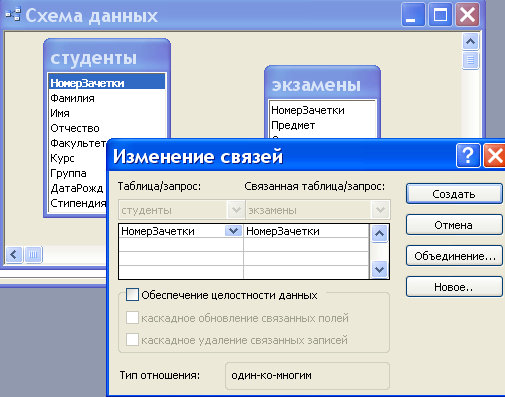
* **Согласованность**. Поскольку каждый элемент данных заносится только один раз в одну таблицу, вероятность появления неоднозначных или несогласованных данных снижается. Например, имя клиента будет храниться только в таблице клиентов, а не в нескольких записях в таблице заказов, которые могут стать несогласованными.
* **Эффективность**. Хранение данных в одном месте позволяет сэкономить место на диске. Кроме того, данные из небольших таблиц извлекаются быстрее, чем из больших таблиц. Наконец, если не хранить данные по различным темам в разных таблицах, возникают пустые значения, указывающие на отсутствие данных, или избыточные данные, что может привести к неэффективному использованию места и снижению производительности.
* Простота. Структуру базы данных легче понять, если данные по различным темам находятся в разных таблицах.
* Связи между таблицами необходимо иметь в виду еще на этапе планирования таблиц. С помощью мастера подстановок можно создать поле внешнего ключа, если таблица с соответствующим первичным ключом уже существует. Мастер подстановок помогает создать связь между таблицами.

1. ***Определение связей между таблицами***

При реализации отношений один-ко-многим и много-ко-многим данные разбиваются на несколько таблиц, определяются ключевые поля, обеспечивающие связь таблиц.  После определения структуры таблиц ***БД*** нужно указать Access способ, которым эти таблицы связаны.

Для определения связей нужно выполнить **Работа с базами данных/Схема данных** и выбрать соответствующие таблицы в окне **Добавление таблицы**. В окне **Добавление таблицы** нужно выделить имена таблиц, добавляемых в схему данных, и нажать на кнопку  
 ***Добавить*.**

В окне **Схема данных** появятся имена всех указанных таблиц вместе со списками полей.  Имена общих полей выделены полужирным шрифтом.  Нужно  выделить ключевое поле ключевой таблицы, нажать кнопку мыши, перетащить ее курсор на аналогичное поле в связываемой таблице, после чего кнопку мыши отпустить. В результате появится диалоговое окно **Изменение связей (**рис.5.4-1**)**.



*Рис. 5.4-1. Определение связей*

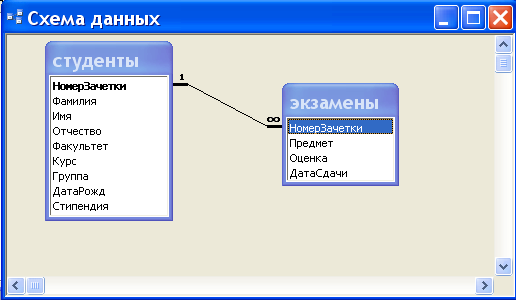
В этом окне Access заполнил первую строку именем поля, по которому связывались таблицы. Чтобы в связанных таблицах не нарушалась целостность данных, нужно щелкнуть по *флажку* ***Обеспечение целостности данных***.  После этого Access  сделает невозможным запись в неключевую таблицу  такого значения общего поля, которого нет в таблице ключевой.

После установления целостности данных можно включить две дополнительные опции: **Каскадное обновление связанных полей** и **Каскадное удаление связанных полей**.

Если выбрать первую опцию, то при изменении какого-либо значения ключевого поля в ключевой таблице Access автоматически обновит значения этого поля для соответствующих записей во всех связанных таблицах. Например, если у одного из студентов изменился номер зачетки в таблице **СТУДЕНТЫ**, то он автоматически должен поменяться и в таблице **ЭКЗАМЕНЫ**.

Выбор второй опции при удалении одной из записей в ключевой таблице приведет к удалению тех записей в таблице со стороны много, которые имеют такое же значение ключа. Например, если из таблицы **СТУДЕНТЫ** удалить запись об одном из студентов, то записи о результатах сданных им экзаменов будут удалены автоматически.

Для завершения процесса создания связей, нужно щелкнуть по кнопке ***Создать***. Accessнарисует линию между таблицами в окне **Схема данных**, указывающую на наличие связи между ними (рис.5.4-2).

******

*Рис. 5.4-2. Схема данных*

#### 5.4.2. Работа с формами

Формы являются основным средством организации интерфейса пользователя в приложениях Access. Хорошо разработанные формы позволяют работать с приложением даже неподготовленному пользователю.

Чаще всего формы создаются в следующих целях:

* ввод и редактирование данных – это наиболее распространенный способ использования форм. Формы обеспечивают вывод на экран данных в удобном для пользователя виде;
* управление ходом выполнения приложения – в этом случае формы используются для запуска макросов;
* вывод сообщений – с помощью форм можно вывести на экран информацию, предупреждение или сообщение об ошибках;
* печать информации – несмотря на то, что для печати информации чаще всего используются отчеты, можно напечатать информацию, содержащуюся в форме.

Формы позволяют задавать два различных набора параметров: один – для вывода формы на экран, а другой – для вывода на печать.

Для создания формы следует в окне базы данных перейти на вкладку Создание и нажать кнопку ***Форма*** или ***Конструктор форм,*** или ***Мастер форм.***

1. ***Создание формы инструментом Форма***

С помощью инструмента **Форма** можно создать форму одним щелчком мыши. При использовании этого средства все поля базового источника данных размещаются в форме. Можно сразу же начать использование новой формы либо при необходимости изменить ее в режиме макета или конструктора.

* В области навигации щелкните таблицу или запрос с данными, которые должны отображаться в форме.
* На вкладке **Создать** в группе **Формы** нажмите кнопку ***Форма***.

Будет создана новая форма и отображена в режиме макета. В режиме макета можно внести изменения в структуру формы при одновременном отображении данных. Например, при необходимости можно настроить размер полей в соответствии с данными.

Если Access обнаруживает одну таблицу, связанную отношением один-ко-многим с таблицей или запросом, который использовался для создания формы, оно добавляет таблицу данных в форму, основанную на связанной таблице или запросе. Например, если создается простая форма, основанная на таблице **Сотрудники**, и между таблицами **Сотрудники и ЗАКАЗЫ** определено отношение "один-ко-многим", то в таблице данных будут отображаться все записи таблицы **ЗАКАЗЫ**, относящиеся к текущей записи сотрудника. Если таблица данных в форме не нужна, ее можно удалить. Если имеется несколько таблиц, связанных отношением "один-ко-многим" с таблицей, которая использовалась для создания формы, то Access не добавляет таблицы данных в форму.

1. ***Создание формы Разделенная форма***

**Разделенная форма** позволяет одновременно отображать данные в двух представлениях — в режиме формы и в режиме таблицы.

Разделенная форма отличается от сочетания формы и подчиненной формы тем, что эти два представления связаны с одним источником данных и всегда синхронизированы друг с другом. При выделении поля в одной части формы выделяется то же поле в другой части. Данные можно добавлять, изменять или удалять в любой части (при условии, что источник записей допускает обновление, а параметры формы не запрещают такие действия).

Разделенная форма позволяет использовать преимущества обоих типов форм в одной форме. Например, можно воспользоваться табличной частью формы, чтобы быстро найти запись, а затем просмотреть или изменить запись в другой части формы.

Чтобы создать разделенную форму инструментом **Разделенная форма**, выполните указанные ниже действия.

* В области навигации щелкните таблицу или запрос с данными, которые должны отображаться в форме, либо откройте таблицу или запрос в режиме таблицы.
* На вкладке **Создать** в группе **Формы** нажмите кнопку ***Форма***.

Accessсоздаст форму и отобразит ее в режиме макета. В режиме макета можно внести изменения в структуру формы при одновременном отображении данных. Например, при необходимости можно настроить размер полей в соответствии с данными.

1. ***Создание формы Несколько элементов***

В форме, созданной с помощью средства **Форма**, одновременно отображается только одна запись. Если нужна форма, в которой отображается сразу несколько записей, и при этом требуются более широкие возможности настройки, чем у таблицы, можно воспользоваться инструментом **Несколько элементов.**

* В области навигации щелкните таблицу или запрос с данными, которые должны отображаться в форме.
* На вкладке **Создание** в группе **Формы** нажмите кнопку ***Другие формы*** и выберите пункт ***Несколько элементов***.

Создаваемая при помощи инструмента **Несколько элементов** форма внешне напоминает таблицу. Данные расположены в строках и столбцах, одновременно отображается несколько записей. Однако форма "Несколько элементов" предоставляет больше возможностей настройки, чем таблица. Например, в нее можно добавлять графические элементы, кнопки и другие элементы управления.

1. ***Создание формы при помощи Мастера форм***

Для получения большей свободы выбора полей, отображаемых на форме, вместо упомянутых выше инструментов можно воспользоваться **Мастером форм**. Кроме того, можно указать способ группировки и сортировки данных, а также включить в форму поля из нескольких таблиц или запросов при условии, что заранее заданы отношения между этими таблицами и запросами. На вкладке **Создание** в группе **Формы** нажмите кнопку **Мастер форм**. Следуйте инструкциям на страницах мастера форм.

Для добавления в форму полей из нескольких таблиц или запросов не нажимайте кнопки ***Далее*** или ***Готово*** после выбора полей из первой таблицы или запроса на первой странице мастера форм. Повторите действия для выбора другой таблицы или запроса и щелкните все дополнительные поля, которые требуется включить в форму. Чтобы продолжить, нажмите кнопку ***Далее*** или ***Готово***. На последней странице мастера нажмите кнопку **Готово**.

1. ***Создание формы инструментом Пустая форма***

Если мастер или инструменты создания форм не подходят, для создания формы можно воспользоваться инструментом **Пустая форма**. Так можно очень быстро построить форму, особенно если на ней будет лишь несколько полей.

* На вкладке **Создать** в группе **Формы** нажмите кнопку ***Пустая форма***. Access откроет пустую форму в режиме макета и отобразит область **Список полей**.
* В области **Список полей** щелкните знак плюс (**+**) рядом с таблицей или таблицами, содержащими поля, которые нужно включить в форму.
* Чтобы добавить поле в форму, дважды щелкните его и перетащите на форму.
* После добавления первого поля можно добавить одновременно несколько полей. Для этого необходимо щелкнуть несколько полей, удерживая при этом нажатой клавишу <CTRL>, а затем одновременно перетащить их на форму.
* Порядок таблиц в области **Список полей** может изменяться в зависимости от того, какая часть формы выделена в текущий момент. Если поле, которое требуется добавить, не отображается, попробуйте выделить другую часть формы и повторите попытку.
* Используя инструменты группы **Колонтитулы** на вкладе **Конструктор**, можно добавить в форму эмблему компании, заголовок или дату и время.
* Для добавления в форму элементов управления других типов используйте инструменты группы **Элементы управления** на вкладке **Конструктор**.

Немного больший выбор элементов управления доступен в режиме конструктора после щелчка формы правой кнопкой мыши и выбора пункта **Режим конструктора**.

#### 5.4.3. Запросы-выборки

При работе с  таблицами можно в любой момент выбрать из базы данных необходимую информацию с помощью  запросов.

***Запрос*** - это обращение к ***БД*** для поиска или изменения в базе данных информации, соответствующей заданным критериям.

Для подготовки запросов чаще всего используются следующие средства описания запросов:

* ***QBE (Query By Example)*** – язык запросов по образцу (используется при визуальном построении запроса);
* ***SQL (Structured Query Language)*** – структурированный язык запросов **SQL** (используется при описании запросов).

Существуют различные виды запросов. Основным является **запрос на вы­борку**. С его помощью выбираются данные из взаимосвязанных таблиц и других запросов, а также представляются результаты в таблице. На основе запроса этого вида строятся запросы других видов:

* **Запросы с параметрами**
* **Перекрестные запросы**
* **Запросы на изменения**
* **Запросы на объединение**
* **Запросы к серверу**
* **Управляющие запросы**
* **Подчиненные запросы**

Таблицу с результатами запроса можно использовать для работы с данными в таблицах, на которых построен запрос. Например, через таблицу запроса мож­но корректировать данные в исходных таблицах базы данных.

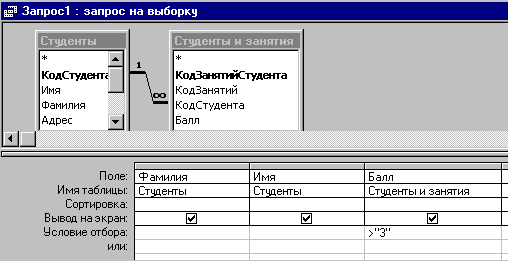
Запрос строится на основе одной или нескольких взаимосвязанных таблиц. При этом могут использоваться таблицы базы данных, а также сохраненные таблицы, полученные в результате выполнения других запросов. Кроме того, запрос может строиться непосредственно на другом запросе с использова­нием его временной таблицы результатов.

Простейшие запросы некоторых видов могут быть созданы с помощью мас­тера.

Так, например, MS Acce**ss** предоставляет удобное для пользователя диалоговое графическое средство формирования запроса **по образцу – QBE   
(Query By Example),** с помощью которого легко может быть построен сложный запрос.

Имена таблиц должны быть представлены в окне **Конструктора запроса** (рис. 5.4-3).

Окно **Конструктора** разделено на две части. В верхней части находятся окна таблиц со списками полей. Имя каждой таблицы отображается в строке заголовка такого окна.



*Рис 5.4-3. Окно конструктора запроса. Пример ввода условия*

Например, на рис. 5.4-3 представлены таблицы **СТУДЕНТЫ** и **СТУДЕНТЫ** и **ЗАНЯТИЯ**.

Нижняя часть является  бланком запроса, или, как его называют,   **QBE** – областью. Здесь указываются параметры запроса и данные, которые нужно отобрать, а также определяется способ их отображения на экране.

При конструировании запроса достаточно добавить таблицы в схему данных запроса, которые содержат необходимые данные, выделить и перетащить требуемые поля из этих таблиц в бланк запроса и, если нужно, ввести условия отбора записей.

В схеме данных запроса отображаются используемые в запросе таблицы со списком полей, а также связи между ними, установленные в схеме данных базы. Первая строка в списке полей, отмеченная звездочкой (\*), обозначает все множество полей таблицы. Кроме связей, внесенных в запрос в соответ­ствии со схемой данных базы, Access автоматически устанавливает связи для объединения таблиц, если таблицы имеют поля, которые содержат данные одинаковых типов с одинаковыми именами. Связи, не установленные программой Access, автоматически может установить пользователь, перетащив задействованные в связи поля из одного списка полей в другой.

Выбранный тип объединения таблиц задается при установлении связи между таблицами и определяет способ формирования записей запроса. По умолча­нию связи устанавливаются с параметром объединения первого типа, т.е. объединения только тех записей, в которых значения связанных полей обеих таблиц совпадают.

При создании запроса можно задать  ***условия (критерии)***, вследствие чего по запросу  будет осуществлен отбор только нужных записей.

Чтобы найти записи с  конкретным  значением, в каком либо  поле, нужно ввести это  значение в данное поле  в строке бланка **QBE** **Условие отбора**.

Рассмотрим примеры условий для запросов на выборку.

В табл.5.4-3 приводятся примеры условий, которые можно использовать в запросах на выборку для сокращения объема возвращаемых данных.

Таблица 5.4-3

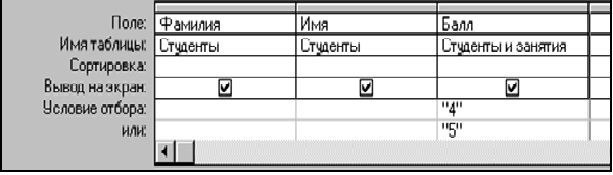
|  |  |
| --- | --- |
| **УСЛОВИЕ** | **РЕЗУЛЬТАТ** |
| **> 234** | Возвращает все числа больше 235. Чтобы найти числа меньше 234, используйте условие <234. |
| **>= "Новосибирск"** | Возвращает все записи от Новосибирск до конца алфавита |
| **Between #2.2.2006# And #1.12.2006#** | Возвращает даты в период между 2 февраля 2006 и 1 декабря 2006 (ANSI-89). Если в базе данных используются подстановочные знаки ANSI-92, вместо знака # следует использовать одинарные кавычки ('). Пример:  **Between '02.02.2006'And**  **'01.12.2006'** |
| **Not "Германия"** | Выполняется поиск всех записей, кроме тех, в которых содержимое поля является точным соответствием значению **Германия**. По этому условию будут возвращены записи, содержащие дополнительные символы, кроме входящих в слово «**Германия**». Например, «**Германия** (евро)» или «**Европа** (Германия)». |
| **Not "Т\*"** | Выполняется поиск всех записей, кроме тех, которые начинаются с буквы «**Т**». Если в базе данных применяются подстановочные знаки ANSI-92, используйте вместо звездочки (**\***) знак процента (**%**). |
| **Not "\*т"** | Выполняется поиск всех записей, кроме тех, которые оканчиваются на «**т**». Если в базе данных применяются подстановочные знаки ANSI-92, используйте вместо звездочки (**\***) знак процента (**%**). |
| **In(Канада,Великобритания)** | Выполняется поиск всех записей списка, содержащих слова «**Канада**» или «**Великобритания**». |
| **Like "[А-Г]\*"** | Находит в поле с типом данных «**Текстовый**» все записи, начинающиеся с букв «**А-Г**». Если в базе данных используются подстановочные знаки ANSI-92, используйте знак процента (**%**) вместо знака звездочки (**\***). |
| **Like "\*ар\*"** | Выполняется поиск всех записей, которые содержат последовательность букв «**ар**». Если в базе данных применяются подстановочные знаки ANSI-92, используйте вместо звездочки (**\***) знак процента (**%**). |
| **Like "Мария Сергиенк?"** | Будут найдены все записи, начинающиеся с имени «**Мария**» и содержащие вторую строку из 9 букв, из которых 8 букв составляют начало фамилии «**Сергиенк**», а последняя буква неизвестна. Если в базе данных применяются подстановочные знаки ANSI-92, используйте вместо знака вопроса знак подчеркивания (**\_**). |
| **#02.02.2006#** | Выполняется поиск записей от **2 февраля 2006**. Если в базе данных используются подстановочные знаки ANSI-92, вместо знака **#** следует использовать одинарные кавычки (**'**) (**'2.2.2006'**) |
| **< Date( ) - 30** | Возвращает все даты, которые более, чем на 30 дней раньше текущей даты |
| **Date()** | Возвращает все записи, содержащие текущую дату |
| **Between Date() And DatAdd("M",3,Date())** | Возвращает все записи для временного интервала между текущей датой и датой, отстоящей от нее на три месяца. |
| **Is Null** | Возвращает все записи, содержащие пустое (незаполненное или неопределенное) значение. |
| **Is Not Null** | Возвращает все записи с любым (не пустым) значением. |
| **""** | Возвращает все записи с пустой строкой. Пустые строки используются, когда требуется добавить значение в обязательное поле, но фактическое значение еще неизвестно. Например, требуется ввести в поле номер факса, но у некоторых клиентов нет факсимильного аппарата. В этом случае вместо того, чтобы вводить число, следует ввести прямые кавычки без пробела (**""**). |

Критерии, устанавливаемые в **QBE** – области, должны быть  заключены  в кавычки. Если **Access** идентифицирует введенные символы как критерии отбора, то заключает их в кавычки автоматически, а если нет, то сообщает о синтаксической ошибке.

Для создания запроса  с несколькими критериями пользуются различными операторами.

Можно задать несколько условий отбора, соединенных логическим  оператором или  (**or**), для некоторого поля одним из двух способов:

* Ввести все условия в одну ячейку строки **Условие отбора**, соединив их логическим оператором **или (or)**. В этом случае будут выбраны данные, удовлетворяющие хотя бы одному из условий (Например, запись   **4 or 5** будет соответствовать тому, что будут выбраны фамилии с оценками **4 или5**).
* Ввести второе условие в отдельную ячейку строки  **ИЛИ**. И если используется несколько строк **ИЛИ**, то чтобы запись была выбрана, достаточно выполнения условий, хотя бы в одной из строк **ИЛИ**, как, например, показано на рис. 5.4-4.



*Рис 5.4-4. Пример записи условия с использованием* ***or***

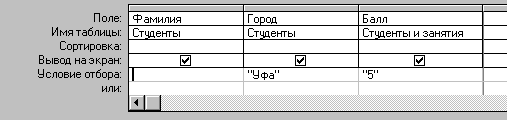
При такой записи условия также будут выбраны фамилии с оценками   
**4 или 5**.

Логическая операция **И (and)** используется в том случае, когда должны быть выполнены оба условия и только в этом случае запись будет выбрана.

Например, записав условие   **>2 and <5** будут выбраны только оценки   
**3 и 5*.***

Чтобы объединить несколько условий отбора оператором **и (and)**, следует привести их в одной строке.

Например, на рис. 5.4-5 показано, как  можно задать условие для выбора фамилий студентов, проживающих в городе **Уфе**, которые учатся только на **5**.



*Рис. 5.4-5. Пример записи условия с использованием оператора* ***and***

Исключить группу данных из состава анализируемых запросом записей   позволяет следующий критерий:   **< > 5.**

В этом случае можно не использовать кавычки.

Операторы **и** и **или** применяются как отдельно, так и в комбинации. Следует помнить, что условия связанные оператором **и*,***  выполняются раньше условий, объединенных оператором **или**.

Можно ввести дату и время,  при этом значения должны быть заключены между символами **#**.  Например:

**#10 мая 1998#**

**>#31.12.96#**

В Access используется ряд других функций, которые помогут задать условия отбора для даты и времени, например:

**Day(***дата***)**  – возвращает значение дня месяца в диапазоне от 1 до 31;

**Month(***дат***а)**  – возвращает значение месяца года от 1 до 12;

**Year(***дата***)**  – возвращает значение года в диапазоне от 100 до 9999.

Данные можно упорядочить  по возрастанию или убыванию. Для упорядочения отобранных записей целесообразно воспользоваться возможностями сортировки самого запроса в строке **Сортировка окна конструктора запроса**.

Можно задать вычисления над любыми полями таблицы и сделать ***вычисляемое значение*** новым полем в запросе.

Для этого в строке Поле  бланка **QBE** вводится формула для вычисления, причем имена полей заключаются в  квадратные скобки.

Например:

**=[Оклад]\*0.15**

В выражениях можно использовать следующие операторы:

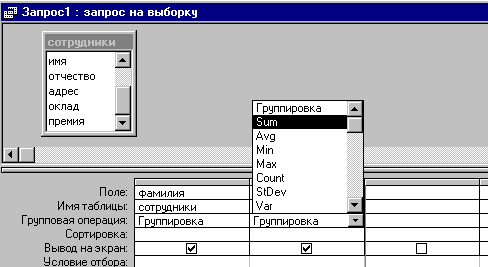
* арифметические: **\*** умножение; **+** сложение; - вычитание; **/** деление; **^** возведение в степень;
* соединение частей текста при помощи знака **&.**

***Итоговые запросы*.** Итоговые запросы значительно отличаются от обычных. В них поля делятся на 2 типа:

* поля, по которым осуществляется группировка  данных;
* поля, для которых проводятся вычисления.

Для составления итогового запроса, находясь в режиме конструктора, следует нажать кнопку **Групповые операции** Access14  на панели инструментов  или воспользоваться командой ***Групповые операции*** из меню **Вид**.

В результате чего в бланке запроса появится строка **Групповая операция**. Если  для соответствующего поля из списка выбрать функцию **Группировка** (рис. 5.4-6), то при выполнении запроса записи по этому полю группируются по значениям в этом поле, но итог не подводится.



*Рис. 5.4- 6. Строка* ***Групповая операция*** *в бланке* ***QBE***

Access предоставляет ряд функций, обеспечивающих выполнение групповых операций. Можно задать нужную функцию, набрав на клавиатуре ее имя в строке Групповая операция, или выбрав  ее из раскрывающегося списка.

Основные  групповые функции, которыми можно воспользоваться:

**SUМ** - вычисляет сумму всех значений заданного поля (для числовых

или денежных полей), отобранных запросом;

**AVG** - вычисляет среднее значение в тех записях определенного поля,

которые отобраны запросом (для числовых или денежных по

лей);

**MIN** - выбирает минимальное значение в записях определенного поля,

отобранных запросом;

**MAX** – выбирает максимальное значение в записях определенного поля,

отобранных запросом;

**COUNT** – вычисляет количество записей, отобранных запросом

**FIRST** - определяет первое значение в указанном поле записей;

**LAST** - определяет последнее значение в указанном поле записей.

Запросы можно создавать для отбора данных как из одной, так и из нескольких таблиц. Запросы к нескольким таблицам производятся аналогично запросам к однотабличным БД  с той лишь разницей, что в окно конструктора запроса добавляются все таблицы, данные которых нужны в запросе.  При этом следует учитывать наличие связей между таблицами.

#### 5.4.4. Язык SQL. Оператор Select

Фактическим стандартом доступа к реляционным данным стал язык **SQL (Structured Query Language).** Язык **SQL** представляет собой смесь операторов реляционной алгебры и выражений реляционного исчисления, использующий синтаксис, близкий к фразам английского языка и расширенный дополнительными возможностями, отсутствующими в реляционной алгебре и реляционном исчислении.

Все СУБД, претендующие на название «реляционные», реализуют тот или иной диалект **SQL**. Многие не реляционные системы также имеют в настоящее время средства доступа к реляционным данным. Целью стандартизации является переносимость приложений между различными СУБД.

Нужно заметить, что в настоящее время ни одна система не реализует стандарт SQL в полном объеме. Кроме того, во всех диалектах языка имеются возможности, не являющиеся стандартными. Таким образом, можно сказать, что каждый диалект – это надмножество некоторого подмножества стандарта SQL.

Язык SQL оперирует терминами, несколько отличающимися от терминов реляционной теории. Например, вместо «отношений» используются «таблицы», вместо «кортежей» – «строки», вместо «атрибутов» - «колонки» или «столбцы».

Стандарт языка SQL, хотя и основан на реляционной теории, но во многих местах отходит он нее. Например, отношение в реляционной модели данных не допускает наличия одинаковых кортежей, а таблицы в терминологии SQLмогут иметь одинаковые строки. Имеются и другие отличия.

Однако язык SQL является реляционно полным. Это означает, что любой оператор реляционной алгебры может быть выражен подходящим оператором SQL.

Основу языка SQL составляют операторы, условно разбитые на несколько групп по выполняемым функциям.

Можно выделить следующие группы:

* **Операторы DDL (Data Definition Language)** – операторы определения объектов базы данных (их мы рассматривать не будем);
* **Операторы защиты и управления данными** (их мы рассматривать не будем);
* Операторы **DML (Data Manipulation Language)** – **операторы манипу-лирования данными:**
* **SELECT - отобрать строки из таблиц;**
* **INSERT - добавить строки в таблицу;**
* **UPDATE - изменить строки в таблице;**
* **DELETE - удалить строки в таблице.**

Кроме того, есть группы операторов установки параметров сеанса, получения информации о базе данных, операторы статического **SQL**, операторы динамического **SQL**.

Наиболее важными для пользователя являются операторы манипулирования данными (DML), а наиболее часто встречаемым оператором среди них – оператор **Select**.

Не все операторы являются независимыми, т.е. некоторые из этих операторов могут быть выражены через другие реляционные операторы.

Кроме основных реляционных операторов в реляционной алгебре активно используются вспомогательные операторы, например операторы переименования атрибутов, агрегирования, расширения.

Поясним необходимость вспомогательных операторов. Некоторые реляционные операторы (например, объединение) требуют, чтобы отношения имели одинаковые заголовки. Действительно, отношения состоят из заголовка и тела. Операция объединения двух отношений есть просто объединение двух множеств кортежей, взятых из тел соответствующих отношений. Но будет ли результат отношением? Во-первых, если исходные отношения имеют разное количество атрибутов, то, очевидно, что множество, являющееся объединением таких разнотипных кортежей нельзя представить в виде отношения. Во-вторых, пусть даже отношения имеют одинаковое количество атрибутов, но атрибуты имеют различные наименования. Как тогда определить заголовок отношения, полученного в результате объединения множеств кортежей? В-третьих, пусть отношения имеют одинаковое количество атрибутов, атрибуты имеют одинаковые наименования, но определены на пересечении различных доменов. Тогда снова объединение кортежей не будет образовывать отношение.

Будем называть отношения совместимыми по типу, если они имеют идентичные заголовки, а именно:

* отношения имеют одно и то же множество имен атрибутов, то есть для любого атрибута в одном отношении найдется атрибут с таким же наименованием в другом отношении;
* атрибуты с одинаковыми именами определены на одних и тех же доменах.

Некоторые отношения не являются совместимыми по типу, но становятся таковыми после некоторого переименования атрибутов. Для того чтобы такие отношения можно было использовать в реляционных операторах, и вводится вспомогательный оператор переименования атрибутов.

В **MS Access** запросы можно строить и на языке **SQL**. В этом языке оператор **Select** является фактически самым важным для пользователя и самым сложным оператором **SQL**. Он предназначен для выборки данных из таблиц, т.е. он, собственно, и реализует одно их основных назначений базы данных - предоставлять информацию пользователю.

Оператор **Select** всегда выполняется над некоторыми таблицами, входящими в базу данных. Здесь необходимо отметить, что на самом деле в базах данных могут быть не только постоянно хранимые таблицы, но и временные таблицы, и так называемые представления. Представления - это просто хранящиеся в базе данных запросы, т.е. **Select**-выражения. С точки зрения пользователей представления – это таблица, которая не хранится постоянно в базе данных, а "возникает" в момент обращения к ней. С точки зрения оператора **Select** и постоянно хранимые таблицы, и временные таблицы и представления выглядят совершенно одинаково.

Результатом выполнения оператора **Select** всегда является таблица. Таким образом, по результатам действий оператор **Select** похож на операторы реляционной алгебры. Любой оператор реляционной алгебры может быть выражен подходящим образом сформулированным оператором **Select**. Необходимо обратить внимание и на то, что выполнение всех запросов в MS Access происходит в SQL.

Выборка данных осуществляется с помощью оператора **Select**, который является самым часто используемым оператором языка SQL:

|  |
| --- |
| **SELECT [ALL/DISTINCT] <***список атрибутов***>/\***  **FROM <***список таблиц***>**  **[WHERE <***условие выборки***>]**  **[ORDERBY <***список атрибутов***>]**  **[GROUPBY <***список атрибутов***>]**  **[HAVING <***условие***>]**  **[UNION <***выражение с оператором* **SELECT>]** |

В квадратных скобках указываются элементы, которые могут в запросе отсутствовать.

Ключевое слово **ALL** означает, что результатом будут все строки, удовлетворяющие условию запроса, в том числе и одинаковые строки, **DISTINCT** означает, что в результирующий набор не включаются одинаковые строки. Далее идет список атрибутов исходной таблицы, которые будут включены в таблицу-результат.

Символ **\*** означает, что в таблицу-результат включаются все атрибуты исходной таблицы.

Обязательным ключевым словом является слово **FROM**, за ним следуют имена таблиц, к которым осуществляется запрос.

В предложении с ключевым словом **WHERE** задаются **условия выборки строк** таблицы. В таблицу-результат включаются только те строки, для которых условие, указанное в предложении **WHERE**, принимает значение истина.

Ключевое слово **ORDER BY** задает операцию упорядочения строк таблицы-результата по указанному списку атрибутов.

В предложении с ключевым словом **GROUP BY** задается список атрибутов группировки (разъяснение этого и последующего ключевого слова будет представлено немного позднее).

В предложении **HAVING** задаются условия, накладываемые на каждую группу (отбора групп).

Отдельно отметим, что ключевые слова **FROM, WHERE ORDER BY** используются аналогичным образом и в других операторах манипулирования данными языкаSQL**.**

Рассм**о**трим реализацию запросов для конкретного примера   
(рис. 5.4-7):



*Рис. 5.4-7. ER-схема БД*

Здесь три таблицы: студент (код студента, фамилия), оценки (код студента, код экзамена, оценка) и экзамен (код экзамены, предмет, дата).

|  |
| --- |
| **student (id\_st, surname)**  **exam\_st (id\_ex, subject, date)**  **mark\_st (id\_ex, id\_st, mark)** |

**Пример 5.4-1. Выдать список всех студентов.**

|  |
| --- |
| **SELECT \* FROM student**  **или**  **SELECT id\_st, surname FROM student** |

Заметим, что если добавить к данному запросу предложение   
**ORDER BY surname**, то список будет упорядочен по фамилии. По умолчанию подразумевается, что сортировка производится по возрастанию. Если необходимо упорядочение по убыванию, после имени атрибута добавляется слово **DESC**.

**Пример 5.4-2. Выдать список оценок, которые получил студент с кодом «1».**

|  |
| --- |
| **SELE**C**T id\_st, mark FROM mark\_st Whereid\_st = 1** |

**Пример 5.4-3. Выдать список кодов студентов, которые получили на экзаменах хотя бы одну двойку или тройку.**

В предложении **WHERE** можно записывать выражение с использованием арифметических операторов сравнения **(<, >**, и т.д.) и логических операторов (**AND, OR, NOT**), как и в обычных языках программирования.

|  |
| --- |
| **SELECT id\_st, mark FROM mark\_st WHERE (mark>= 2 ) AND ( mark<= 3 )** |

Наряду с операторами сравнения и логическими операторами для составления условий в языке SQL (из-за специфики области применения) существует ряд специальных операторов, которые, как правило, не имеют аналогов в других языках. Вот эти операторы:

**IN** – вхождение в некоторое множество значений;

**BETWEEN** – вхождение в некоторый диапазон значений;

**LIKE** – проверка на совпадение с образцом;

**IS NULL** – проверка на неопределенное значение.

Оператор **IN** используется для проверки вхождения в некоторое множество значений. Так, запрос

|  |
| --- |
| **SELECT id\_st, mark FROM mark\_st WHERE mark IN (2,3)** |

дает тот же результат, что и вышеуказанный запрос (выведет идентификаторы всех абитуриентов, получивших хотя бы одну двойку или тройку на экзаменах).

Того же результата можно добиться, используя оператор **BETWEEN**:

|  |
| --- |
| **SELECT id\_st, mark FROM mark\_st WHERE mark BETWEEN 2 AND 3** |

**Пример 5.4-4. Выдать список всех студентов, фамилии которых начинаются с буквы А.**

В этом случае удобно использовать оператор **LIKE**.

Оператор **LIKE** применим исключительно к символьным полям, и позволяет устанавливать, соответствует ли значение поля образцу. Образец может содержать специальные символы:

**\*** – замещает любой одиночный символ;

**?** – замещает последовательность любого числа символов.

|  |
| --- |
| **SELECT id\_st, surname FROM student WHERE surname LIKE “А?”** |

Очень часто возникает необходимость произвести вычисление минимальных, максимальных или средних значений в столбцах. Так, например, может понадобиться вычислить средний балл. Для осуществления подобных вычислений SQL предоставляет специальные агрегатные функции:

**MIN** – минимальное значение в столбце;

**MAX** – максимальное значение в столбце;

**SUM** – сумма значений в столбце;

**AVG** – среднее значение в столбце;

**COUNT** – количество значений в столбце, отличных от **NULL**.

**Пример 5.4-5.** **Рассчитать среднее среди всех баллов, полученных студентами на экзаменах.**

|  |
| --- |
| **SELECT AVG (mark) FROM mark\_st** |

Естественно, можно использовать агрегатные функции совместно с предложением **WHERE**:

|  |
| --- |
| **SELECT AVG (mark) FROM mark\_st WHERE id\_st = 100** |

Данный запрос вычислит средний балл студента с кодом 100 по результатам всех сданных им экзаменов.

|  |
| --- |
| **SELECT AVG (mark) FROM mark\_st WHERE id\_ex = 10** |

Данный запрос вычислит средний балл студентов по результатам сдачи экзамена с кодом 10.

В дополнение к рассмотренным механизмам язык SQL предоставляет мощный аппарат для вычисления агрегатных функций не для всей таблицы результатов запроса, а для разных значений по группам. Для этого в SQL существует специальная конструкция **GROUP BY**, предназначенная для указания того столбца, по значениям которого будет производиться группировка.

**Пример 5.4-6.** **Вычислить средний балл по всем экзаменам для каждого студента.**

Для этого достаточно выполнить следующий запрос:

|  |
| --- |
| **SELECT id\_st, AVG (mark) FROM mark\_st GROUP BY id\_st** |

Все это, как обычно, может быть совмещено с предложением **WHERE**. При этом, не вдаваясь в тонкости выполнения запроса внутри СУБД, можно считать, что сначала выполняется выборка тех строк таблицы, которые удовлетворяют условиям из предложения **WHERE**, а потом производится группировка и агрегирование.

**Пример 5.4-7.** **Вычислить средний балл по оценкам, полученным на экзамене с кодом 100, для каждого студента.**

|  |
| --- |
| **SELECT id\_st,AVG(mark) FROM mark\_st WHERE id\_ex=100 GROUP BY id\_st** |

Для запросов, содержащих секцию **GROUP BY,** существует важное ограничение: такие запросы могут включать в качестве результата столбцы, по которым производится группировка, и столбцы, которые содержат собственно результаты агрегирования.

Для того чтобы форматировать вывод, существуют различные возможности SQL.

**Пример 5.4-8.** **Допустимым является включение текста в запрос.** Рассмотрим пример того, как это делается:

|  |
| --- |
| **SELECT “Средний балл=”, AVG(mark) FROM mark\_st WHERE id\_ex = 10** |

В результате данного запроса пользователь увидит не просто некоторое число, а число, сопровожденное поясняющим текстом.

В реляционных БД существуют три основные операции извлечения данных: выборка, проекция и слияние. Выборка извлекает поднабор строк, проекция извлекает поднабор столбцов, слияние объединяет данные из двух таблиц (для объединения данных из более двух таблиц используется больше одной операции слияния).

SQL выполняет все эти операции через одно выражение **SELECT**:

|  |
| --- |
| **SELECT “Средний балл=”, AVG(mark) FROM mark\_st WHERE id\_ex = 10** |

Необходимыми элементами являются слова **SELECT** и **FROM**. Операция проекции осуществляется путем указания списка столбцов.

Запишем выражение **SELECT**, предназначенное для извлечения перечня книг с именами авторов и соответствующим количеством экземпляров:

|  |
| --- |
| **SELECT автор, кол FROM книги;** |

Для отображения всех столбцов и строк необходимо записать:

|  |
| --- |
| **SELECT \* FROM книги;** |

Порядок вывода столбцов в списке определяет порядок, в котором они будут располагаться в результате.

Порядок отображения данных можно изменить, используя слово   
**ORDER BY** с указанием имени столбца, на основе данных которого они будут отсортированы в алфавитном порядке (по умолчанию):

|  |
| --- |
| **SELECT автор, название FROM книги ORDER BY название;** |

Если вы хотите, чтобы строки располагались в порядке убывания, необходимо добавить слово **DESC**:

|  |
| --- |
| **SELECT автор, название FROM книги ORDER BY название DESC;** |

Операция выборки в БД отображает поднабор строк из таблицы. Для этого необходимо в слове **WHERE** указать условие, по которому выбираются строки таблицы:

|  |
| --- |
| **SELECT автор, название FROM книги WHERE кол > 10;** |

В данном случае выбираются только те книги, количество экземпляров которых превышает 10:

|  |
| --- |
| **SELECT автор, название FROM книги WHERE автор='Пушкин';** |

Здесь выбираются книги одного автора. Текстовые значения необходимо помещать в одинарные кавычки.

Слово **Like** позволяет выбирать строки, в которых совпадают некоторые символы:

|  |
| --- |
| **SELECT автор, название FROM книги WHERE автор Like “П\*”;** |

Здесь выбираются книги, у которых имена авторов начинаются на букву **П**. Знак **\*** используется как «маска», обозначающая любую комбинацию символов:

|  |
| --- |
| **SELECT автор, название FROM книги WHERE автор Like “П\*” and кол<3;** |

Здесь условие сложное, выбираются только те книги авторов, начинающихся на букву “**П”**, количество экземпляров которых меньше 3.

В слове **WHERE** можно использовать те же виды критериев, которые были описаны выше.

При работе с группами строк (групповые операции) можно получить итоговые значения по группам. К примеру, если вы хотите получить количество экземпляров по каждому автору, используйте следующий запрос:

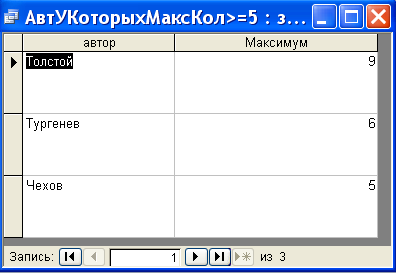
|  |
| --- |
| **SELECT автор, Sum(количество) AS кол FROM книги GROUP BY автор;** |

После слова **AS** мы ввели то имя, которое определяет название столбца, выводимого на экран в результате запроса.

Слово **WHERE** в операторе **SELECT** используется для ограничения количества строк, включенных в результаты запроса. Слово же **HAVING** выполняет похожую функцию, но для групп, а не для отдельных строк. К примеру, если мы хотим получить список всех авторов (если условно считать одной группой одного автора, у которого есть разные книги), у которых максимальное количество экземпляров какой-либо книги больше либо равно 5, то можно использовать следующий оператор:

|  |
| --- |
| **SELECT автор, MAX(количество) AS максимум FROM Книга**  **GROUP BY автор HAVING MAX(количество)>=5;** |

Слово **HAVING** должно быть после слова **GROUP BY**. Слово **HAVING** ограничивает результаты на основе встроенной функции **MAX**, а слово **WHERE** – на значениях отдельной строки. Запомните, что почти всегда, когда вы включаете в запрос слово **HAVING**, вам нужно использовать слово **GROUP BY**. Из результата (рис. 5.4-8) видно, что строки соответствуют группам.



*Рис. 5.4-8. Результат группировки*

При объединении данных из нескольких таблиц реляционные БД используют совпадающие значения первичных и внешних ключей для связи строки одной таблицы со строкой в другой таблице.

Объединение данных из нескольких таблиц в один результирующий набор называется операцией слияния.

В операторе **Select** при связывании таблиц сначала указывается левая связываемая таблица, затем тип связывания, и затем правая таблица. Далее после ключевого слова **ON** указывается условие связывания.

Условие связывания задает логическое условие связывания двух таблиц. Чаще всего используется оператор равенства, например:

|  |
| --- |
| **ON ЧИТАТЕЛИ.КодЧитателя = ВЫДАЧА.КодЧитателя** |

Выдаются только те строки (записи), у которых в таблицах **ЧИТАТЕЛИ** и **ВЫДАЧА** совпадают коды читателей.

Типы связывания могут быть разнообразными. Приведем только некоторые из них.

Если в типе связывания указывается **INNER JOIN**, то в качестве кандидатов на включение в результат запроса рассматриваются только те пары строк, которые удовлетворяют условию связывания в обеих таблицах. Не включаются строки из левой таблицы, для которых не имеются пары в связанной таблице, а также не включаются строки из правой таблицы, для которых нет соответствующей строки в левой таблице.

Если в типе связывания указывается **LEFT JOIN**, то в результат включаются все строки из левой таблицы, даже если для записей из правой таблицы отсутствуют совпадающие значения.

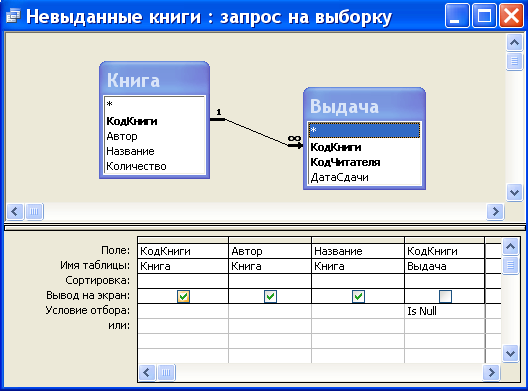
Если в типе связывания указывается **RIGHT JOIN,** то в результат включаются все записи из правой таблицы, даже если нет совпадающих значений с записями из левой таблицы.

**Пример 5.4-9. Построить запросы для** **БД БИБЛИОТЕКА**.

1. **Выбрать книги, которые не выданы читателям.**

Запрос, построенный средствами QBE, представлен на рис. 5.4-9.

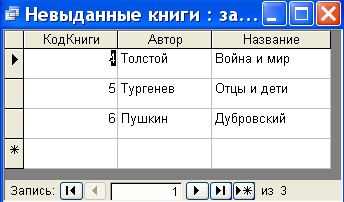
Запрос, сконструированный средствами Access на языке SQL, представлен на рис. 5.4-10. Здесь и далее приводится вид запроса на SQL в формате, принятом в Access.



*Рис. 5.4-9. QBE запроса на выборку*

|  |
| --- |
| **SELECT КНИГА.КодКниги, КНИГА.Автор, КНИГА.Название, КНИГА.Кол**  **FROM КНИГА LEFT JOIN ВЫДАЧА ON КНИГА.КодКниги = ВЫДАЧА.КодКниги**  **WHERE (((ВЫДАЧА.КодКниги) Is Null));** |

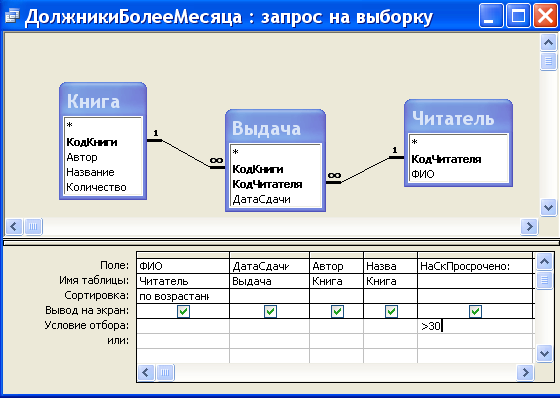
*Рис. 5.4-10. Реализация запроса с использованием SQL*

**

*Рис. 5.4-11. Результат запроса примера 5.4-9*

1. **Выбрать читателей, которые имеют задолженность более месяца.**

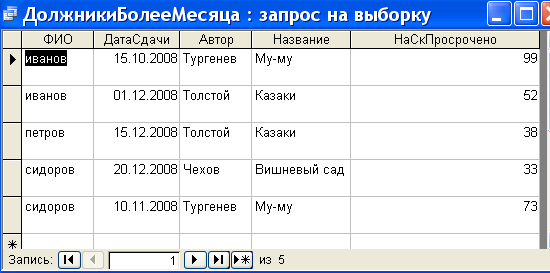
Запрос, построенный средствами QBE**,** представлен на рис. 5.4-12.



*Рис. 5.4-12. QBE запроса ДолжникиБолееМесяца*

|  |
| --- |
| **SELECT ЧИТАТЕЛЬ.КодЧитателя, Читатель.Имя, Выдача.ДатаСдачи,**  **Книга.КодКниги, Книга.Автор, Книга.Название,**  **Date()-Выдача!ДатаСдачи AS НаСкПросрочено**  **FROM Читатель INNER JOIN (Книга INNER JOIN**  **Выдача ON Книга.КодКниги = Выдача.КодКниги)**  **ON Читатель.КодЧитателя = Выдача.КодЧитателя**  **WHERE (((Date()-[Выдача]![ДатаСдачи])>30));** |

*Рис. 5.4-13. Реализация запроса ДолжникиБолееМесяца с использованием SQL*



*Рис. 5.4-14. Результат запроса ДолжникиБолееМесяца*

### [5.5. Задачи для самостоятельного решения](file:///F:\2015УчП-Раз-5-Инф-Шакин-П+Л\Тема-05-05%20ИСПРАВЛЕНО%203_11_2014.docx#_5.5.5._Задачи_для)

Общее задание: создать БД, содержащую информацию в одной таблице. Разработать соответствующий запрос.

|  |  |
| --- | --- |
| № | **Имя и структура таблицы БД и запрос** |
| 1) | **СТУДЕНТЫ: дата проверки, группа, фамилия студента, число пропусков по математике, физике и физкультуре.**  Найти для заданной группы суммарное число пропусков. |
| 2) | **Студенты: дата проверки, группа, фамилия студента, число пропусков по математике, физике и физкультуре.**  Выбрать фамилии студентов, имеющих пропуски занятий в заданные даты. |
| 3) | **Студенты: дата проверки, группа, фамилия студента, дата рождения, число пропусков по математике, физике и физкультуре.**  Выбрать фамилии самых молодых студентов, имеющих суммарно наименьшее число пропусков. |
| 4) | **Студенты: дата проверки, группа, фамилия студента, число пропусков по математике, физике и физкультуре**.  Выбрать даты, в которые пропуски по математике превышали пропуски по физкультуре с указанием фамилии студентов и их групп. |
| 5) | **Вуз: название вуза, название специальности, год приема, план приема, количество принятых студентов.**  Выбрать список вузов, в которых имеется заданная специальность, и план приема не превосходит количество принятых. |
| 6) | **Магазин: название магазина, название товара, стоимость товара, дата продажи.**  Выбрать все товары, не превосходящие заданной стоимости, которые были проданы после заданной даты. |
| 7) | **Сотрудники: табельный номер, должность, год рождения, оклад, фамилия**.  Выбрать самых молодых сотрудников с максимальным окладом. |
| 8) | **Товар: наименование товара, количество на складе, дата поступления, изготовитель**.  Выбрать список товаров заданного изготовителя, количество которых находится в заданных пределах. |
| 9) | **Сотрудники: табельный номер, должность, оклад, фамилия**.  Выбрать сотрудников заданной должности с максимальным окладом. |
| 10) | **Аэропорт: марка самолета, количество мест, средняя стоимость билета, количество проданных билетов.**  Выбрать марки самолетов, для которых количество проданных билетов меньше половины всех мест самолета. |
| 11) | **Магазин: дата покупки, фамилия покупателя, название товара, цена.**  Выбрать покупателей, купивших самый дешевый товар. |
| 12) | **Детали: код детали, название, норма расхода материала на деталь, сорт**.  Выбрать детали первого и второго сорта. |
| 13) | **Предприятие: название предприятия, количество сотрудников предприятия, профессии.**  Выбрать предприятия, имеющие заданные профессии. |
| 14) | **Сотрудники: табельный номер, должность, оклад, фамилия**.  Выбрать сотрудников, средний заработок которых больше заданного. |
| 15) | **Сотрудники: табельный номер, профессия, оклад, фамилия.**  Выбрать сотрудников, сгруппировав их по профессиям. Фамилии для каждой профессии должны быть расположены по алфавиту. |
| 16) | **Автомобили: модель авто, стоимость, цвет кузова, наличие или отсутствие автоматической коробки передач.**  Выбрать автомобили с автоматической коробкой передач, имеющих заданный цвет и стоимость, не превосходящую заданный предел. |
| 17) | **Автомобили: модель, стоимость. год выпуска, страна сборки**.  Выбрать автомобили, упорядочив их по году выпуска. |
| 18) | **Изделия: код изделия, название предприятия, план выпуска, фактический выпуск.** Вывести предприятия, сгруппировав их по наименованиям изделий. Вывести предприятия, перевыполнившие план. |
| 19) | **Изделия: код изделия, наименование изделия, название предприятия, план выпуска, фактический выпуск.**  Вывести изделия, напечатав их по названиям предприятиям, фактический выпуск которых превосходит среднее значение по всем предприятиям. |
| 20) | **Предприятия: название, количество сотрудников, профессии**. Вывести предприятия, количество сотрудников которых находится в заданных пределах. |
| 21) | **Предприятия: название, количество сотрудников, объем выпускаемой продукции, профессии.**  Вывести предприятия, имеющие минимальное количество сотрудников и наибольший объем выпуска. |
| 22) | **Радиолокационные станции: адрес станции, год ввода в эксплуатацию, тип используемой антенны, диапазон.**  Выбрать для каждого диапазона самые старые станции. |
| 23) | **Радиолокационные станции: адрес станции, год ввода в эксплуатацию, тип используемой антенны, диапазон.**  Выбрать станции, сгруппировав их по типам антенн. |
| 24) | **Студенты: год рождения, адрес, дата зачисления, факультет, группа**.  Выбрать самых молодых студентов для каждого факультета. |
| 25) | **Преподаватели: год рождения, адрес, телефон, кафедра, должность, учёная степень.**  Выбрать самых молодых преподавателей, не имеющих ученой степени, и самых пожилых преподавателей с ученой степенью. |
| 26) | **Книги издательства: авторы, название, раздел (техническая, общественно-политическая и т.п.), год издания, количество страниц, цена.**  Выбрать все книги, цена которых выше средней цены для всех книг данного издательства. |
| 27) | **Читатели: номер читательского билета, ФИО, год рождения, адрес, дата записи, вид читателя (студент, аспирант, преподаватель, сотрудник) названия взятых книг и даты их выдачи.**  Определить общее количество книг по каждому виду читателя. |
| 28) | **Продажи: наименования товаров, поставка из магазина или со склада, количество и стоимость товара.** Определить самый дорогой товар из магазина и самый дешевый со склада. |
| 29) | **Блюда: цена, название, категория (мясное, рыбное, салат и т.п.), является ли вегетарианским, время приготовления (в мин.), есть ли в наличии.**  Выбрать все вегетарианские блюда, время приготовления которых меньше заданного. |
| 30) | **Проекты: название, дата начала, дата окончания, размер финансирования, тип финансирования (периодический, разовый), исполнитель**.  Выбрать все проекты по каждому типу финансирования, размер финансирования которых не превышает заданную сумму. |

### 5.6. Лабораторная работа по теме *«Основы проектирования баз данных средствами СУБД ACCESS»*

**Цель** данной работы состоит в получении навыков в проектировании и работе с базой данных (БД) реляционного типа на примере использования СУБД ACCESS.

#### 5.6.1. Вопросы, подлежащие изучению

1. Основные понятия и определения БД.
2. БД реляционного типа.
3. Принципы построения БД реляционного типа.
4. Инфологическое проектирование и модель «Сущность-связь».
5. Создание реляционной БД в среде СУБД Access.
6. Разработка форм и построение запросов к БД.

#### 5.6.2. Общее задание

1. ***Изучите основные понятия, определения и способы работы с БД в среде СУБД******Access (п.п. 5.1 – 5.4)****.*
2. ***Выберите вариант индивидуального задания из табл. 5.6-1.***
3. ***Проанализируйте предметную область проектируемой БД*** *и* ***уточните индивидуальное задание****, определив атрибуты предметной области, которые необходимо хранить в проектируемой* ***БД****.*
4. ***Разработайте модель «Сущность-связь» – ER-модель*** *проектируемой* ***БД*** *(для этого необходимо выявить необходимый набор сущностей (отношений-таблиц), уточнить требуемый набор атрибутов для каждой сущности и задать имена этих атрибутов, определить ключи и связи между сущностями, формализовать связи).*
5. ***Создайте БД средствами СУБД Access******в соответствии с разработанной ER-моделью:***

* ***создайте три таблицы,*** *соответствующие сущностям ER-модели;*
* ***создайте схему БД****, определив две связи типа «один-ко-многим» между двумя основными таблицами и таблицей-связкой, а затем заполните таблицы данными (заполнять необходимо сначала основные таблицы, а затем таблицу сущность-связку в режиме* ***Конструктора таблиц****, причем количество данных в таблицах должно обеспечивать выдачу не менее 3-5 записей по запросу задания);*
* ***создайте форму для ввода данных в БД*** *и с ее помощью* ***введите несколько записей*** *в* ***БД****, а затем проверьте их существование в БД.*

1. ***Создайте запросы либо с помощью Конструктора запросов либо с помощью соответствующего Мастера запросов, а затем их выполните.***
2. ***Просмотрите запросы в режиме SQL.***

#### 5.6.3. Индивидуальные варианты заданий

Таблица 5.6-1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название БД** | **Таблицы** | **Примечание** | **Запрос** |
| 1) | **БАНК** | **БАНК(код банка, название, адрес).**  **КРЕДИТ(код банка, номер клиента, дата выплаты кредита, размер кредита).**  **КЛИЕНТ(номер клиента, ФИО, телефон клиента).** | Банк имеет много клиентов и один клиент может брать кредиты в разных банках. | Выбрать данные по банкам и их клиентам.  Выбрать клиентов, даты выплаты которых просрочены. |
| 2) | **ВУЗ** | **ПРЕПОДАВАТЕЛЬ (номер препод., ФИО, адрес и должность препод., ученая степень).**  **ЗАНЯТИЯ (номер препод., код дисциплины, номер аудитории, дата)**  **ДИСЦИПЛИНА (код дисциплины, название, количество часов).** | Один преподаватель может вести несколько дисциплин и одна дисциплина может вестись несколькими преподава-телями. | Выбрать данные по преподавателям и соответствующим дисциплинам.  Выбрать наиболее трудоемкую дисциплину. |
| 3) | **СКЛАД** | **ТОВАР (код товара, название товара, количество на складе, единица измерения, стоимость единицы товара).**  **ПОСТАВКА (код поставщика, код товара, дата поставки, количество товаров в поставке).**  **ПОСТАВЩИК (код поставщика, ФИО).** | Один и тот же товар может доставляться несколькими поставщикам и один и тот же поставщик может доставлять несколько видов товаров. | Выбрать данные по поставщикам и по поставляемым ими товарам.  Определить самый дешевый товар и соответствующего поставщика. |
| 4) | **ПРОИЗВОДСТВО** | **ИЗДЕЛИЕ (код изделия, название изделия, объем выпуска).**  **СПЕЦИФИКАЦИЯ МАТЕРИАЛОВ (код изделия, код материала, количество материала в спецификации изделия, дата установления спецификации).**  **МАТЕРИАЛ (код материала, название, единица измерения материала, цена за единицу).** | Одно изделие может содержать много типов материалов и один и тот же материал может входить в состав разных изделий. | Выбрать данные по изделиям, в которые входит соответствующий материал.  Вывести список изделий, которые не производились в 2000 г. |
| 5) | **СЕТЬ**  **МАГАЗИНОВ** | **ВЛАДЕЛЕЦ (номер владельца, ФИО, адрес, телефон владельца магазина, номер регистрации, дата регистрации).**  **ВЛАДЕНИЕ (номер владельца, номер магазина, дата регистрации).**  **МАГАЗИН (номер магазина, название, адрес магазина, профиль).** | Один и тот же магазин может иметь несколько владельцев и один и тот же владелец может иметь в собственности много магазинов. | Выбрать данные по владельцам, имеющим собственность в соответствующем районе.  Вывести список профилей магазинов для данного владельца. |
| 6) | **АВТОМАСТЕРКИЕ** | **АВТОМЕХАНИК (номер автомеханика, ФИО).**  **НАРЯД (номер автомобиля, номер автомеханика, название, адрес ремонтной мастерской).**  **АВТОМОБИЛЬ (номер, марка, год выпуска, ФИО автовладельца).** | Один и тот же автомобиль может обслуживаться разными механиками и один и тот же механик может обслуживать несколько автомобилей. | Выбрать механиков, которые работают с автомобилями заданной марки.  Определить тех владельцев автомобилей, которых обслуживает данный механик. |
| 7) | **ДЕКАНАТ** | **ГРУППА (код группы, курс, количество студентов).**  **СЕССИЯ (код группы, код дисциплины, ФИО преподавателя, вид контроля, дата).**  **ДИСЦИПЛИНА (код дисциплины, название дисциплины, количество часов).** | Одна группа изучает несколько дисциплин и одна дисциплина может преподаваться нескольким группам. | Определить преподавателя, который в сессию принимает экзамены у данной группы по данной дисциплине.  Определить группу, у которой число студентов наибольшее. |
| 8) | **ОРГАНИЗАЦИЯ** | **РАБОТА (шифр работы, название, трудоемкость).**  **ПОРУЧЕНИЕ (шифр работы, табельный номер сотрудника, плановая и реальная даты окончаний).**  **СОТРУДНИК (табельный номер, ФИО сотрудника, должность).** | Одна и та же работа может выполняться несколькими сотрудниками и один и тот же сотрудник может участвовать в нескольких работах. | Выбрать данные по работам и выполняющих их сотрудникам.  Определить общее количество работ, находившихся на выполнении у некоторого сотрудника. |
| 9) | **ПОЛИКЛИНИКА** | **ПАЦИЕНТ (номер, ФИО, дата рождения, социальный статус)**  **ЛЕЧЕНИЕ (номер пациента, номер врача, диагноз, срок потери трудоспособности, дата начала лечения).**  **ВРАЧ (номер врача, ФИО, должность, специализация).** | Один и тот же пациент может лечиться у нескольких врачей и один врач может лечить несколько пациентов. | Определить те случаи, когда конкретное заболевание лечилось врачом конкретной специализации.  Вывести имена тех врачей, которые не работают с пенсионерами. |
| 10) | **СПОРТ** | **ВИД СПОРТА (название вида спорта, единица измерения, мировой рекорд и его дата).**  **РЕЗУЛЬТАТ (название вида спорта, код спортсмена, показанный результат, дата проведения, место проведения).**  **СПОРТСМЕН (код спортсмена, ФИО спортсмена, год рождения).** | Один и тот же спортсмен может заниматься разными видами спорта | Вывести виды спорта для каждого спортсмена.  Вывести список спортсменов, превысивших мировой рекорд по данному виду спорта. |
| 11) | **ПОСТАВКИ**  **ПРОДУКЦИИ** | **ПРЕДПРИЯТИЕ (код предприятия, наименование, дата регистрации, вид собственности, число работников, основной вид продукции).**  **ПОСТАВКА (код продукции, код предприятия, дата поставки, объем).**  **ПРОДУКЦИЯ (код продукции, название продукции, единица измерения, закупочная цена).** | Одно и то же предприятие может выпускать разные виды продукции и одна и та же продукция может выпускаться разными предприятиями. | Вывести все данные по конкретному предприятию.  Вывести список продуктов и их дату поставки, для которых закупочная цена ниже заданной. |
| 12) | **ТРАНСПОРТ** | **ВИД ТРАНСПОРТА (название вида транспорта, средняя скорость движения, количество машин в парке, стоимость проезда).**  **МАРШРУТ (номер пути, название вида транспорта, количество остановок на маршруте, количество машин на маршруте).**  **ПУТЬ (номер пути, начальный пункт пути, конечный пункт, расстояние).** | Один и тот же вид транспорта может на разных маршрутах использовать разные пути следования и один и тот же путь может использоваться разными видами транспорта. | Вывести данные по маршрутам трамваев в порядке убывания их протяженности.  Вывести данные по маршруту, у которого максимальное количество видов транспорта. |
| 13) | **НАСЕЛЕНИЕ** **СТРАН** | **СТРАНА (название страны, регион, столица, площадь территории).**  **НАСЕЛЕНИЕ (название страны, название национальности, количество людей данной национальности).**  **НАЦИОНАЛЬНОСТЬ (название национальности, язык, общая численность).** |  | Выбрать данные по численности населения по странам.  Определить столицу той страны, где проживает более всего представителей конкретной национальности. |
| 14) | **ДОМО-**  **УПРАВЛЕНИЕ** | **КВАРТИРА (номер квартиры, номер дома, число жильцов, площадь).**  **ОПЛАТА (номер квартиры, номер дома, вид оплаты, сумма оплаты, дата оплаты).**  **ВИД ОПЛАТЫ (название вида оплаты, цена за единицу площади, цена за одного жильца).** | В одной квартире используются разные виды оплаты. | Выбрать данные по квартирам и соответствующим оплатам.  Определить задолженность по оплате конкретного вида конкретной квартиры. |
| 15) | **АЭРОПОРТ** | **САМОЛЕТ (номер самолета, тип, число мест, скорость полета).**  **РЕЙС (номер самолета, номер маршрута, дата и время вылета, дата и время прилета, количество проданных билетов).**  **МАРШРУТ (номер маршрута, расстояние, пункт вылета, пункт назначения).** | Один самолет может летать на разных маршрутах и по одному маршруту могут летать разные самолеты. | Выбрать данные по самолетам соответствующего рейса.  Выбрать тип самолета, который чаще всего летает по конкретному маршруту. |
| 16) | **КОМПЬЮТЕРЫ** | **ФИРМА (номер фирмы, название и место размещения фирмы).**  **РЫНОЧНОЕ ПРЕДЛОЖЕНИЕ (номер фирмы, номер конфигурации, объем партии, цена партии).**  **КОНФИГУРАЦИЯ ПК (номер конфигурации, тип процессора, тактовая частота, объем ОЗУ, объем жесткого диска, дата выпуска).** | Одна конфигурация ПК может изготавливаться разными фирмами и одна и та же фирма может собирать ПК разной конфигурации. | Выбрать данные по фирмам и их предложениям.  Выбрать модель с наибольшей тактовой частотой, которая выпускается в конкретном городе. |
| 17) | **УЧЕБНЫЙ**  **ПРОЦЕСС** | **СТУДЕНТ (номер зачетной книжки, ФИО студента, дата рождения студента).**  **ОБУЧЕНИЕ (название факультета, номер зачетной книжки, номер группы, сумма стипендии студента, год зачисления).**  **ФАКУЛЬТЕТ (название факультета, число мест).** | Один и тот же студент может обучаться на разных факультетах. | Выбрать данные по студентам соответствующего факультета.  Определить суммарную стипендию конкретной группы. |
| 18) | **ЗООПАРК** | **ВИД ЖИВОТНОГО (название вида животного, суточное потребление корма, семейство, континент обитания).**  **РАЗМЕЩЕНИЕ (название вида животного, номер помещения, количество животных)**  **ПОМЕЩЕНИЕ (номер помещения, наличие водоема, отопления в помещении).** | Один и тот же вид животного может в зоопарке находиться в разных помещениях и в одном помещении может находиться несколько видов животных. | Выбрать данные по животным соответствующего помещения.  Определить общую численность представителей конкретного вида животных. |
| 19) | **ШАХМАТЫ** | **ШАХМАТИСТ (код спортсмена, ФИО, дата рождения, страна, спортивный разряд, рейтинг).**  **УЧАСТИЕ (код спортсмена, код турнира, стартовый номер спортсмена в данном турнире, занятое место).**  **ТУРНИР (код турнира, страна, город, дата проведения, уровень турнира).** | Один шахматист может участвовать в разных турнирах. | Выбрать данные по гроссмейстерам конкретного турнира.  Выбрать турнир с самым высоким рейтингом участников. |
| 20) | **СУДОХОДСТВО** | **КОРАБЛЬ (название корабля, водоизмещение, порт приписки, капитан).**  **ПОСЕЩЕНИЕ ПОРТА (название корабля, название порта, дата посещения порта, дата убытия, номер причала, цель посещения).**  **ПОРТ (название порта, страна).** | Один корабль может посещать несколько портов. | Выбрать список кораблей, посещавших конкретный город в конкретное время.  Определить страны, в которые никогда не приходят корабли с целью  “Туризм” |
| 21) | **НАУЧНЫЕ**  **КОНФЕРЕНЦИИ** | **УЧЕНЫЙ (имя ученого, организация, страна, ученая степень).**  **УЧАСТИЕ (имя ученого, название конференции, тип участия, тема доклада, публикация (да/нет)).**  **КОНФЕРЕНЦИЯ (название конференций, место проведения, дата).** | Один ученый может участвовать в разных конференциях и с разными типами докладов.  Тип участия: доклад, сообщение, стендовый доклад и т.д. | Выбрать данные по ученым конкретной конференции.  Определить количество докторов наук в каждой конференции. |
| 22) | **ПРОГРАММНЫЕ** **ПРОДУКТЫ** | **ПРОГРАММНЫЙ ПРОДУКТ (название программного продукта, версия, фирма, дата выпуска, прикладная область, стоимость лицензии).**  **ИНСТАЛЛЯЦИЯ (название программного продукта, название пользователя, стоимость инсталляции, дата инсталляции, количество лицензий при инсталляции).**  **ПОЛЬЗОВАТЕЛЬ (название пользователя, регион, сфера применения).** | Один и тот же программный продукт может инсталлироваться разными пользователями и один пользователь может инсталлировать разные продукты.  Прикладная область: делопроизводство, коммерция, бизнес и т.д.  Тип программного продукта: ОС, сервер приложений, СУБД и т.п. | Вывести данные по конкретному пользователю и его программным продуктам.  Определить затраты на инсталляции программного обеспечения, сделанные конкретным пользователем |
| 23) | **ОПЕРАЦИОННАЯ** **СИСТЕМА** | **ПРОЦЕСС (идентификатор процесса, идентификатор владельца процесса, название процесса, приоритет, класс).**  **ПЛАНИРОВАНИЕ (идентификатор процесса, название ресурса, запрошено/выделено).**  **РЕСУРС (название ресурса, объем, цена за единицу).** | Один и тот же процесс может задействовать много разных ресурсов и один и тот же ресурс может быть задействован разными процессами.  Примечание:  Классы процессов: критический, серверный, нормальный, запасной. | Вывести данные по ресурсам для каждого процесса.  Определить владельца процесса, у которого "самые большие аппетиты" в ценовом выражении. |
| 24) | **ДОБЫЧА**  **ПОЛЕЗНЫХ**  **ИСКОПАЕМЫХ** | **ПОЛЕЗНОЕ ИСКОПАЕМОЕ (название полезного ископаемого, единица измерения, годовая потребность, цена за единицу).**  **МЕСТОРОЖДЕНИЕ (название полезного ископаемого, название пункта, название месторождения, запасы, способ разработки, годовая добыча).**  **ПУНКТ (название пункта, пропускная способность путей сообщения данного пункта).** | Из одного и того же пункта вывоза могут вывозиться разные полезные ископаемые, одно и то же полезное ископаемое может вывозиться с разных пунктов вывоза. | Определить, какие ископаемые добываются в конкретном месторождении.  Выбрать список месторождений, в которых добывается конкретное ископаемое заданным способом разработки. |
| 25) | **АВТОТРАНС**-**ПОРТНОЕ** **ПРЕД**-**ПРИЯТИЕ** | **АВТОМОБИЛЬ (номер автомобиля, марка автомобиля, его техническое состояние, пробег, расход топлива).**  **РЕЙС (номер автомобиля, табельный номер водителя, дата выезда, дата прибытия, расстояние).**  **ВОДИТЕЛЬ (табельный номер водителя, ФИО, дата рождения, оклад).** | Один и тот же автомобиль может использоваться разными водителями и один и тот же водитель может использовать разные автомобили. | Вывести данные по всем рейсам, в которых участвовал данный автомобиль.  Выбрать автомобиль с наименьшим расходом горючего. |
| 26) | **ТЕАТР** | **АКТЕР (номер актера, ФИО, звание, амплуа, пол).**  **ПОСТАНОВКА НА РОЛЬ (номер актера, название роли, дата назначения на роль, тип роли, режиссер, номер состава);**  **РОЛЬ (название роли, тип роли, название пьесы).** | На одну и ту же роль могут назначаться разные актеры.  Амплуа: герой-любов-  ник, инженю, злодей т.д., тип роли: главная, вторая, эпизод. | Определить список актеров конкретного режиссера.  Выбрать пьесы, в которых исполнителем главной роли является конкретный актер. |
| 27) | **СПРАВОЧНАЯ** **АПТЕК** | **ЛЕКАРСТВО (название лекарства, показания к использованию, противопоказания, производитель).**  **НАЛИЧИЕ (название лекарства, номер аптеки, тип, цена, количество, срок годности).**  **АПТЕКА (номер аптеки, специализация аптеки, район, телефон).** | Тип: таблетки, микстура, мазь и т.д. | Вывести ассортимент предлагаемых лекарств жителям конкретного района.  Определить, в каких аптеках дешевле всего конкретное лекарство. |
| 28) | **КУЛИНАРИЯ** | **БЛЮДО (название блюда, категория, рецепт, вес порции).**  **СОСТАВ (название блюда, название продукта, количество, обработка).**  **ПРОДУКТ (название продукта, категория, калорийность, цена за ед., ед. измерения).** | Одно блюдо состоит из разных продуктов и один и тот же продукт может входить в состав разных блюд. Категории блюда: первое, гарнир, и т.д.,  продукта: мясо, мучное и т.д. | Определить данные по соответствующему блюду.  Определить, для каких блюд продукты данной категории предварительно подвергаются заданной обработке. |
| 29) | **ИЗУЧЕНИЕ**  **СТУДЕНТАМИ ДИСЦИПЛИН ПО ВЫБОРУ** | **СТУДЕНТ (номер зачетной книжки, ФИО студента, адрес, телефон).**  **ВЫБОР (номер зачетной книжки, наименование дисциплины, деканат, адрес деканата, отметка о сдаче дисциплины).**  **ДИСЦИПЛИНА (наименование дисциплины, количество лекционных часов, семинарских и лабораторных занятий).** | Каждый студент должен изучить несколько дисциплин по выбору и каждая дисциплина может изучаться несколькими студентами. | Получить список студентов, изучающих заданную дисциплину и сдавших ее.  Получить список дисциплин, изученных заданным студентом и объем дисциплины. |
| 30) | **ОТДЕЛ** **КАДРОВ** | **СОТРУДНИК (номер сотрудника, ФИО, образование, телефон, дата рождения).**  **ЗАЧИСЛЕНИЕ (номер сотрудника, наименование подразделения, должность, дата зачисления, оклад).**  **ПОДРАЗДЕЛЕНИЕ (наименование подразделения, количество штатных единиц, фонд заработной платы).** | Один и тот же сотрудник может числиться в разных подразделениях | Выбрать сотрудников, которые занимают инженерные должности, не имея высшего образования.  Определить, в каких подразделениях работает данный сотрудник. |

#### 5.6.4. Содержание отчета

1. Тема и название работы.
2. Индивидуальное задание на разработку БД.
3. Формализация и уточнение индивидуального задания.
4. Разработанная модель сущность-связь (ER-модель) БД.
5. Элементы разработанной БД средствами СУБД Access:
   1. Структуры таблиц БД и их содержимое.
   2. Схема БД.
   3. Форма для ввода данных в БД.
   4. Запросы и результаты их выполнения (запросы должны быть

представлены как средствами QBE, так и на языке SQL).

1. Выводы по работе

#### 5.6.5. Пример создания БД «Экзамены»

1. ***Тема и название работы.***

Основы проектирования **БД** средствами **СУБД Access.**

База данных **Экзамены**.

1. ***Задание на разработку БД предметной области Экзамены.***

БД должна содержать список студентов и результаты экзаменов каждого студента.

1. ***Уточнение задания.***

Данные о каждом студенте должны содержать номер зачетки, фамилию, имя, отчество, название факультета, курс, номер группы, дату рождения, размер стипендии. Данные об экзаменах: название предмета, оценка, дата сдачи экзамена.

1. ***Разработка ER-модели БД.***
   1. **Определите Сущности, атрибуты и ключевые атрибуты:**

Опишем сущности на языке инфологического моделирования:

|  |
| --- |
| **СТУДЕНТЫ (НомерЗачетки, Фамилия, Имя, Отчество,**  **Факультет, Курс, Группа, ДатаРождения, Стипендия)**  **ЭКЗАМЕНЫ (Предмет, Оценка, Дата сдачи)** |

В сущности **СТУДЕНТЫ** первичный ключ – **НомерЗачетки**.

* 1. **Определите Связи между сущностями *СТУДЕНТ и ЭКЗАМЕН***

Один (один экземпляр сущности **СТУДЕНТЫ**) может сдавать несколько ***ЭКЗАМЕНОВ***, поэтому один ***СТУДЕНТ*** связан со многими ***ЭКЗАМЕНАМИ***. Один факт сдачи одного ***ЭКЗАМЕНА*** (один экземпляр сущности ***ЭКЗАМЕНЫ***) связан только с одним ***СТУДЕНТОМ***. Таким образом, связь между сущностями **СТУДЕНТЫ** и **ЭКЗАМЕНЫ** «один-ко-многим».

* 1. **Формализуйте Связи.**

Класс принадлежности со стороны сущности **ЭКЗАМЕНЫ** – обязательный (факт сдачи экзамена обязательно должен быть связан с соответствующим студентом из сущности **СТУДЕНТЫ**), поэтому ключ   
**Номер зачетки** сущности **СТУДЕНТЫ** должен присутствовать как внешний ключ (**FK – Foreign Key**) в сущности **ЭКЗАМЕНЫ**. Таким образом, сущность **ЭКЗАМЕНЫ** на языке инфологического моделирования должна быть следующей:

|  |
| --- |
| **ЭКЗАМЕНЫ (НомерЗачетки(FK), Предмет, Оценка, Дата сдачи)** |

ER-модель разрабатываемой БД представлена на рис. 5.7-1.

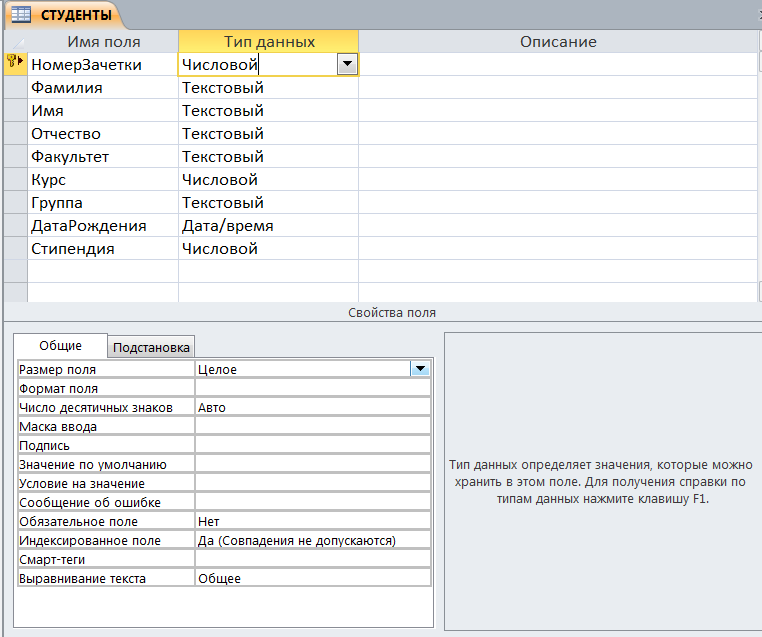


*Рис. 5.6-1****.*** *ER-модель разрабатываемой БД*

1. **Разработка БД средствами СУБД Access:**

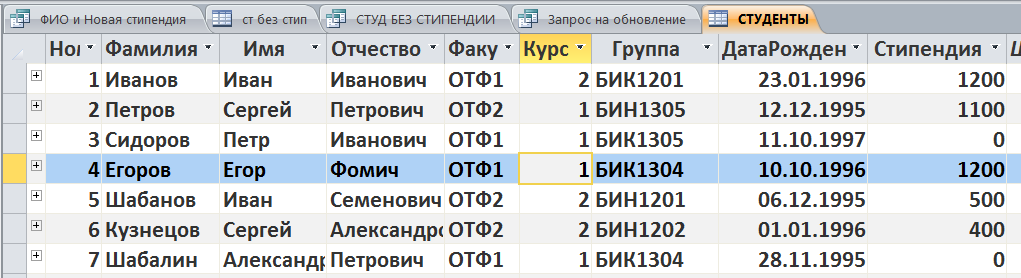
Создайте новую базу данных в СУБД Accessс именем **ЭКЗАМЕНЫ**. Далее создайте структуру первой  таблицы БД, определив ключевое поле и индексы, и затем сохраните таблицу под именем **СТУДЕНТЫ**. Для создания структуры таблицы **СТУДЕНТЫ** рекомендуется использовать режим **Конструктора таблиц**. В бланке **Свойства поля** обязательно укажите длину текстовых полей, формат числовых полей и дат.

Поле **НомерЗачетки** в таблице **СТУДЕНТЫ**  объявите ключевым и индексированным со значением **Совпадения не допускаются** (Рис. 5.6-2).

****

*Рис. 5.6-2. Создание структуры таблицы* ***СТУДЕНТЫ*** *в режиме* ***Конструктора таблиц***

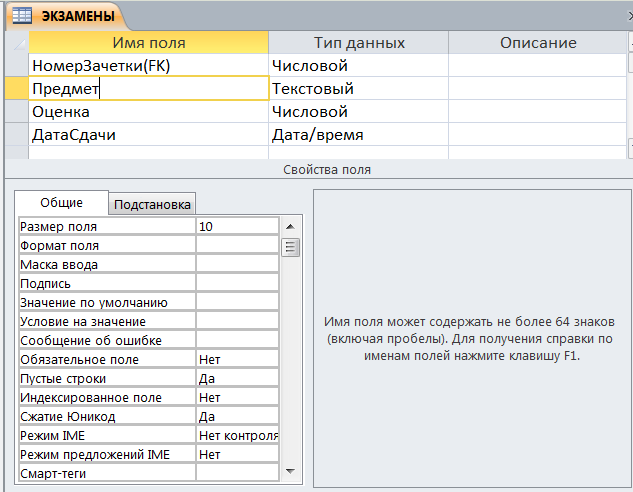
После определения всех полей таблицы закройте окно и сохраните таблицу под именем **СТУДЕНТЫ.** Введите в  таблицу **СТУДЕНТЫ** 10-15 записей и сохраните их (рис. 5.6-3). Для сохранения записей достаточно просто закрыть окно таблицы.



*Рис. 5.6-3. Содержание записей таблицы* ***СТУДЕНТЫ***

Создайте структуру  таблицы **ЭКЗАМЕНЫ** (рис 5.6-4). Обязательно определите формат и длину полей в бланке **Свойства поля**.

Тип поля **НомерЗачетки** определите **Мастером подстановок**, который использует для подстановки, данные из таблицы **СТУДЕНТЫ**. Для поля   
**НомерЗачетки** в **Типе данных** выберите **Мастер подстановок**. В качестве доступных при подстановке полей выберите ***Фамилию*** и ***Имя***.

****

*Рис. 5.6-4. Структура таблицы* ***ЭКЗАМЕНЫ***

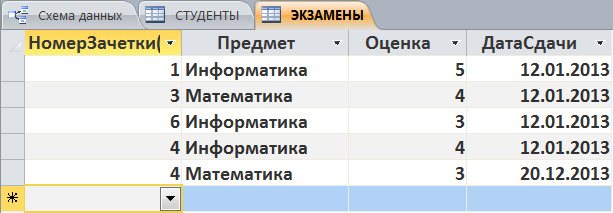
На этапе выбора ширины столбцов необходимо снять галочку **Скрыть ключевой столбец**. Поле **НомерЗачетки** в таблице **ЭКЗАМЕНЫ**  объявите индексированным со значением **Совпадения  допускаются**.

Обратите внимание, что при сохранении структуры неключевой таблицы Access может предупредить об отсутствии ключевого поля и предложит создать это поле сейчас.

После определения всех полей таблицы закройте окно и сохраните таблицу под именем **ЭКЗАМЕНЫ**. Введите в  таблицу **ЭКЗАМЕНЫ** 10-15 записей и сохраните их (рис. 5.6-5).

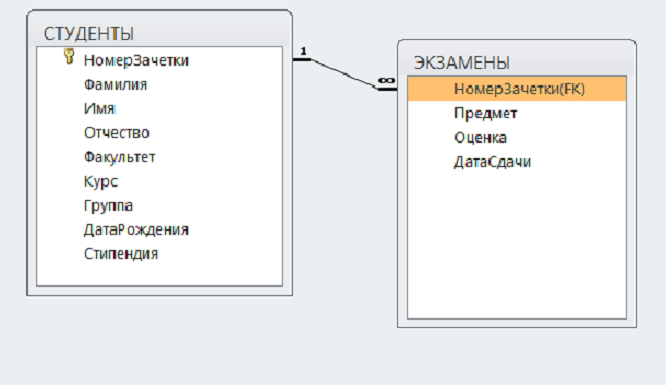
Заполните таблицу **ЭКЗАМЕНЫ** данными в режиме таблицы, используя созданный с помощью **Мастера подстановок** список в поле   
**НомерЗачетки**.

Для сохранения записей достаточно просто закрыть окно таблицы.



*Рис. 5.6-5. Содержимое таблицы* ***ЭКЗАМЕНЫ***

* 1. Вызовите из элемента меню **Работа с базами данных** окно **Схема данных** (Рис. 5.6-6).



*Рис. 5.6-6. Окно* ***Схема данных***

Выделите связь левой кнопкой мыши. Проверьте тип связи (должен быть «один-ко-многим»). Включите переключатель **Обеспечение целостности данных**, включите опции **Каскадное обновление связанных полей и Каскадное удаление связанных полей.**

* 1. Создайте форму при помощи инструмента **Форма**. Перед тем как приступать к созданию формы, определите таблицу, которая будет источником данных для этой таблицы.

Выделите таблицу **ЭКЗАМЕНЫ** и щелкните на кнопке **Форма** в группе **Формы** на вкладке **Создание**. Будет создана форма на основе одной таблицы. Сохраните форму.

Выделите таблицу **СТУДЕНТЫ** и создайте форму аналогично. Будет создана форма на двух таблицах – на таблице **СТУДЕНТЫ** (родительская таблица) и имеющая связь «один-ко-многим» с ней таблице **ЭКЗАМЕНЫ** (дочерняя таблица). Сохраните форму.

Изучите различные виды форм, создаваемых с помощью **Мастера Форм,** для этого создайте формы **В один столбец, Ленточный, Табличный** и сохраните их.

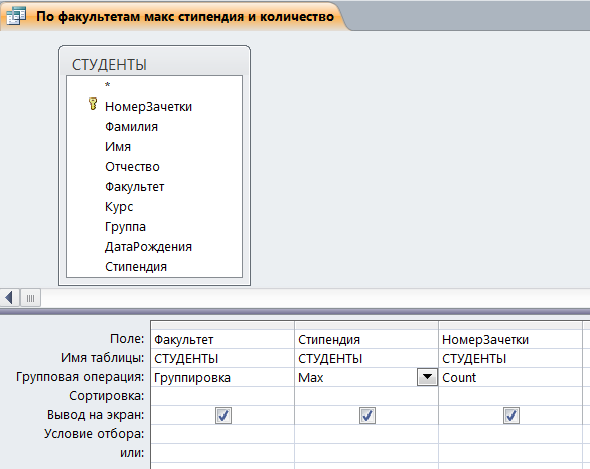
* 1. Создайте запросы на выборку из двух таблиц. Для этого на вкладке **Создание** нажмите на кнопку **Мастер запросов**, выберите **Простой запрос**.  Выберите несколько произвольных полей из таблицы **СТУДЕНТЫ** и **ЭКЗАМЕНЫ,** сохраните запрос и выполните его. Здесь и далее просматривайте запросы в режиме SQL. Для этого в контекстном меню выбирайте **Режим SQL**.
* Создайте с помощью **Конструктора  запросов** запросы, удовлетворяющие условиям:
* выбрать всех студентов одного факультета;
* выбрать фамилии  и имена студентов-отличников одного факультета. В запрос должны быть включены поля **Фамилия*,* Имя*,* Отчество** и те поля соответствующих таблиц, где вводятся критерии.
* выбрать студентов, год рождения которых больше определенной даты (например, **>#01.01.1995#**).

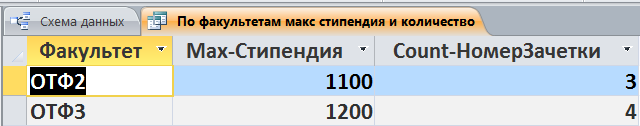
После задания критерия для запроса, запрос выполните и сохраните под именем, подходящим по смыслу.

* Создайте итоговый запрос: оставьте в запросе поля **Факультет**,   
  **Стипендия, НомерЗачетки**, вычислите максимальное значение стипендии для каждого факультета и подсчитайте количество студентов на каждом факультете (используя функции агрегирования **Max** и **Count**).

Для этого в Конструкторе запросов добавьте таблицу **СТУДЕНТЫ**, в список полей включите поля **Факультет*,* СтипендияиНомерЗачетки*.*** На контекстной вкладке (в меню) **Работа с запросами/ Конструктор** нажмите на кнопку ***Итоги***.

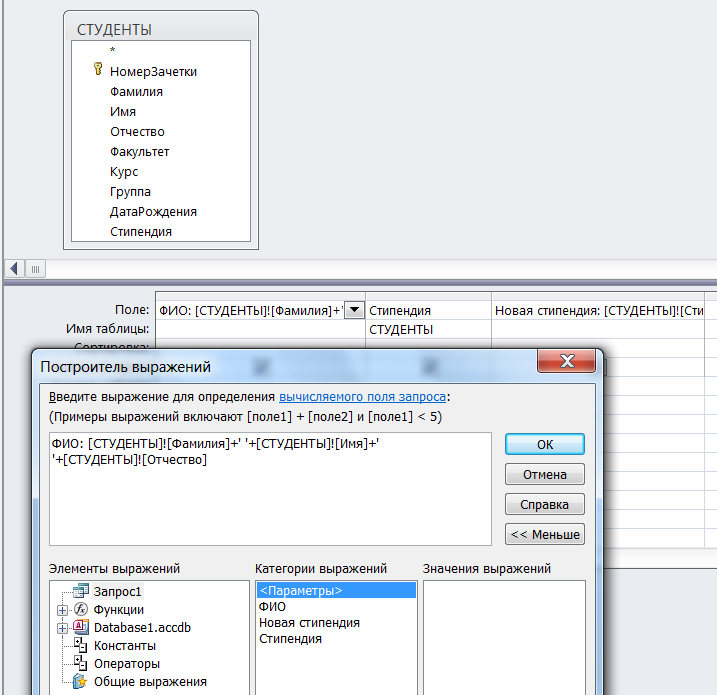
В запрос будет включена **Групповая операция**. В соответствующих полях включите функции **Countи Max** (Рис. 5.6-7).





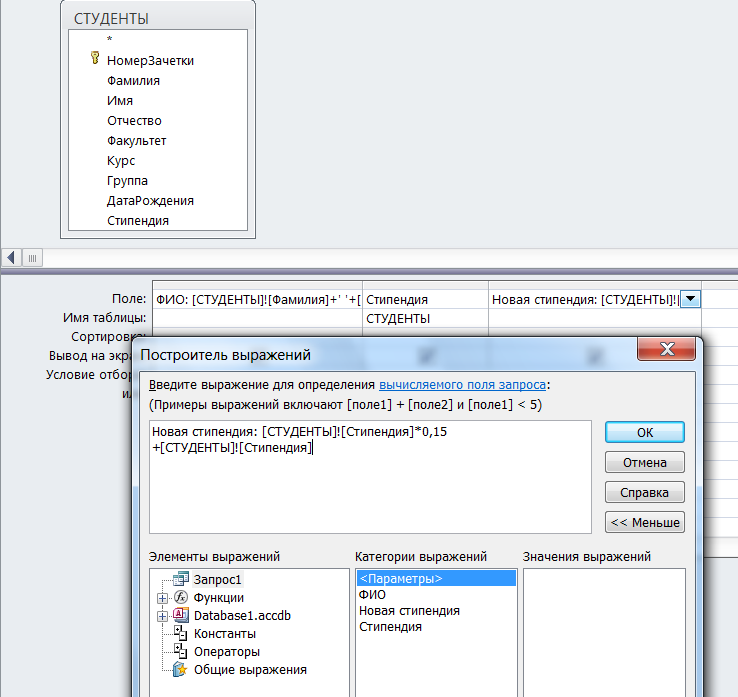
*Рис. 5.6-7. Запрос и результат запроса***По факультетам макс стипендия и количество**

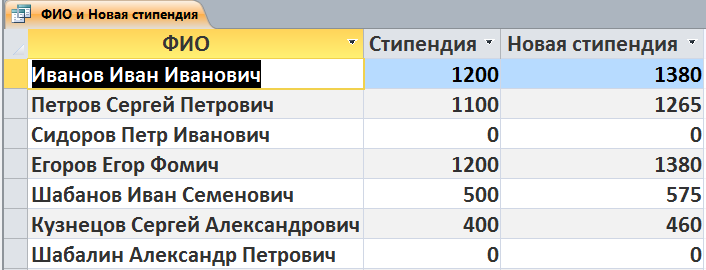
* Создайте запрос с вычисляемыми полями с использованием построителя выражений (Рис. 5.6 -8):
* включите в запрос вычисляемое поле (**ФИО**), которое является результатом сцепления текстовых полей **Фамилия*,* Имя*,* Отчество**.



*Рис. 5.6-8. Создание запроса с вычисляемыми полями*

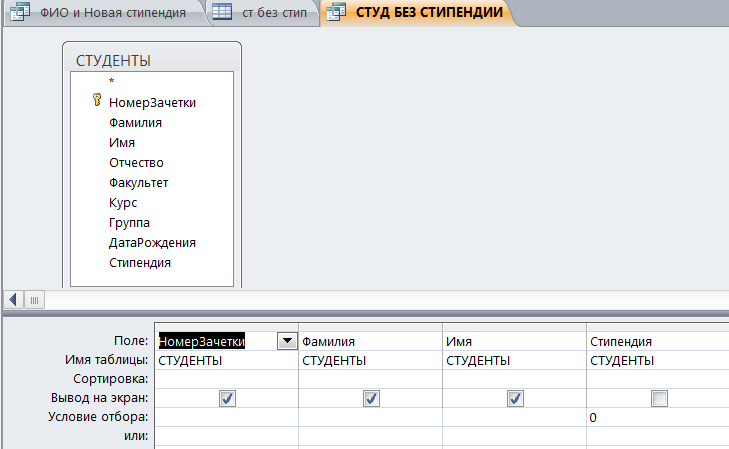
* включите в запрос подсчет надбавки  студентам, равной 15%  от стипендии (Рис. 5.6-9).

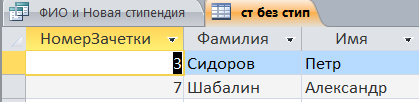




*Рис. 5.6-9. Запрос и результат запроса* **Надбавка**

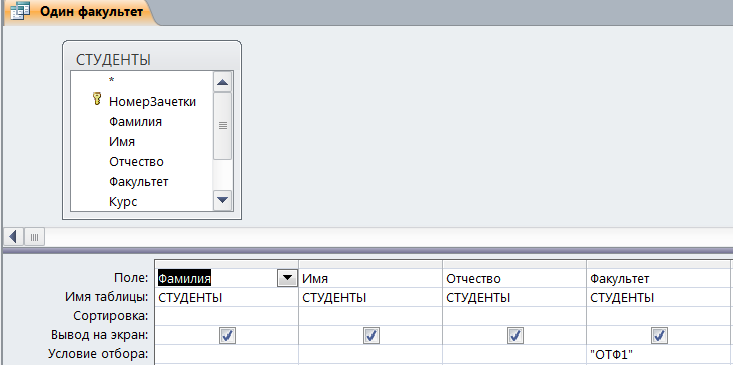
* Сконструируйте запрос-выборку с именем **СТУДЕНТЫ БЕЗ СТИПЕНДИИ** к таблице  **СТУДЕНТЫ**, выбрав тех, кто не получает стипендии. Для этого в поле **Стипендия** поставьте условие отбора – 0 (рис. 5.6-10). Выполните запрос и просмотрите результат, сохраните запрос.

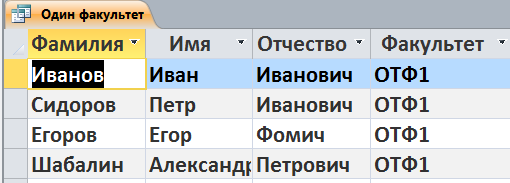




*Рис. 5.6-10. Запрос и результат запроса* **СТУДЕНТЫ БЕЗ СТИПЕНДИИ**

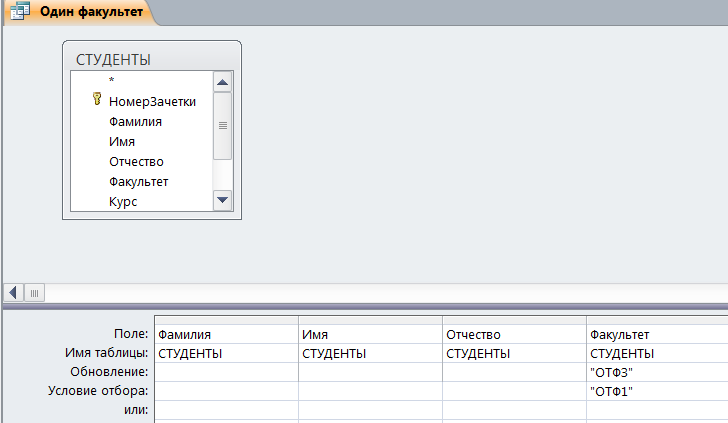
* Преобразуйте его в запрос для создания таблиц (в режиме **Конструктор**) с именем **СТУДЕНТЫ БЕЗ СТИПЕНДИИ**. Для этого нажмите на кнопку **Создание таблицы**, введите имя таблицы. Выполните запрос, подтвердите помещение записей в таблицу. Просмотрите новую таблицу.
* Создайте запрос-выборку всех студентов одного факультета. Выполните запрос, проверьте правильность отбора записей   
  (Рис.5.6-11).

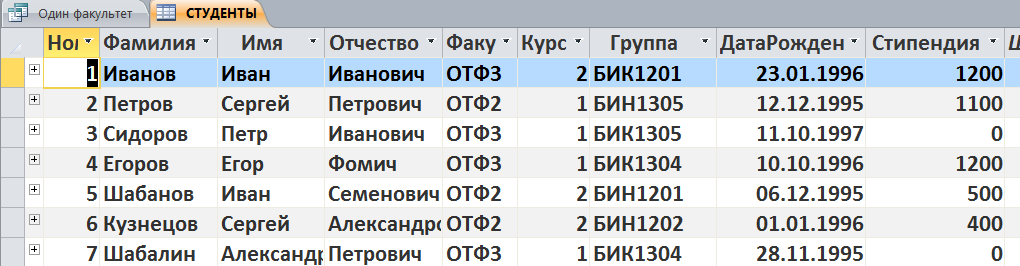




*Рис. 5.6-11. Запрос и результат запроса* ***Один факультет***

* Перейдите в режим **Конструктора** для этого запроса и преобразуйте его в запрос на обновление данных. В строке **Обновление** задать новое значение для поля **Факультет**. Выполнить запрос. Проверьте результат выполнения запроса (Рис. 5.6-12).





*Рис. 5.6-12. Запрос и результат запроса (измененная таблица* ***СТУДЕНТЫ****)*

### 5.7. Контрольные вопросы

1. Как осуществляется фиксация данных, и их интерпретация?
2. Что такое предметная область?
3. Перечислите основные этапы интеграции данных.
4. Что такое модель данных?
5. Перечислите основные модели данных.
6. Что такое СУБД?
7. Что такое домен?
8. Перечислите основные реляционные термины и соответствующие им табличные.
9. Что такое сущность, экземпляр сущности? Приведите примеры.
10. Перечислите основные свойства реляционных таблиц?
11. Что такое модель «Сущность-связь» (ER-модель)?
12. Какие способы изображения связей используются?
13. Перечислите основные этапы разработки ER-модели предметной

области.

1. Типы связей "один-к-одному" и "один-ко-многим": как они реализуются в реляционных базах данных?
2. В каких режимах можно создать структуру таблицы в Access?
3. Что определяется при формализации связей?
4. Что такое ключ и какие требования предъявляются к нему?
5. Как в СУБД Access можно определить ключевое поле?
6. Как и для чего в СУБД Access создается схема данных?
7. Назовите основные типы данных в Access.
8. Как в Access определить связи между таблицами?
9. Как используется Мастер подстановок?
10. Каковы возможности Access по изменению структуры таблиц?
11. Как отсортировать записи по одному полю? По нескольким полям?
12. Как создать простой запрос, итоговый запрос, запрос с вычисляемыми полями, запрос на создание таблицы?
13. Как создать форму в Access?