**Кафедра цифровой экономики**

# БАЗЫ ДАННЫХ

**Методические указания**
**к лабораторным работам №№ 1,2,3,4,5 для студентов направления подготовки 38.03.05 «Бизнес-информатика»**
**1 семестр**

Ульяновск
2017

**ЛАБОРАТОРНЫЙ ПРАКТИКУМ НА ЭВМ**
**по дисциплине "Базы данных"**

Оглавление

# Лабораторная работа №1

## Знакомство с СУБД MS Access

**Цель работы:** изучить и закрепить на практике методы и средства СУБД по корректному заполнению и модификации таблиц БД и методы контроля вводимых данных путем связывания таблиц.
**Результат:** неформализованная, нормализованная инфологическая модель базы данных, моделирующей предметную область согласно задания.
**Теоретическая справка:**

### *Базы данных как средство хранения и обработки информации*

**Базы данных** — это совокупность сведений (о реальных объектах, процессах, событиях или явлениях), относящихся к определенной теме или задаче, организованная таким образом, чтобы обеспечить удобное представление этой совокупности, как в целом, так и любой ее части.

Почти все современные системы основаны на реляционной (relational) модели управления базами данных. ***Реляционная база данных*** представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного типа. Название «реляционная» связано с тем, что каждая запись содержит информацию, относящуюся только к одному объекту. В таких базах данные не дублируются, а связываются по определенным полям.

Можно выделить три основные функции СУБД:

- *определение данных* (Data definition) - вы можете определить, какая именно информация будет храниться в вашей базе данных, задать структуру данных и их тип (например, максимальное количество цифр или символов), а также указать, как эти данные связаны между собой. В некоторых случаях вы можете также задать форматы и критерии проверки данных;
- *обработка данных* (Data manipulation) - данные можно обрабатывать самыми различными способами. Можно объединять данные с другой связанной с ними информацией и вычислять итоговые значения;
- *управление данными* (Data control) - вы можете указать, кому разрешено знакомиться с данными, корректировать их или добавлять новую информацию. Можно также определить правила коллективного пользования данными.

Система управления базами данных Microsoft Access является одним из самых популярных приложений в семействе настольных СУБД.

Запуск Microsoft Access. Создание базы данных

Для того, чтобы запустить Microsoft Access необходимо:

1. Нажать кнопку *Пуск* на Панели задач в нижней части рабочего стола.
2. Открыть в главном меню пункт *Программы*.
3. Выбрать программу *Microsoft Access*.

Выбрать пустой шаблон действие 1 задать имя файла действие 2 и нажать кнопку создать.

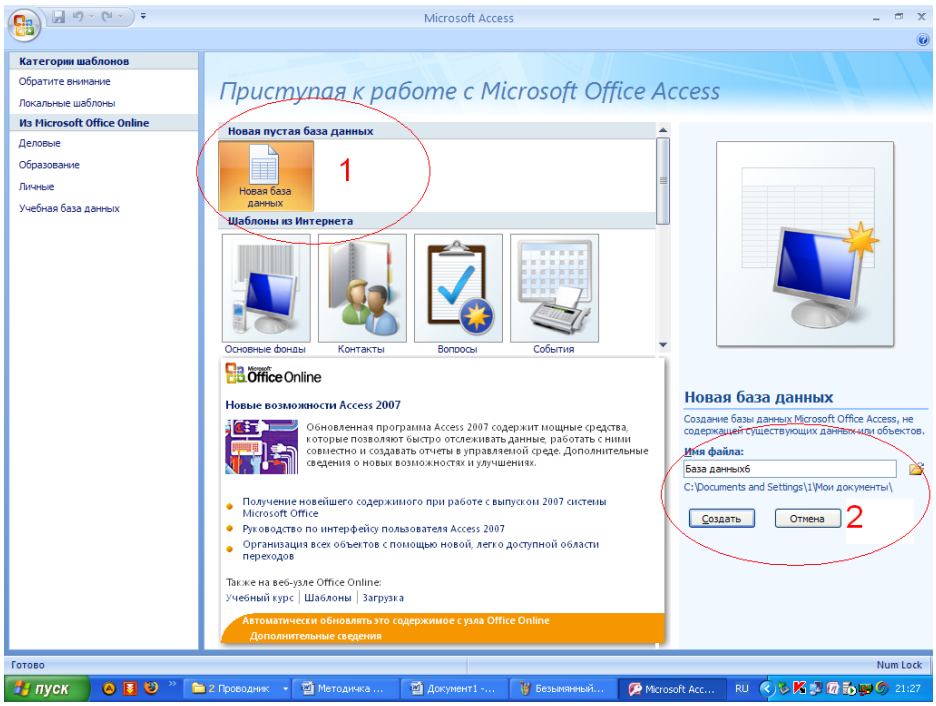



Рис.1. Создание базы данных

Под *базой данных* (БД) понимают хранилище структурированных данных, при этом данные должны быть непротиворечивы, минимально избыточны и целостны.

*Реляционные БД* представляют связанную между собой совокупность таблиц-сущностей базы данных (ТБД). Связь между таблицами может находить свое отражение в структуре данных, а может только подразумеваться, то есть присутствовать на неформализованном уровне. Каждая таблица БД представляется как совокупность строк и столбцов, где строки соответствуют экземпляру объекта, конкретному событию или явлению, а столбцы - атрибутам (признакам, характеристикам, параметрам) объекта, события, явления.

При практической разработке БД таблицы-сущности зовутся таблицами, строки-экземпляры - записями, столбцы-атрибуты - полями ТБД.

Одно из важнейших достоинств реляционных баз данных состоит в том, что можно хранить логически сгруппированные данные в разных таблицах и задавать *связи* между ними, объединяя их в единую базу. Такая организация данных позволяет уменьшить избыточность хранимых данных, упрощает их ввод и организацию запросов и отчетов.

## *Понятие первичного ключа*

В каждой таблице БД может существовать первичный ключ. Под первичным ключом понимают поле или набор полей, однозначно (уникально) идентифицирующих запись. Первичный ключ должен быть минимально достаточным: в нем не должно быть полей, удаление которых из первичного ключа не отразится на его уникальности.

Таблица 1. Данные отношения «Преподаватель»

| Таб. № | ФИО | Уч. степень | Уч. звание | Код кафедры |
|---|---|---|---|---|
| 101 | Андреев А.П. | Д-р техн. наук | Профессор | 01 |
| 102 | Апухтин И.С. | Канд. техн. наук | Доцент | 01 |
| 103 | Глухов И.Л. | Канд. техн. наук | Доцент | 01 |
| 104 | Сеченов Ю.Б. | Канд. техн. наук | Доцент | 01 |

В качестве первичного ключа в таблице «Преподаватель» может выступать только «Таб. №», значения других полей могут повторяться внутри данной таблицы.

Правила нормализации баз данных, и чисто практические соображения должны побудить разработчика **всегда определять первичный ключ для таблицы базы данных.**

## *Реляционные отношения (связи) между таблицами базы данных*

Между двумя или более таблицами базы данных могут существовать отношения подчиненности. Отношения подчиненности определяют, что для каждой записи главной таблицы *{master,* называемой еще *родительской}* может существовать одна или несколько записей в подчиненной таблице *{detail,* называемой еще *дочерней}.*

Существует три разновидности связей между таблицами базы данных:

- *«один-ко-многим»,*

- *«один-к-одному»,*
- *«многие-ко-многим».*

**Отношение «один-ко-многим»** имеет место, когда одной записи родительской таблицы может соответствовать несколько записей в дочерней таблице.
Связь "один-ко-многим" является самой распространенной для реляционных баз данных.
В широко распространенной нотации структуры баз данных IDEF1X отношение «один-ко-многим» изображается путем соединения таблиц линией, которая на стороне дочерней таблицы оканчивается кружком или иным символом. Поля, входящие в первичный ключ для данной ТБД, всегда расположены вверху и отчеркнуты от прочих полей линией.

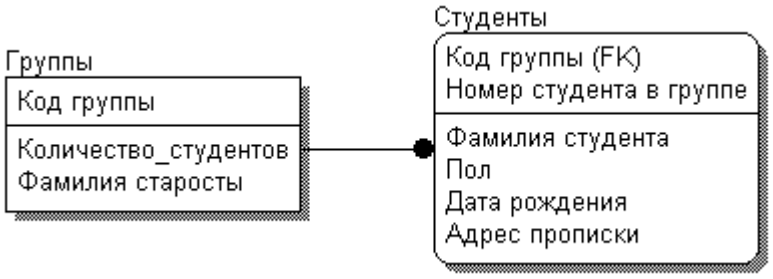



Рис.2. Отношение один-ко многим

**Отношение «один-к-одному»** имеет место, когда одной записи в родительской таблице соответствует одна запись в дочерней таблице.

| № сотрудника | ФИО | Должность | Отдел |
| --- | --- | --- | --- |
| 1 | Иванов П.П. | лаборант | 2 |
| 2 | Сидорова А.М. | бухгалтер | 1 |
| 3 | Петров А.Н. | инженер | 2 |

| № сотрудника | Год рождения | Число детей |
| --- | --- | --- |
| 1 | 1967 | 1 |
| 2 | 1955 | 1 |
| 3 | 1943 | 2 |

Рис.3. Отношение один-к одному

Данное отношение используют, если не хотят, чтобы таблица БД «не распухала» от второстепенной информации.

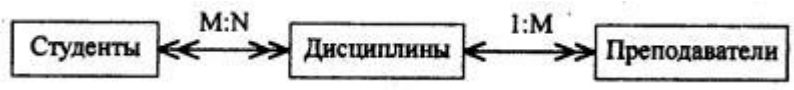
**Отношение «многие-ко-многим»** имеет место, когда:
а) записи в родительской таблице может соответствовать больше одной записи в дочерней таблице;
б) записи в дочерней таблице может соответствовать больше одной записи в родительской таблице.
Например, каждой студент изучает несколько дисциплин. Каждая дисциплина изучается несколькими студентами.

| Группа | № студента | ФИО студента |
| --- | --- | --- |
| ИДМ-107 | 1 | Иванов |
| ИДМ-107 | 2 | Петров |
| ИДМ-107 | 3 | Сидоров |

| Код дисциплины | Название дисциплины |
| --- | --- |
| 1 | Информатика |
| 2 | Математика |
| 3 | История |

Рис.4. Отношение многие-ко многим

Реляционные СУБД (в частности Access) не поддерживают связи «многие-ко-многим» на уровне индексов и ссылочной целостности. Считается, что всякую связь «многие-ко-многим» можно заменить на одну или более связей «один-ко-многим».
Например,



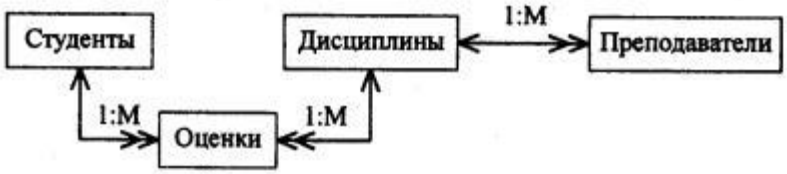



Рис.5. Трансформации связи многие-ко многим

## *Ссылочная целостность и каскадные воздействия*

Рассмотрим наиболее часто встречающуюся в базах данных связь «один-ко-многим». Как можно заметить, дочерняя и родительская таблицы связаны между собой по общему полю «Шифр группы». Назовем это поле **полем связи.**

**Таблица «Группа»**

| Шифр группы | Кол-во студ-в | Прох. балл |
| --- | --- | --- |
| 101 | 30 | 4,50 |
| 102 | 32 | 4,50 |
| 103 | 29 | 4,80 |

**Таблица «Студент»**

| Шифр группы | № студ. | ФИО | Год рожд. | Прох. балл |
| --- | --- | --- | --- | --- |
| 101 | 01 | Аристов Р.П. | 1979 | 4,25 |
| 101 | 02 | Борисова С.А. | 1979 | 4,25 |
| 102 | 01 | Макова Н.В. | 1978 | 4,75 |

Рис.6. Отношение один-ко многим

Возможны два вида изменений, которые приведут к утере связей между записями в родительской и дочерней таблицах:

- изменение значения поля связи в записи родительской таблицы без изменения значений полей связи в соответствующих записях дочерней таблицы;
- изменение значения поля связи в одной из записей дочерней таблицы без соответствующего изменения значения полей связи в родительской и дочерней таблицах.

**Таблица «Группа»**

| Шифр группы | Кол-во студ-в | Прох. балл |
| --- | --- | --- |
| А-101 | 30 | 4,50 |
| 102 | 32 | 4,50 |
| 103 | 29 | 4,80 |

**Таблица «Студент»**

| Шифр группы | № студ. | ФИО | Год рожд. | Прох. балл |
| --- | --- | --- | --- | --- |
| 101 | 01 | Аристов Р.П. | 1979 | 4,25 |
| 101 | 02 | Борисова С.А. | 1979 | 4,25 |
| 102 | 01 | Макова Н.В. | 1978 | 4,75 |

**Таблица «Группа»**

| Шифр группы | Кол-во студ-в | Прох. балл |
| --- | --- | --- |
| 101 | 30 | 4,50 |
| 102 | 32 | 4,50 |
| 103 | 29 | 4,80 |

**Таблица «Студент»**

| Шифр группы | № студ. | ФИО | Год рожд. | Прох. балл |
| --- | --- | --- | --- | --- |
| А-101 | 01 | Аристов Р.П. | 1979 | 4,25 |
| 101 | 02 | Борисова С.А. | 1979 | 4,25 |
| 102 | 01 | Макова Н.В. | 1978 | 4,75 |

Рис.7. Примеры нарушения целостности базы данных

И в первом, и втором случаях наблюдается *нарушение целостности базы данных***,** поскольку информация в ней становится недостоверной. Следовательно, нужно блокировать действия, которые нарушают целостность связей между таблицами, которую называют *ссылочной целостностью***.**

Чтобы предотвратить потерю ссылочной целостности, используется механизм *каскадных изменений.* Он состоит в обеспечении следующих требований:

- необходимо запретить изменение поля связи в записи дочерней таблицы без синхронного изменения полей связи в родительской таблице;
- при изменении поля связи в записи родительской таблице, следует синхронно изменить значения полей связи в соответствующих записях дочерней таблицы;
- при удалении записи в родительской таблице, следует удалить соответствующие записи в дочерней таблице.

Необходимость разрешения или запрещения каскадных изменений обычно реализуется в СУБД при определении связей между таблицами. Собственно, таким образом, и происходит создание ссылочной целостности.

### *Понятие внешнего ключа*

Для обеспечения ссылочной целостности в дочерней таблице создается внешний ключ. Во внешний ключ входят поля связи дочерней таблицы. Для связей типа "один-ко-многим" внешний ключ по составу полей должен совпадать с первичным ключом родительской таблицы.

### *Индексы и методы доступа*

Индексы представляют собой механизмы быстрого доступа к данным в таблицах БД.

Сущность индексов состоит в том, что они хранят значения индексных поле (т.е. полей, по которым построен индекс) и указатель на запись в таблице.

При ***последовательном методе доступа*** для выполнения запроса к таблице БД просматриваются *все* записи таблицы, от первой до последней.

При ***индексно-последовательном методе доступа*** для выполнения запроса к таблице БД указатель в индексе устанавливается на первую строку, удовлетворяющую условию запроса (или его части), и считывается запись из таблицы по хранящемуся на нее в индексе указателю.

Определение первичных и внешних ключей таблиц БД приводят к созданию индексов по полям, объявленным в составе первичных или внешних ключей.

### *Нормализация таблиц при проектировании БД*

При проектировании структуры новой БД определяют сущности (объекты, явления) предметной области, которые должны найти свое отражение в базе данных. В конечном итоге анализ предметной области должен привести к созданию *эскиза* БД. Сначала желательно изобразить *сущности* и *связи* между ними. Как правило, каждой сущности в БД соответствует *таблица*. Затем - в эскизе второго порядка - для каждой таблицы БД приводится список *атрибутов* - *полей* записи.

На этом этапе процесс проектирования структур БД является процессом творческим, неоднозначным, с другой стороны, узловые его моменты могут быть формализованы.

Одной из таких формализаций является требование, согласно которому реляционная база данных должна быть нормализована. Процесс *нормализации* имеет своей целью устранение избыточности данных и заключается в приведении к третьей нормальной форме (3НФ).

**Первая нормальная форма** (1НФ) требует, чтобы каждое поле таблицы БД:

- было неделимым;
- не содержало повторяющихся групп.

Неделимость поля означает, что значение поля не должно делиться на более мелкие значения. Например, если в поле "Подразделение" содержится название факультета и название кафедры, требование неделимости не соблюдается и необходимо из данного поля выделить или название факультета, или кафедры в отдельное поле.

Повторяющимися являются поля, содержащие одинаковые по смыслу значения. Например, если требуется получить статистику сдачи экзаменов по предметам, можно создать поля для хранения данных об оценке по каждому предмету. Однако в этом случае мы имеем дело с повторяющимися группами.

**Вторая нормальная форма (2НФ)** требует, чтобы все поля таблицы зависели от первичного ключа, то есть, чтобы первичный ключ однозначно определял запись и не был избыточен. Те поля, которые зависят только от части первичного ключа, должны быть выделены в составе отдельных таблиц.

**Третья нормальная форма** (3НФ) требует, чтобы значение любого поля таблицы, не входящего в первичный ключ, не зависело от значения другого поля, не входящего в первичный ключ.

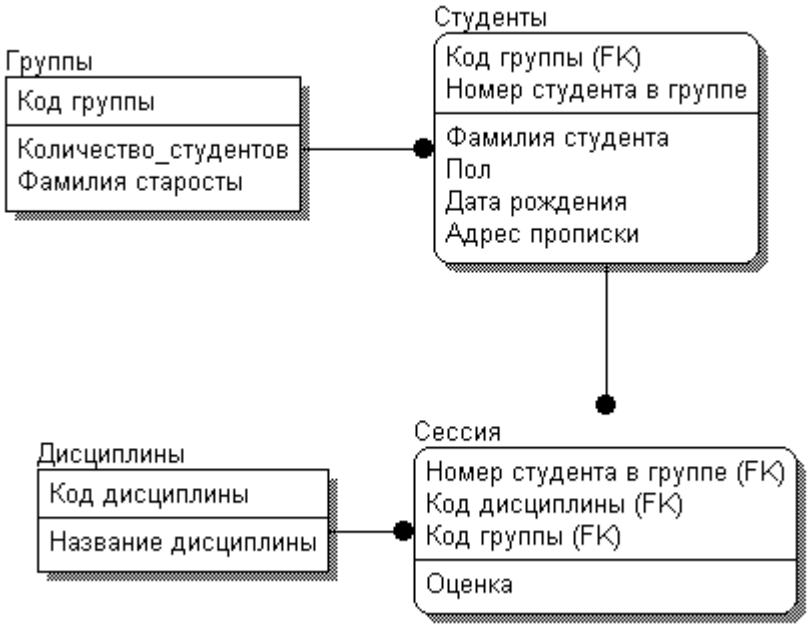



Рис.7. Пример логической модели базы данных "Сессия"

Основным средством хранения информации в СУБД Access являются плоские таблицы, состоящие из строк (записей) и именованных столбцов (полей). Каждое поле записи содержит одну характеристику объекта и имеет строго определенный тип данных (например, текстовая строка, число, дата).

**Порядок выполнения работы:**

## Варианты заданий:

Спроектировать базу данных согласно вариантам:

Вариант 1. Деятельность торговой фирмы.
В базе данных учесть следующие признаки: дату, количество, наименование, тип, цену проданного товара, покупателя, его фирму, город, телефон.

Вариант 2. Деятельность предприятия по сборке изделий.
В базе данных учесть следующие признаки: наименование, тип, цену продажи некоторого изделия, количество дней на его сборку, количество компонент в изделии, описание, изготовитель компонент, тип и стоимость каждого компонента.

Вариант 3. Деятельность стола заказов.
В базе данных учесть следующие признаки: дату получения и исполнения заказа, скидку на заказ, количество и цену товара, вошедшего в заказ, имя клиента, его расчетный счет и величину кредита.

Вариант 4. Оплата коммунальных услуг.
В базе данных учесть следующие признаки: фамилию квартиросъемщика, его адрес, жилую площадь, число проживающий в квартире, дату и период оплаты коммунальных услуг, стоимость одного квадратного метра жилплощади, стоимость потребления холодной воды на одного проживающего.

Вариант 5. Работа фирмы с поставщиками.

В базе данных учесть следующие признаки: дату продажи некоторого товара, количество, цену, скидку при продаже и налог на продажу, а также поставщиков товара, страну и наличие лицензии на продажу.

Вариант 6. Начисление зарплаты.
В базе данных учесть следующие признаки: фамилию, адрес, телефон сотрудника, дату его рождения и дату устройства на работу, дату, вид и количество в часах выполненной работы, описание выполненной работы, тип освобождения от
налога, нижнюю и верхнюю границы оплаты одного часа.

Вариант 7. Деятельность бюро добрых услуг.
В базе данных учесть следующие признаки: вид услуги, ее описание и стоимость, дату оказания этой услуги, скидку при оплате в зависимости от социального положения клиента, имя и место проживания клиента.

Вариант 8. Оплата междугородних телефонных разговоров.
В базе данных учесть следующие признаки: дату и время, продолжительность телефонного разговора, город, с которым состоялся разговор, фамилию, адрес, номер телефона клиента, тарифы городов и скидки на время разговора в течение
суток.

Вариант 9. Поваренная книга.
В базе данных учесть следующие признаки: названия и типы блюд, описание компонент блюда с указанием количества в граммах, калорийности и стоимости 1 грамма, количества жиров, углеводов и белков в 1 грамме компонента.

Вариант 10. Книжная палата.
В базе данных учесть следующие признаки: дату и количество проданных книг, название, автора, издательство, тематику, цену проданной книги, сведения об авторе: фамилию, пол, дату рождения.

Вариант 11. Музыкальная коллекция.
В базе данных учесть следующие признаки: дату, количество и стоимость проданного альбома, страну, авторов слов и музыки, исполнителя, длительность каждой композиции в альбоме.

Вариант 12. Видеотека.
В базе данных учесть следующие признаки: дату продажи видеокассеты, название фильма, страну, режиссера, тематику фильма, наличие Оскаров, дату выпуска фильма, стоимость кассеты, информацию о покупателе: возраст, пол,
социальное положение.

Вариант 13. Олимпийские игры.
В базе данных учесть следующие признаки: номер, символ олимпиады, город проведения, наличие в городе гор или моря. дату открытия и закрытия, число видов спорта, по которым проводятся соревнования, команды-участницы,
количество спортсменов в команде, число завоеванных золотых, серебряных и бронзовых медалей каждой командой.

Вариант 14. Учебный процесс.
В базе данных учесть следующие признаки: фамилии студентов, дату рождения, курс, дату сдачи, оценку и название предмета для каждого студента, для каждого предмета указать число часов на изучение, код предмета: гуманитарный
блок, математический или профессиональный и кафедру, которая ведет данный предмет.

Вариант 15. Учебная нагрузка преподавателя.
В базе данных учесть следующие признаки: фамилию, должность, звание преподавателя, кафедру, на которой он работает, название предмета, который он ведет, для предмета указать название, длительность в часах, код предмета:
гуманитарный блок, математический или профессиональный, для каждой должности и звания указать стоимость часа.

Вариант 16. Продажа билетов на самолеты.
В базе данных учесть следующие признаки: дату продажи билета, номер рейса, дату вылета рейса, номер места, фамилию пассажира, его социальное положение, данные по типу самолета, обслуживающего рейс: тип самолета, стоимость
билета, конечный пункт, продолжительность маршрута, экипаж, квалификация командира, его возраст и стаж полетов.

Вариант 17. Автобусный парк.
В базе данных учесть следующие признаки: дату подачи и дату исполнения заявки, продолжительность и вид поездки, число участников, сумма аванса, выплаченного за поездку, марка выделенного автобуса, его техническое состояние,
количество мест, стоимость билета, налоги и скидки в зависимости от вида поездки.

Вариант 18. Финансовое состояние вузов.
В базе данных учесть следующие признаки: число госбюджетных и хоздоговорных студентов в каждой студенческой группе, число преподавателей на факультете, среднюю стоимость обучения одного студента на факультете, среднюю зарплату преподавателя по университету, название университета, город, фамилию ректора, объем госбюджетных поступлений и дотаций из местного бюджета для каждого университета.

Вариант 19. Областное УВД.
В базе данных учесть следующие признаки: дату совершения и дату раскрытия преступления, вид и тяжесть преступления, описать участников: фамилию, дату рождения, вид участия, описать примененное оружие: марку, страну
изготовления, за кем числится.

Вариант 20. Фирма по продаже подержанных автомобилей.
В базе данных учесть следующие признаки: дату продажи, продавца, вид оплаты, данные о покупателе: фамилию, пол, возраст, социальное положение, информацию об автомобиле: марка, цвет, изготовитель, дата изготовления,
техническое состояние, мощность двигателя.

**Контрольные вопросы**

1. Дайте определение базы данных.
2. Чем поле отличается от записи?
3. Какие основные функции СУБД и что это такое?
4. Опишите возможности СУБД MS Access.

5. Какие объекты входят в состав файла базы данных MS Access?
6. Какие ограничения на имена полей, элементов управления и объектов действуют в MS Access?
7. Чем отличаются режимы работы с объектами базы данных в MS Access: оперативный режим, режим конструктора?
8. Опишите, какие типы данных могут иметь поля в MS Access. Каков их предельный размер?
9. Каково назначение справочной системы MS Access? Чем отличается поиск подсказки на вкладках: Содержание, Мастер ответов и Указатель?
10. Что такое выражения в MS Access? Какие бывают выражения и для чего они используются?
11. Какие особенности в записи различных операндов выражений: имя поля, число, текст?
12. Каково назначение построителя выражений?
13. С какой целью выполняется проектирование базы данных и в чем оно заключается?
14. Какие операции с данными в таблице базы данных вы знаете?
15. Каково назначение сортировки данных в таблице? Какие бывают виды сортировки?
16. Что подразумевает понятие абстрагирование в СУБД?

# Лабораторная работа №2

## Построение ER-диаграммы

**Цель работы**: Разработка ER-модели предметной области. Приобретение навыков моделирования предметной области, построения диаграмм «сущность-связь».
**Результат:** концептуальная ER-диаграмма, физическая ER-диаграмма, разработанные с помощью доступного редактора деловой графики.
**Теоретическая справка:**
**Пример**:
Построение базы данных предметной области «Расписание экзаменов».
Проектируемая система должна выполнять следующие действия:

- Хранить информацию о студентах, предметах, экзаменах (по какому предмету, какой группой и когда сдается), группах.
- Печатать расписание для каждой группы, списки групп, результаты экзаменов.

Разработаем ER-модель данной предметной области.
***Примечание:*** *исключительно ради рассмотрения различных вариантов связей и атрибутов примем за истину следующие условия:*
*1) дата рождения студента может быть не указана;*
*2) один и тот же экзамен сразу могут сдавать несколько групп.*

**Студент, предмет, группа, экзамен** - явные кандидаты на сущность.

Связи между сущностями:

- «каждый студент должен обучаться в одной группе», «группа может состоять из нескольких студентов»,
- «каждая группа может сдавать несколько экзаменов», «каждый экзамен может проводиться у одной или нескольких групп»,
- «каждый экзамен должен сдаваться по одному предмету», «по каждому предмету могут сдаваться несколько экзаменов».

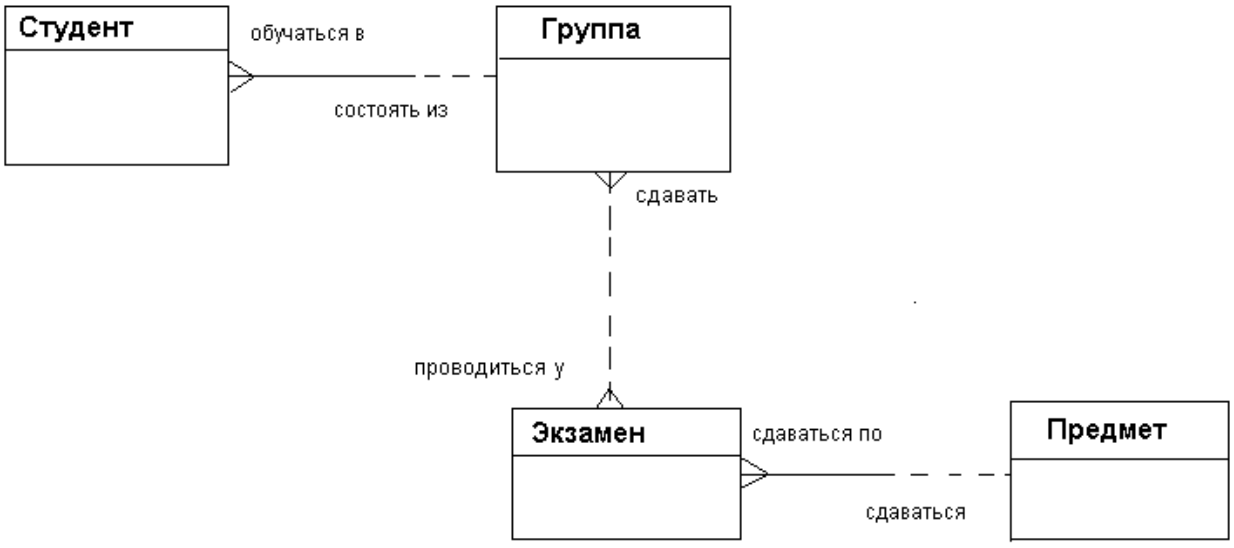



Рис.8. Пример логической модели базы данных

**Атрибуты сущностей:**

1. Атрибутами студента могут быть ***фамилия, имя, отчество, дата рождения.***
2. Атрибутами группы могут быть ***номер группы и форма обучения*** (дневная или заочная).
3. Атрибутом экзамена может быть ***дата*** его сдачи.
4. Атрибутом предмета может быть его ***название***.

*Однако **оценки** за экзамен не могут быть атрибутом никакой из имеющихся сущностей, а, по-видимому, представляют собой отдельную сущность.*
*Связи:*

- *«каждый студент может получить несколько оценок», «каждая оценка должна быть поставлена одному студенту»,*
- *«каждый экзамен может оценивать несколько оценок», «каждая оценка должна быть поставлена за один экзамен»*

5. Атрибутом оценки может быть ее ***значение***.

Отсюда

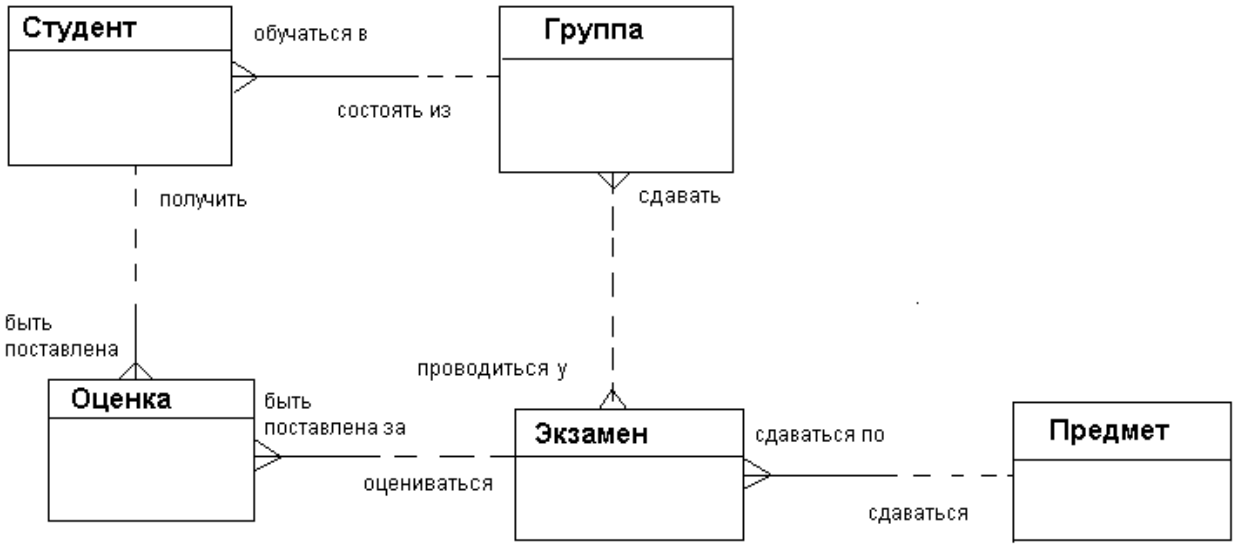



Рис.9. Пример логической модели базы данных

Сущности группа и экзамен связаны друг с другом отношением много-ко-многим. Такая связь должна быть разделена на две связи типа один ко многим. Для этого требуется дополнительная сущность. Введем сущность **«Строка расписания экзаменов»**.
Связи:

- «каждая строка расписания экзаменов должна включать одну группу», «каждая группа может упоминаться в одной или нескольких строках расписания»;
- «каждая строка расписания экзаменов должна включать один экзамен», «каждый экзамен может упоминаться в одной или нескольких строках расписания».

Отсюда

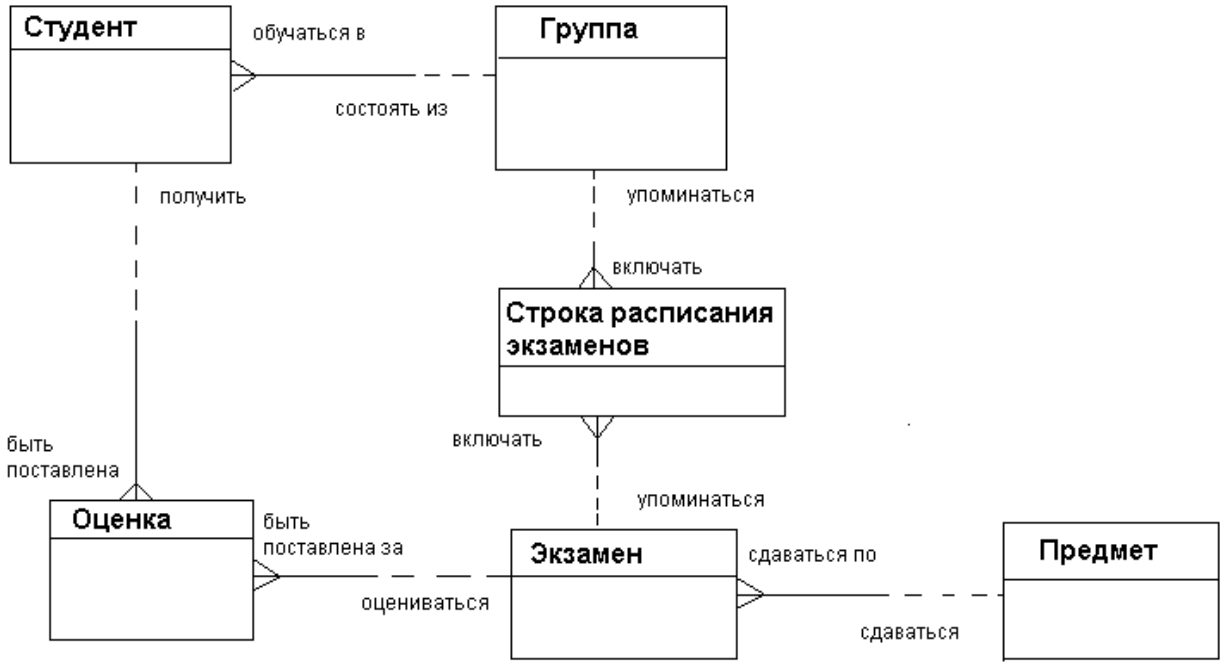



Рис.10. Пример логической модели базы данных

То есть мы получаем **концептуальную ER-диаграмму**.

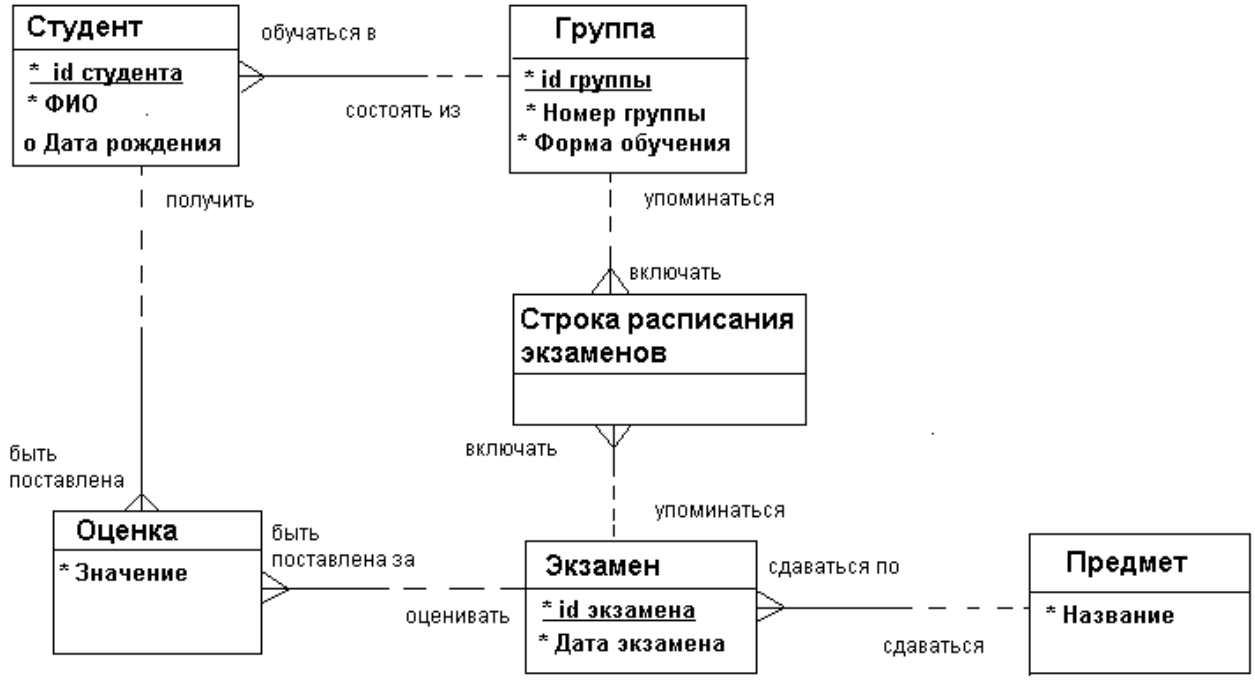



Рис.11. Пример концептуальной ER-диаграммы

**Физическая ER-диаграмма.**

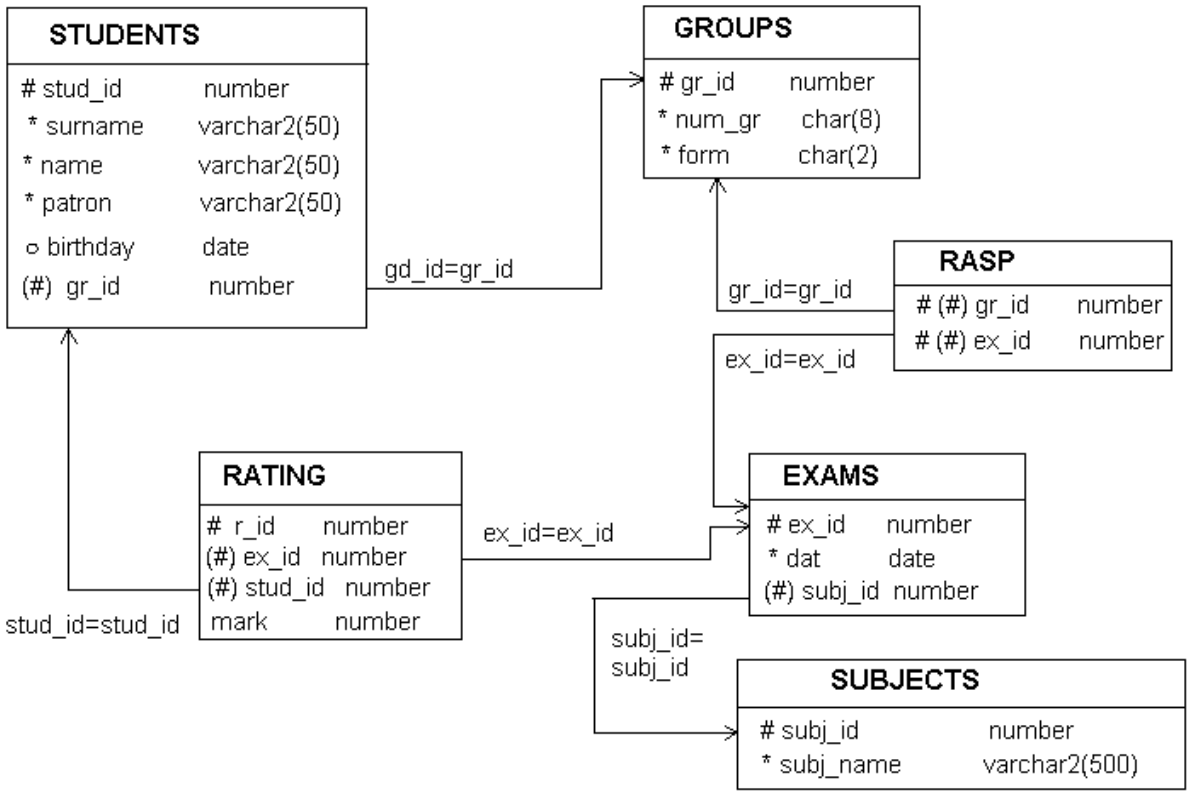



Рис.12. Пример физической модели базы данных

ER-диаграммы рекомендуется разрабатывать в среде онлайн-редактора деловой графики https://www.draw.io. Допускается также использование CASE-средства AllFusion ERwin Data Modeler, редактора деловой графики MS Visio.

**Варианты заданий:**

1. Институт (деканаты, кафедры, учебный отдел).

* Студенты: паспортные данные, адрес, дата зачисления, номер приказа, факультет, группа, является ли старостой, кафедра (специализация), изучаемые (изученные) предметы, оценки, задолженности, стипендия.
* Учебные курсы: название, факультет(ы), групп(ы), кафедра, семестр(ы), форма отчётности, число часов.
* Преподаватели: паспортные данные, адрес, телефон, фотография, кафедра, должность, учёная степень, начальник (зав. кафедрой), предмет(ы), число ставок, зарплата.

2. Библиотека института.

* Книги: авторы, название, раздел УДК, раздел (техническая, общественно-политическая и т.п.), место и год издания, издательство, количество страниц, иллюстрированность, цена, дата покупки, номер сопроводительного документа (чек, счёт/накладная), вид издания (книги, учебники, брошюры, периодические издания), инвентарный номер (есть только для книг и некоторых учебников), длительность использования читателями (год, две недели, день), электронная версия книги или ее реферата (отсканированный текст).
* Читатели: номер читательского билета, ФИО, год рождения, адрес, дата записи, вид (студент, аспирант, преподаватель, сотрудник), курс, номер группы, названия взятых книг и даты их выдачи.

3. Отдел кадров и бухгалтерия некоторой компании.

* Сотрудники: ФИО, паспортные данные, фотография, дом. и моб. телефоны, отдел, комната, раб. телефоны (в т.ч. местный), подчинённые сотрудники, должность, тип(ы) работы, задание(я), проект(ы), размер зарплаты, форма зарплаты (почасовая, фиксированная).
* Отделы: название, комната, телефон(ы), начальник, размер финансирования, число сотрудников.
* Проекты: название, дата начала, дата окончания, размер финансирования, тип финансирования (периодический, разовый), задачи и их исполнители, структура затрат и статьи расходов.

4. Отдел поставок некоторого предприятия.

* Поставщики: название компании, ФИО контактного лица, расчётный счёт в банке, телефон, факс, поставляемое оборудование (материалы), даты поставок (по договорам и реальные), метод и стоимость доставки.

* Сырьё: тип, марка, минимальный запас на складе, время задержки, цена, продукты, при производстве которых используется, потребляемые объемы (необходимый, реальный, на единицу продукции).

5. Технологический отдел некоторого предприятия.

* Производственные процессы: продукты, объёмы их производства, необходимые материалы, количества разных видов материалов на единицу продукции, отходы производства; используемое оборудование и его тип, даты ввода оборудования в строй, сроки амортизации, производительность оборудования; человеческие ресурсы (сколько всего и сколько по производству единицы продукции — сколько необходимо и сколько реально).
* Материалы: тип (категория), марка, является ли сырьем (или производится на предприятии), потребляемые объемы (в т.ч. на единицу конечной продукции), в рамках каких технологических процессов используется, цена.

6. Отдел продаж некоторой фирмы.

* Клиенты: название компании, ФИО контактного лица, адрес выставления счёта, адрес доставки, телефон, факс.
* Заказы: тип заказа (покупка, гарантийный ремонт, негарантийный ремонт), общая стоимость, скидка, товар(ы), их изготовители, модели (марки), серийные номера, описание неисправностей, необходимые ресурсы, клиент, дата получения заказа, срок завершения, дата выставления счёта и его оплаты, метод оплаты, дата поставки, метод и стоимость доставки.
* Ресурсы: ФИО, отдел(ы) и телефон(ы) исполнителя(ей), число рабочих часов для выполнения заказа, ставка зарплаты, ответственный за выполнение заказа, необходимое оборудование и расходные материалы, их количество и стоимость, а также наличие материалов на складе.

7. Магазин (внутренний учет).

* Клиенты: юридическое или физическое лицо, ФИО, адрес, телефон, адрес выставления счёта, вид и номер карточки, факс.
* Продажи: наименования, модели (марки) и серийные номера товаров, поставка из магазина или со склада, количество и общая стоимость товаров, размер скидки, тип скидки, форма оплаты (наличными, оплата счёта, по карточке), необходимость доставки, стоимость и тип доставки, адрес доставки.
* Товары: категория, модель, название производителя, адрес производителя, цена, количество в магазине и на складе.

8. Электронный магазин (информация для клиентов).

* Товары: категория, модель, производитель, цены (в т.ч. средняя и минимальная), есть ли в наличии, описание, характеристики, внешний вид; магазины, где можно купить товар, их телефоны и адреса; аксессуары, их цены и где их купить.
* Магазины: название, компания-владелец, её юридический адрес и home-site, контактные телефоны, адрес, схема проезда, эмблема; товары и цены на них; рекламная информация: некоторые товары с фотографиями, описаниями и ценами, основные отделы (категории товаров).

9. Пункт проката видеозаписей (внутренний учет).

* Видеокассеты: идентификационный номер видеокассеты, тип видеокассет, дата его создания, компания-поставщик, число штук данного типа (общее, в магазине, выдано в настоящее время, выдано всего, выдано в среднем за месяц), общая длительность записей; записи видеокассет: название, длительность, категория, год выпуска и производитель (оригинала).
* Клиенты: ФИО, паспортные данные, адрес, телефон; заказы, т.е. взятые видеокассеты (сейчас и в прошлом): номер, дата выдачи, дата возвращения, общая стоимость заказа.

10. Пункт проката видеозаписей (информация для клиентов).

* Видеокассеты: краткое описание, внешний вид (этикетка), марка (пустой) видеокассеты, цена за единицу прокатного времени (например: 1 день, 3 дня, неделя), есть ли в наличии, общая длительность записей; записи на видеокассете: название, длительность, жанр (категория), тема, год и страна выпуска (оригинала), кинокомпания, описание, актеры, режиссер.
* Заказы: идентификационные номера и названия выданных видеокассет, дата выдачи, дата возвращения (продления), общая стоимость заказа, возвращены ли кассеты заказа.

11. Кинотеатры (информация для зрителей).

* Фильмы: название, описание, жанр (категория), длительность, популярность (рейтинг, число проданных билетов в России и в мире), показывается ли сейчас (сегодня, на текущей неделе), в каких кинотеатрах показывается, цены на билеты (в т.ч. средние).
* Кинотеатры: название, адрес, схема проезда, описание, число мест (в разных залах, если их несколько), акустическая система, широкоэкранность, фильмы и цены на них: детские и взрослые билеты в зависимости от сеанса (дневной, вечерний и т.п.) и от категории мест (передние, задние и т.п.); сеансы показа фильмов (дата и время начала).

12. Ресторан (информация для посетителей).

* Меню: дневное или вечернее, список блюд по категориям.
* Блюда: цена, название, вид кухни, категории (первое, второе и т.п.; мясное, рыбное, салат и т.п.), является ли вегетарианским, компоненты блюда, время приготовления, есть ли в наличии.
* Компоненты блюд: тип (гарнир, соус, мясо и т.п.), калорийность, цена, рецепт, время приготовления, есть ли в наличии, ингредиенты (продукты) и их расходы на порцию.

13. Аналитический отдел некоторой компании (поиск и анализ публикаций).
* Категории: название, тип (область исследований, область приложений и т.п.), родительская категория, дочерние категории, связанные по смыслу категории (с пояснениями о связях), найденные публикации.
* Публикации: название, тип (газетная, книжная, web и т.п.), название, тип, адрес и телефон источника (газета, книга, сайт и т.п.), выходные данные (date-line), язык, реферат, ключевые слова, категории (с указанием степени уверенности отнесения к ним), текст и его тип (обычный, DOC, HTML, отсканированные картинки и т.п.), обзор.
* Задачи: тип задачи (классификация или поиск), сотрудник (создавший категорию или нашедший публикацию, ответственный за категорию или публикацию и т.п.), завершена ли работа над задачей.

14. Аналитический отдел некоторой компании (анализ рынка технологий — например, по публикациям — см. №13).
* Организация: название, тип (промышленная, финансовая, торговая, исследовательская и т.п.), категория(и), организация-владелец (акционеры), страна, контактная информация; договорные отношения с другими организациями.
* Технология (продукт): название, категория(и), организация-разработчик и производитель(и), использующие организации.
* Человек: фамилия, имя, тип (начальник, менеджер, создатель технологии и т.п.), организация(и), в которой работает, контактная информация.

15. Компания по (разработке и) сопровождению программного обеспечения.
* Ошибка (bug): краткое и полное описание, срок поступления информации об ошибке, её источник (пользователь, тестировщик) и его координаты, уровень ошибки (критическая, важная, незначительная и т.п.), категория функциональности (интерфейс, данные, расчетный алгоритм, другое, неизвестная категория), часть проекта, модуль (пакет), программист, ответственный за модуль, программист, ответственный за исправление ошибки, срок исправления (необходимый и реальный), исправлена ли, проверено ли исправление тестировщиком.

**Контрольные вопросы**

1. В чем отличие ERD от DFD?
2. Какие основные цели преследуют ER-диаграммы?
3. Какие основные элементы используются в ER-диаграммах?
4. Какие способы изображения элементов на ER-диаграммах вы знаете?
5. Как на ER-диаграммах указываются ключевые атрибуты. Приведите пример ER-диаграммы с зависимостью один ко-многим, где одна из сущностей обязательна, а вторая нет.
6. Какие типы связей вы знаете?
7. Какими свойствами характеризуются связи?

# Лабораторная работа №3

## *Построение базы данных в СУБД Access. Нормализация отношений*

**Цель работы:** изучить и закрепить на практике методы и средства СУБД по корректному заполнению и модификации таблиц БД и методы контроля вводимых данных путем связывания таблиц.
**Результат:** mdb-файл (accdb-файл), содержащий базу данных в формате MS Access, построенную в соответствии с вариантом задания.
**Краткая теория:**
Для создания новой таблицы необходимо открыть базу данных, перейти на вкладку **Создать** и выбрать желаемый пункт для создания таблицы.

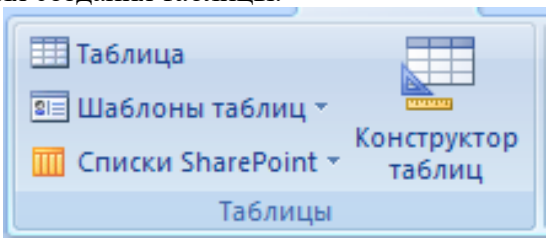



Рис.13. Окно выбора способа создания новой таблицы в СУБД MS Access

В Access используются три способа создания таблиц: путем ввода данных (by entering data), с помощью Мастера создания таблиц (by using wizard) и с помощью Конструктора таблиц (in Design view), который является наиболее универсальным.

В режиме Конструктора таблицы создаются путем задания имен полей, их типов и свойств. Чтобы создать таблицу в режиме Конструктора, необходимо:

1. Щелкнуть левой кнопкой мыши на Вкладке **Создание Конструктор таблицы**.

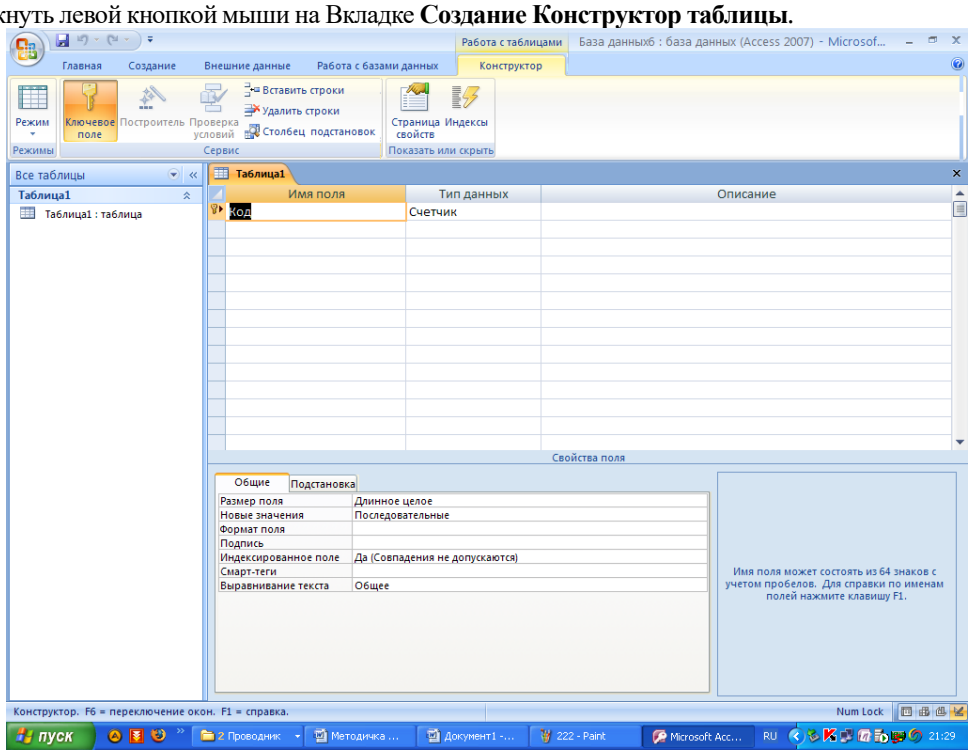



Рис.14. Окно создания новой таблицы в режиме Конструктора

2. В окне Конструктора таблиц в столбец **Имя поля** (Field Name) ввести имена полей создаваемой таблицы.
3. В столбце **Тип данных** (Data Type) для каждого поля таблицы выбрать из раскрывающегося списка тип данных, которые будут содержаться в этом поле.
4. В столбце **Описание** (Description) можно ввести описание данного поля (не обязательно).
5. В нижней части окна Конструктора таблиц на вкладках **Общие** (General) и **Подстановка** (Lookup) ввести свойства каждого поля или оставить значения свойств, установленные по умолчанию.
6. После описания всех полей будущей таблицы нажать кнопку **Закрыть** (в верхнем правом углу окна таблицы).

7. На вопрос **Сохранить изменения макета или структуры таблицы <имя таблицы>?** (Do you want to save changes to the design of table <имя таблицы>?), нажать кнопку Да (Yes).
8. В окне **Сохранить как** (Save As) в поле Имя таблицы (Table Name) ввести имя создаваемой таблицы и нажать кнопку ОК.
9. В ответ на сообщение Ключевые поля не заданы (There is no primary key defined) и вопрос Создать ключевое поле сейчас? (Do you want to create a primary key now?) нажмите кнопку Да (Yes) если ключевое поле необходимо, или кнопку Нет (No) если такого не требуется.
10. После указанных действий в списке таблиц в окне базы данных появятся имя и значок новой таблицы. Ввести данные в созданную таблицу можно, открыв таблицу в режиме Таблицы.

В Microsoft Access имеются следующие типы данных:

- Текстовый (Text) — символьные или числовые данные, не требующие вычислений. Размер текстового поля задается с помощью свойства Размер поля (FieldSize), в котором указывается максимальное количество символов, которые могут быть введены в данное поле.
- Поле MEMO (MEMO) — поле MEMO предназначено для ввода текстовой информации, по объему превышающей 255 символов. Этот тип данных отличается от типа Текстовый (Text) тем, что в таблице хранятся не сами данные, а ссылки на блоки данных, хранящиеся отдельно. За счет этого ускоряется обработка таблиц (сортировка, поиск и т. п.).
- Числовой (Number) — числовой тип применяется для хранения числовых данных, используемых в математических расчетах.
- Дата/Время (Date/Time) — тип для представления даты и времени.
- Денежный (Currency) — тип данных, предназначенный для хранения данных, точность представления которых колеблется от 1 до 4 десятичных знаков.
- Счетчик (AutoNumber) — поле содержит 4-байтный уникальный номер, определяемый Microsoft Access автоматически для каждой новой записи либо случайным образом, либо путем увеличения предыдущего значения на 1. Значения полей типа счетчика обновлять нельзя.
- Логический (Yes/No) — логическое поле, которое может содержать только два значения, интерпретируемых как Да/Нет, Истина/Ложь, Включено/Выключено.
- Поле объекта OLE (OLE object) — содержит ссылку на OLE-объект (лист Microsoft Excel, документ Microsoft Word, звук, рисунок и т. п.). В поле объекта OLE могут храниться произвольные данные, в том числе и данные нескольких типов. Это позволяет обойти основное ограничение реляционных баз данных, которое требует, чтобы в каждом поле хранились данные только одного типа.
- Гиперссылка (Hyperlink) — дает возможность хранить в поле ссылку, с помощью которой можно ссылаться на произвольный фрагмент данных внутри файла или Web-страницы на том же компьютере, в Интернет.

При установке курсора на какое-либо из заполняемых полей таблицы в левой нижней части экрана появляются свойства этого поля. Причем свойства задаваемого поля зависят от принятого для него типа данных.

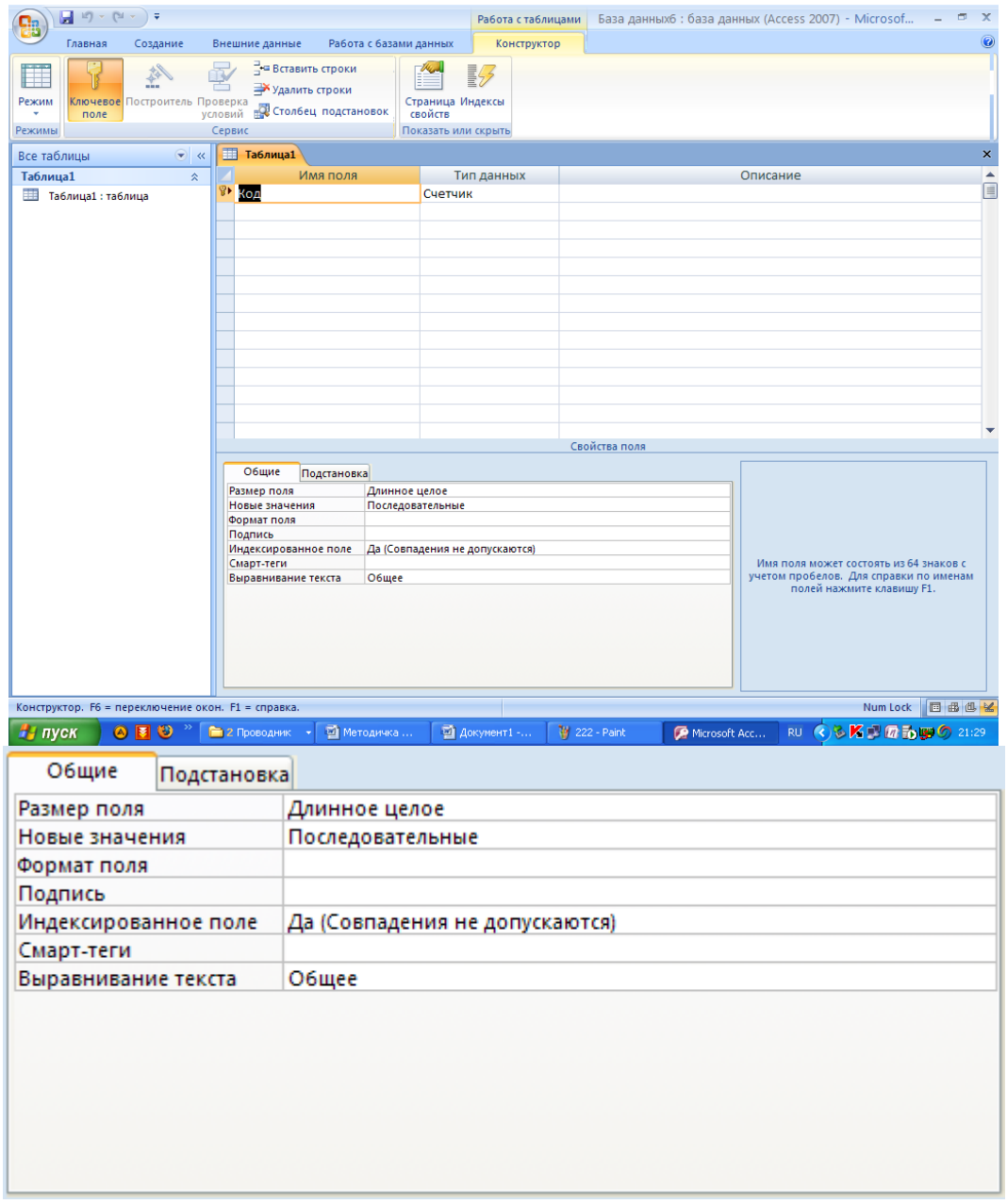



Рис.15. Установка свойств полей таблицы

Свойство ***Размер поля*** зависит от размера хранящейся в нем информации. Например, для ввода данных в поле *Фамилия работника* достаточно 20 символов.

Для большинства типов данных характерно свойство поля - ***Подпись*** (Caption). С помощью этого свойства можно задать названия полей таблицы, которые выводятся в различных режимах (в надписях, присоединенных к элементам управления формы, в заголовке столбца в режиме Таблицы и т.д.). Кроме того, для большинства типов данных существует свойство ***Обязательное поле*** (Required), которое определяет необходимость ввода данных в это поле.

Свойство ***Формат поля*** (Format) указывает формат отображения данных из поля в режиме Таблицы. Для определения формата полей текстового типа используются специальные символы форматирования. Для числовых полей значение формата можно выбрать из раскрывающегося списка. Для логических полей можно выбрать из списка следующие варианты: Да/Нет (Yes/No), Истина/Ложь (True/False), Вкл/Выкл (On/Off).

С помощью свойства ***Маска ввода*** (Input Mask) указывается маска, позволяющая автоматизировать проверку ввода символов в поле. Она применяется к таким полям, как номер телефона, дата и т. д. Задавать маску ввода можно вручную или с помощью Мастера.

Проще всего научиться работать с маской ввода с помощью мастера по созданию масок ввода. Для создания маски с помощью мастера необходимо щелкнуть на соответствующее поле, затем - по ячейке свойства **Маска ввода,** расположенной в нижней части этого окна. Вы увидите справа небольшую кнопку с тремя точками - кнопку построителя.

Нажмите эту кнопку, чтобы воспользоваться помощью мастера по созданию масок ввода. Выберите один из предлагаемых стандартных масок и нажмите далее.

Свойство ***Индексированное поле*** (Indexed) определяет, является ли данное поле индексированным, и если является, то в каком режиме. Существуют два режима индексирования: *Совпадения допускаются* (Duplicates OK) и *Совпадения не допускаются* (No duplicates). В первом случае поле может содержать повторяющиеся значения, во втором — нет.

Два свойства, которые тоже определены для большинства полей, позволяют выполнять проверку данных, вводимых в поле:

- ***Условие на значение*** (Validation Rule) — свойство определяет условие (ограничение), накладываемое на вводимые в это поле данные. При несоответствии вводимых данных указанному условию выдается сообщение об ошибке. Для создания условия на значение с помощью мастера необходимо щелкнуть на соответствующее поле, затем - по ячейке свойства **Условие на значение,** расположенной в нижней части этого окна. Вы увидите справа небольшую кнопку с тремя точками - кнопку построителя. Нажмите эту кнопку, на экране появится построитель выражения, с помощью которого вы сможете задать любое условие.
- ***Сообщение об ошибке (Validation Text)*** — свойство определяет то сообщение, которое будет выдаваться пользователю, если при вводе данных не соблюдается условие, указанное в свойстве Условие на значение (Validation Rule). Например, "Введенное значение выходит за рамки заданного диапазона".

Ввод различных выражений в Microsoft Access возможен не только вручную, но и может быть облегчен с помощью **Построителя выражений**.

Построитель выражений вызывается всякий раз, когда в поле свойства объекта Access вы щелкните кнопку ***Построить***  на строке главного меню или нажимаете кнопку Построителя (кнопка с тремя точками, например, в строке свойства ***Условие на значение***).

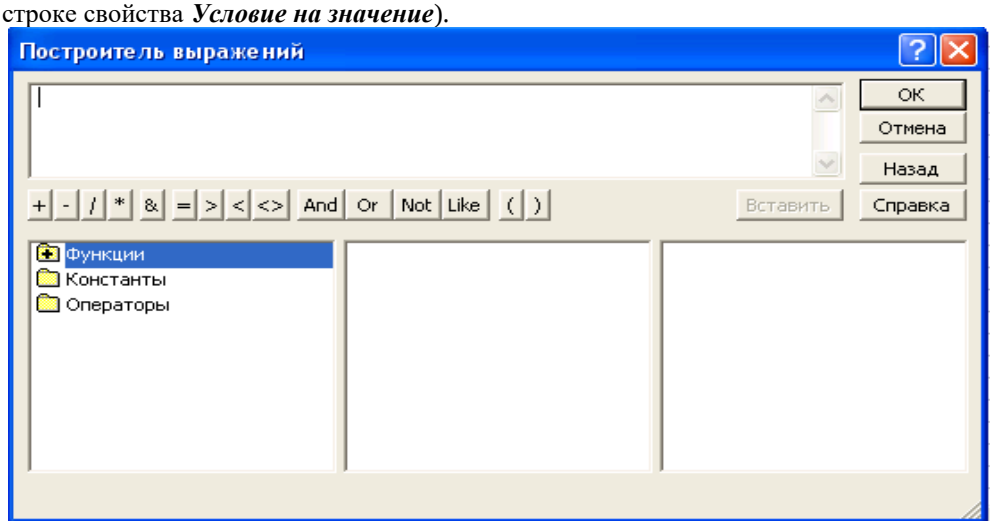



Рис.16. Окно Построителя выражений

С помощью удобных клавиш, расположенных в Построителе можно задавать любое выражение, в том числе использовать различные формулы для вычислений и применять встроенные функции Access. Введение выражений с помощью Построителя позволяет также наглядно определить правильность ввода и тем самым избежать неточностей и ошибок при наборе текста и формул.

Создать поле-список и сделать более простым ввод значений в поле таблицы позволяет ***операция подстановки***. Используя эту операцию, при последующем заполнении таблицы данными можно выбирать значения поля из списка. Список значений может быть как фиксированным, так и содержаться в таблице или запросе. Сформировать столбец подстановок для поля помогает **Мастер подстановок** (Lookup Wizard), который находится в поле ***Тип данных***. Проверить наличие подстановки можно, открыв вкладку **Подстановка,** в свойствах поля.

Выше неоднократно упоминалось понятие ключевого поля. ***Ключевое поле*** — это одно или несколько полей, комбинация значений которых однозначно определяет каждую запись в таблице. Если для таблицы определены ключевые поля, то Microsoft Access предотвращает дублирование или ввод пустых значений в ключевое поле. Ключевые поля используются для быстрого поиска и связи данных из разных таблиц при помощи запросов, форм и отчетов.

Операция сортировки данных используется всегда для удобства нахождения нужной информации. Когда на экране (или на бумаге) отображается таблица, гораздо легче найти нужную строку, если эти строки упорядочены. Вы привыкли к тому, что табличные данные упорядочены по алфавиту, по дате, по увеличению или уменьшению значений в столбцах, содержащих числа. Но в разных ситуациях мы хотели бы сортировать строки по разным признакам (столбцам таблицы). В идеале это должно выполняться легким движением руки. Именно так и позволяет делать Access. По умолчанию, когда таблица открывается

в режиме Таблицы, она упорядочивается по значению ключевого поля. Если ключевое поле для таблицы не определено, записи выводятся в порядке их ввода в таблицу. Если нужно отсортировать записи по значению другого поля, достаточно установить курсор на любую строку соответствующего столбца и нажать одну из кнопок на панели инструментов: **Сортировка по возрастанию** или **Сортировка по убыванию**.

**Создание Связей и индексов*:***

Каждая БД представляет обычно несколько таблиц, число которых может доходить, в общем случае, до десятков и сотен. При этом часто оказывается, что в разных таблицах хранятся одинаковые данные.

Для связывания полей необязательно совпадение их имен, но обязательно совпадение их типов.

Связи между таблицами можно устанавливать двумя путями. Первый путь - *графический.* Войдите в Схему данных, и выберите таблицы для установления связей. Далее следует выбрать в главной таблице поле для связи, нажать левую кнопку мыши и перетащить поле во вторую таблицу. Отпустить левую кнопку мыши над тем полем подчиненной таблицы, с которым устанавливается связь.

Второй путь - создание связей через **Мастер подстановок**, который активизируется при выборе типа данных в конструкторе таблицы.

Чтобы изменить связь нужно войти в Схему данных (ярлык на главной панели инструментов), щелкнуть правой кнопкой мыши по линии связи и войти в диалоговое окно *Изменить связь*, установив обеспечение целостности данных, каскадное обновление и каскадное удаление записей связанных таблиц.

Если отношение между таблицами «один-ко-многим», то слева из списка *Таблица/запрос* выбирается главная таблица и поле в этой таблице, а справа из списка *Связанная таблица/запрос* - подчиненная и соответственно поле в ней. Если отношение «один-к-одному», то порядок таблиц значения не имеет. Если вы уже устанавливали связь графически, то все поля в списке уже выбраны, и нужно только определить правила ссылочной целостности. Для этого устанавливают флажок *Обеспечение целостности данных.*

Когда создается новая связь, можно также воспользоваться кнопкой *Новое* и в окне *Создание* ввести имена связываемых таблиц и имена полей, используемых для связи. Нажать кнопку *ОК*.

**Индексы** - это внутренняя таблица, состоящая из двух столбцов: значения выражения, содержащего все поля, включенные в индекс, и местоположение каждой записи таблицы с данным значением индексного выражения. Допустим, вы часто осуществляете поиск Сотрудников в таблице Сотрудники по Фамилии, Имени и Отчеству. Если индекс отсутствует, то Access просматривает все записи вашей таблицы. Эта операция выполняется быстро только при небольшом количестве записей. Если вы создадите индекс, то Access его использует для прямого поиска записей.

Индексы бывают *по одному полю* и *составные.* Для создания индекса по одному полю откройте таблицу в режиме Конструктора и откройте поле, для которого вы хотите создать индекс. Щелкните по ячейке Свойства Индексированное поле, расположенное в нижней части окна таблицы и установите значение Да. Если поле, по которому вы индексируете имеет повторяющиеся значения, то следует выбрать Да (совпадения допускаются). Однако Access также может создать индекс, содержащий только уникальные значения данного поля - для этого нужно выбрать в списке значений Да (совпадения не допускаются). Access автоматически создает такой индекс по первичному ключу таблицы.

Чтобы создать индекс по нескольким полям, откройте таблицу Сотрудники в режиме Конструктора, затем нажмите кнопку Индексы на панели инструментов в режиме Конструктора.

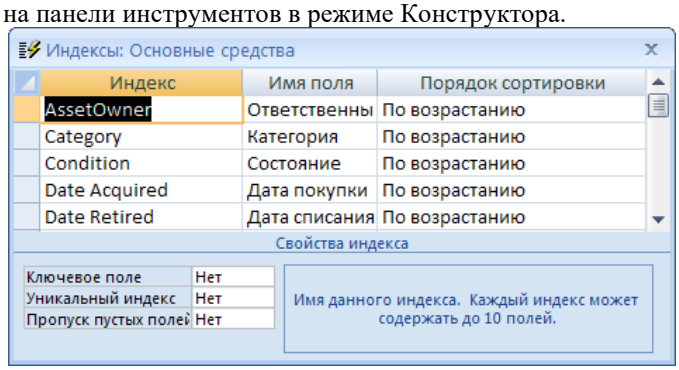



Рис. 17. Диалоговое окно создания Индекса

В столбце *Индекс* заполняется название вашего индекса, в столбце *Имя поля* указывается наименование индексируемого поля, в столбце *Порядок сортировки* можно указать, как вы хотите отсортировать данные, по возрастанию или по убыванию.

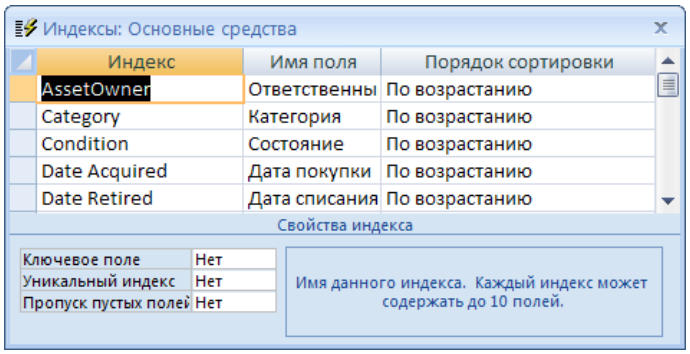



Рис. 18. Создание нескольких индексов

Для построения составного индекса поместите курсор на пустую строку и введите с клавиатуры имя индекса. В столбце *Имя поля* окна индексы выберите необходимое поле, пользуясь раскрывающимся списком, затем, не указывая никакого нового названия индекса, поместите курсор на следующую строку столбца *Имя поля* и укажите следующее поле, которое будет включено в составной индекс.

**Порядок выполнения работы:**

1. Разработать структуру базы данных согласно вариантам задания 1.
2. Создать таблицы в среде Microsoft Access. Для каждого элемента данных определить имя, тип данных, свойства: размер, условия на значение, маску ввода, сообщение об ошибке, значение по умолчанию и т.д. Определить для таблиц
первичные ключи.
3. Организовать связь между таблицами.
4. Заполнить таблицы данными.

**Контрольные вопросы**

17. Какие структуры ACCESS вы знаете?
18. Для чего служит структура "таблица"?
19. Для чего данным задаются различные типы?
20. Что можно настроить в свойствах поля таблицы?
21. Как отсортировать данные в таблицах по возрастанию и убыванию?
22. Для чего служат связи?
23. Виды связей. Способы создания связей.
24. Назначение индексов.
25. Виды индексов.
26. Как создать составной индекс?

# Лабораторная работа №4

## *Создание запросов*

**Цель работы:** изучение и закрепление на практике методов формирования и использования запросов для выборки данных в таблицах.
**Результат:** mdb-файл (accdb-файл), содержащий базу данных в формате MS Access, запросы, выполненные в соответствии с вариантом задания.

**Краткая теория:**

Таблицы только хранят данные, но необходимо иметь возможность выбрать заданные данные из нескольких таблиц. Именно для этого служат **запросы на выборку**.

В **запросах на выборку** данные могут: отбираться по многим критериям; сортироваться; с ними могут производиться вычислительные операции.

Запрос это временная таблица. Это значит, что данные в них не хранятся постоянно, а только временно вызываются из таблиц, по заранее заданному шаблону, в момент активизации запроса. Таким образом, в базе

данных постоянно хранится только шаблоны вызова данных (временные таблицы удаляются после закрытия запроса), а сама информация не дублируется.

Запросы позволяют:

1. формировать сложные критерии для выбора записей из одной или нескольких таблиц;
2. указывать поля, которые должны быть отображены для выбранных записей;
3. редактировать группы записей, удовлетворяющих определенным критериям;
4. выполнять вычисления с использованием выбранных данных.

**Для создания запроса необходимо:**

1. Открыть свою базу данных.
2. Перейти на вкладку Запросы
3. Во вкладке Создание выбрать команду Конструктор запросов**.** После нажатия этой опции появится окно конструктора запроса с диалоговым окном добавления таблиц. Окно добавления таблиц можно также вызвать командой Добавить таблицу из меню Запрос.

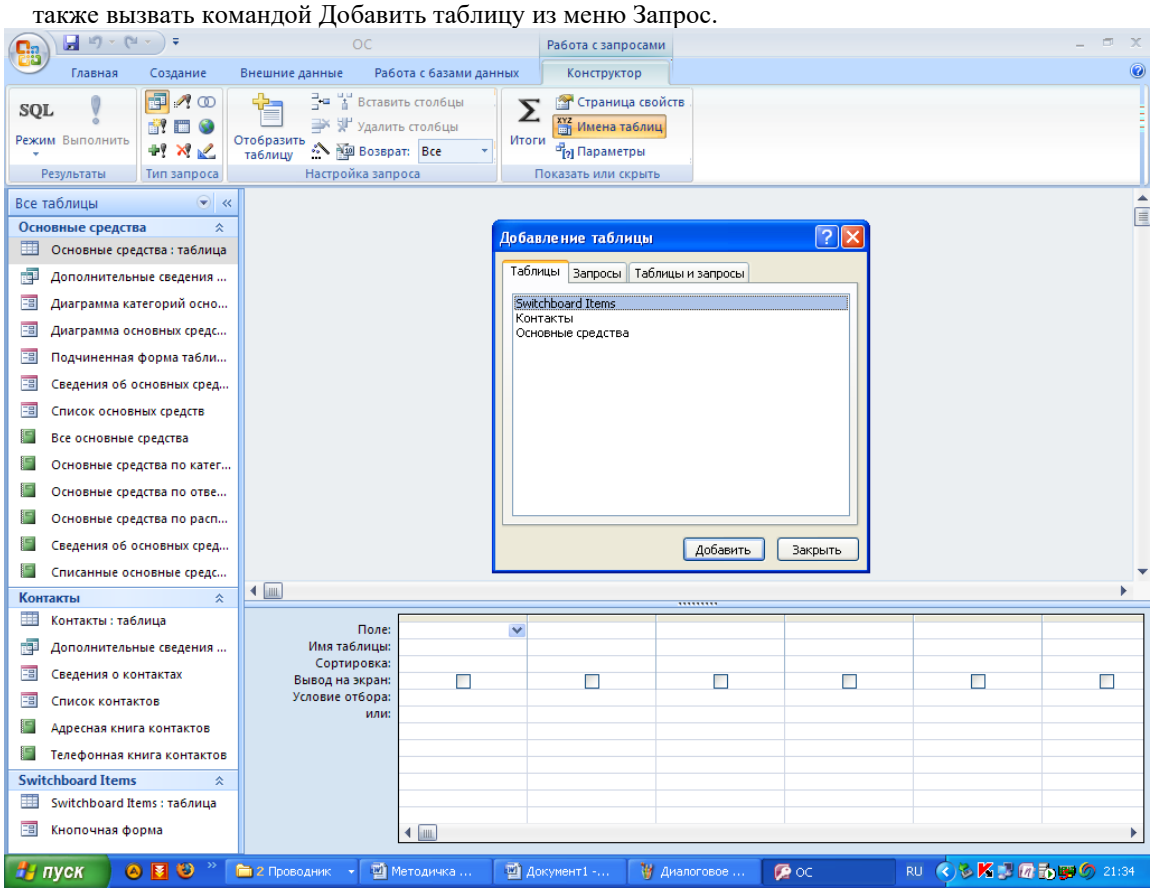



Рис. 19. Окно создания запроса в режиме Конструктора

4. Добавить в запрос необходимые таблицы
5. Убедиться, что между добавленными таблицами установлены связи.
6. После добавления таблиц нажать кнопку **Закрыть** в окне **Добавление таблицы.**
7. Затем нужно указать, какие поля из базовых таблиц будут отображаться в запросе. Для этого, выделить нужное поле в таблице-источнике, подвести указатель мыши к выделенному полю, нажать на левую кнопку мыши и перетащить поле в нужное место бланка запроса.

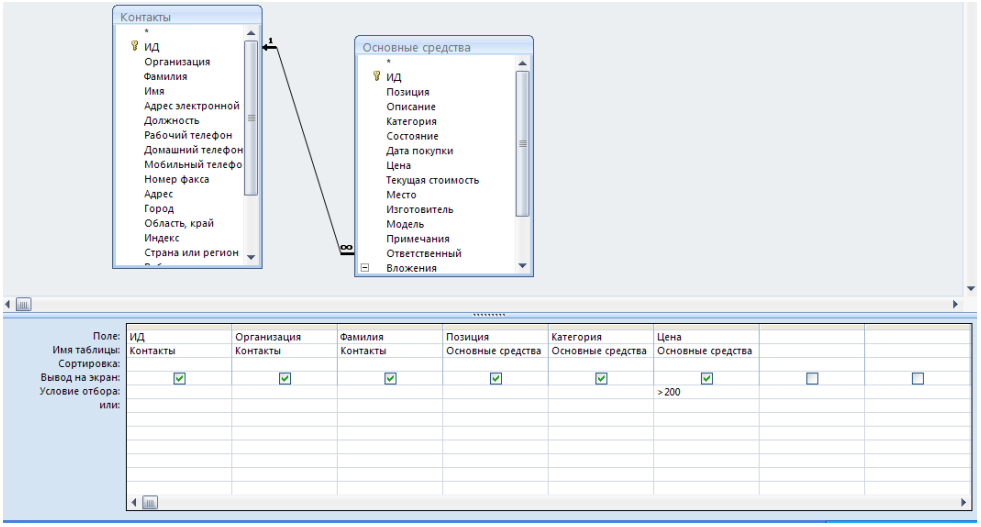



Рис. 20. Окно формирования запроса на выборку

8. В строке Условие отбора указывается необходимые условия для запроса. Например, при задании отобрать товары в ценовой категории Больше 200 в Условии отбора ставится >200
9. Сохранение запроса для дальнейшего использования производится нажатием на панели инструментов кнопки Сохранить. Далее СУБД запросит имя сохраняемого запроса. Целесообразно, чтобы оно имело смысловую нагрузку, что облегчит дальнейшее использование запроса.

Чтобы удалить лишнюю или внесенную по ошибке базовую таблицу из запроса, необходимо выделить ее, щелкнув на любом месте в списке ее полей, и нажать клавишу **Delete.** Чтобы удалить поле из запроса, выделите нужный столбец в бланке запроса, а затем нажмите клавишу **Delete.**

Самое главное в запросе - возможность использования критериев выборки, которые вводятся в строку Условие отбора. Можно выделить следующие типы запросов на основе критериев выборки:

- ***Выборка по строгому совпадению.*** В строку Условие отбора для определенного поля вводится одно из значений, существующих в таблице. Например, название конкретного товара или название фирмы, города.

Данные запросы можно параметризовать, т.е. вводить условия отбора в виде параметра при каждом запуске запроса, что устраняет необходимость предварительно его модификации. Для параметризации необходимо в строке Условие отбора вместо самого условия ввести текст приглашения на его ввод по формату [текст приглашения]. Например, [Введите наименование организации].

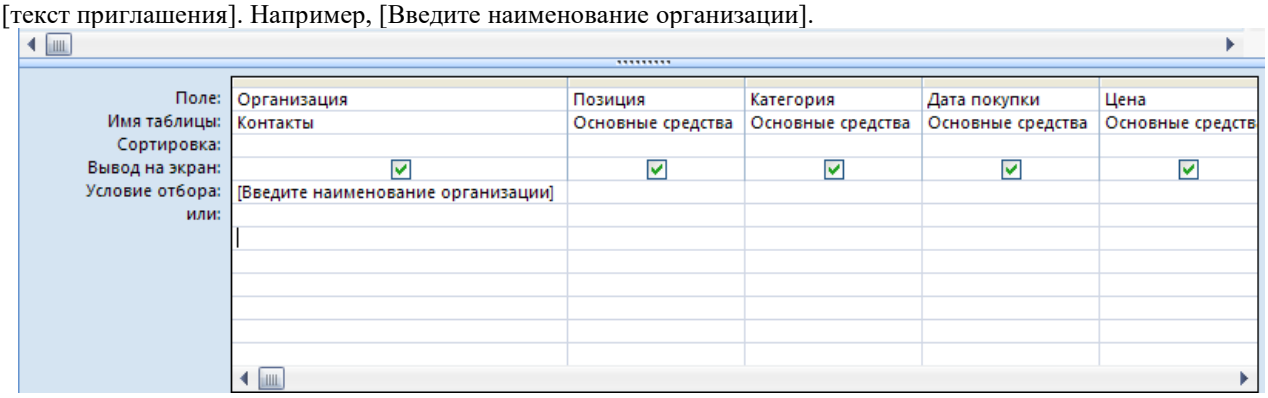



Рис. 21. Окно создания параметризированного запроса

Затем запрос сохраняется обычным способом. Чтобы оценить результаты сформированного запроса, необходимо его запустить двойным щелчком мыши. При запуске параметризованного запроса появляется диалоговое окно (рис. 2.4), в котором пользователь должен ввести собственно условие отбора и нажать клавишу ОК.

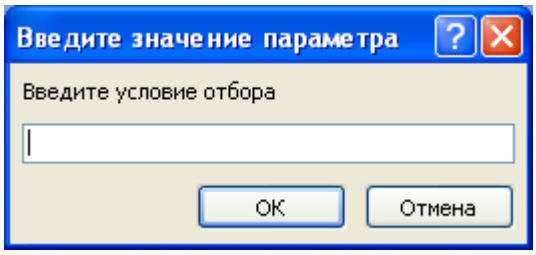



Рис. 22. Диалоговое окно запуска параметризированного запроса

- ***Выборка по строгому несовпадению.*** В этом случае в выборку отбираются все записи таблицы, кроме записей, содержащих значение, указанное в строке Условие отбора. Для реализации данного запроса перед значением вводится префикс Not или <>. Например, Not "МТФ" в поле Факультет запроса к таблице СТУДЕНТ приведет к выборке всех студентов вуза, кроме студентов МТФ.
- ***Выборка по неточному совпадению.*** Для выборки записей в условиях неполноты знаний о требуемых значениях используется оператор Like <условие>.

В списке можно указывать сразу диапазон символов. Например, **Like** "[B-M]*" - выбираются все студенты, ФИО которых начинается на буквы от "B" до "M". Длина ФИО произвольная.

- ***Выборка по диапазону.*** Для формирования данных условий выбора используются операторы сравнения >, >=, <, <= и <>. Операции сравнения могут связываться логическими операциями And (И) и Or (ИЛИ). Для этих же целей используется оператор диапазона Between <нижнее_значение> and <верхнее_значение>. Например, выбор книг стоимостью от 100 до 200 рублей может быть реализован через ввод в запросе условия в поле Стоимость в виде >=100 and <=200 или Between 100 and 200.

Перечень значений в условии выборки можно задать и оператором **In** (значение, значение, ...). Например, выбор студентов факультетов МТФ или ФАПУ можно реализовать, указав в поле *Факультет* запроса условие **In** ("МТФ", "ФАПУ"). Это же условие можно записать и через операцию ИЛИ: "МТФ" **or** "ФАПУ". Также можно указать одно название факультета в строке **Условие отбора** (см. рис. 2.2), а второе в следующей строке **или.** Число строк **или** не ограничено.

В выражениях отбора также можно использовать знаки математических операций +, -, /, * и неограниченное число круглых скобок. Сложные выражения в условиях отбора могут формироваться с помощью соответствующего построителя, который вызывается кнопкой  на панели инструментов.

- ***Запрос с вычислениями.*** Такой запрос позволяет получить дополнительную информацию в процессе выборки, например, стоимость всей партии товара при хранимой в таблице информации о количестве товара и стоимости единицы его продукции. Для этого в строку Поле пустого столбца заносят выражение для вычисления по следующему формату:

  <Название_формируемого_поля>:<выражение>.

  Например: Стоимость партии:[Товар]![количество товара]*[стоимость единицы товара].
- ***Запрос с групповыми операциями.*** Рассмотренные запросы анализируют отдельные записи таблицы. Вместе с тем, СУБД Access позволяет находить интегральные показатели для групп записей в таблице. Каждая такая группа характеризуется одинаковым значением по какому-то полю, например, одинаковым названием факультета или семейным положением. Для перехода в данный режим запросов необходимо в панели инструментов нажать клавишу Групповые операции , что приведет к появлению в бланке запроса новой второй строки с одноименным названием. В ячейках данной строки указывается или режим группировки по некоторому полю (опция Группировка), или название групповой операции:
  - **Sum** - сумма значений
  - **Avg** - среднее значение по данному полю для всей группы;
  - **Count** - число записей в данной группе;
  - **Max** - максимальное значение поля в каждой группе;
  - **Min** - минимальное значение поля в каждой группе;
  - **First** - первое значение данного поля в каждой группе;
  - **Last** - последнее значение данного поля в каждой группе и др.

  Опции выбора вызываются нажатием кнопки раскрытия в требуемой ячейке.

При запуске запроса СУБД разбивает таблицу на группы, число которых равно числу существующих значений в группируемом поле, и реализует для каждой группы требуемую операцию, т.е. число строк в выборке равно числу групп.

Рассмотренные запросы не изменяют содержимое исходной таблицы. Для реализации подобных действий СУБД Access использует четыре следующих запроса:

- *Запрос-создание новой таблицы.* Предназначен для сохранения результатов запроса в виде новой таблицы.

Исходно формируется обычный запрос на выборку необходимой информации из таблицы. После проверки результатов его выполнения производится возврат в режим конструктора запросов. Далее нажимается кнопка **Тип запроса**  на панели инструментов или выбирается команда главного меню **Запрос.** В появившемся списке выбирается опция **Создание таблицы**, после чего СУБД запрашивает её имя. Указывается имя создаваемой таблицы и нажимается кнопка **ОК.** Непосредственно запрос на создание запускается нажатием кнопки  на панели инструментов.

В окне **Таблицы** БД появляется пиктограмма созданной таблицы.

- *Запрос-добавление выборки в другую таблицу.* Выборку можно добавить к другой таблице, однотипной по структуре или с изменением структуры выборки.

Для этого необходимо сформировать обычный запрос и оценить результаты его выполнения. Далее следует вернуться в режим конструктора запроса.

Нажимается кнопка **Тип запроса** на панели инструментов или выбирается команда главного меню **Запрос.** В появившемся списке выбирается опция **Добавление**, после чего СУБД запрашивает имя таблицы, к которой будет добавлена выборка. Последний шаг - нажатие кнопки **ОК**.

Выборку можно добавлять и к таблицам других БД, что определяется установкой соответствующих переключателей в окне ввода имени целевой таблицы.

Если структура выборки и целевой таблицы не совпадают, то в целевую таблицу добавляются значения только тех полей выборки, имена которых совпадают с именами полей целевой таблицы.

- *Запрос-удаление.* С помощью запросов можно удалить часть или все записи из таблицы.

Для этого необходимо сформировать обычный запрос и оценить результаты его выполнения. Далее следует вернуться в режим конструктора запроса.

Нажимается кнопка **Тип запроса** на панели инструментов или выбирается команда главного меню **Запрос.** В появившемся списке выбирается опция **Удаление**, после чего в бланке запроса появляется новая третья строка с именем **Удаление**, куда можно вводить дополнительные условия на выборку удаляемых записей. Последний шаг - нажатие кнопки **ОК**.

- *Запрос-обновление.* С помощью запросов можно обновлять в единой операции некоторые или все значения выбранных полей.

Для этого необходимо сформировать обычный запрос и оценить результаты его выполнения. Далее следует вернуться в режим конструктора запроса.

Нажимается кнопка **Тип запроса** на панели инструментов или выбирается команда главного меню **Запрос.** В появившемся списке выбирается опция **Обновление**, после чего в бланке запроса появляется новая третья строка с именем **Обновление**. В ней задаются новые значения полей таблицы, в том числе и вычисляемые выражения. Далее запрос запускается на выполнение кнопкой . СУБД указывает число модифицируемых записей и просит подтвердить изменения кнопкой **ОК**. Пользователь на этом этапе еще может отказаться от модификации значений в таблице.

В предыдущих лабораторных работах мы уже использовали так называемый ***Построитель выражений***. Его используют для построения сложных выражений в Access. Так, если вам в работе потребуется применить встроенную функцию, щелкните в построителе выражений по папке Функции в самом левом списке, чтобы вывести категории функции.

Например, в одном из заданий вам необходимо вывести в поле запроса Фамилию +инициалы. Для этого следует использовать встроенную функцию Left текстового типа.

Затем вам необходимо набрать выражение

*ФИО:*

*[Сотрудники]![ФАМИЛИЯ]+"пробел"+Left([Сотрудники]![Имя];1)+"точка"+Left([Сотрудники]![Отчество];1)+"точка"*

*В кавычках ставится реальный пробел и реальная точка.*

*Таким образом, в вашем запросе будут отражаться Иванов И.И.*

**Порядок выполнения работы:**

Для всех вариантов задания 1 сформировать и выполнить следующие запросы:

1. Запрос-выборку по одной таблице с использованием параметров.
2. Запрос-выборку по нескольким таблицам и запросам с использование вычисляемых полей.
3. Запрос на обновление  данных.
4. Запрос на удаление данных.

5. Запрос на создание таблицы.
6. Итоговый запрос.
7. Перекрестный запрос.

Предложенные ниже варианты запросов попытаться выполнить с помощью запросов по образцу QBE, в противном случае реализовать SQL-запросы.

Варианты заданий

Вариант 1.
1. Определить покупателя, который купил максимальное количество товаров.
2. Для каждой покупки рассчитать общую стоимость.
3. Определить сумму продажи для каждого месяца.
4. Определить покупателей, купивших товаров на сумму, превышающую среднюю сумму покупок всех покупателей.
5. Определить тип, товаров которого куплено больше всего.

Вариант 2.
1. Для каждого вида изделия рассчитать его стоимость.
2. Найти изделия, в состав которых входит больше всего компонентов.
3. Определить компоненты, которые входят в большее число изделий.
4. Вычислить прибыль от продажи каждого типа продукции.
5. Найти изделие, на сборку которого уходит дней больше, чем в среднем на сборку изделий.

Вариант 3.
1. Определить заказ, на выполнение которого ушло больше всего дней.
2. Определить клиентов, стоимость заказов которых превысила их кредит.
3. Рассчитать стоимость каждого заказа с учетом скидки.
4. Определить клиента, который купил больше всего товаров.
5. Определить город, где живет клиент, чаще других оформляющий заказы.

Вариант 4.
1. Рассчитать для каждого квартиросъемщика квартплату.
2. Определить задолжников по квартплате за каждый месяц.
3. Определить дом с максимальной жилой площадью.
4. Определить дом с максимальной плотностью населения.
5. Жильцам, просрочившим оплату жилья, назначить пени 1% за каждый просроченный день.

Вариант 5.
1. Рассчитать общую стоимость товара с учетом транспортных расходов, скидки и налога.
2. Определить прибыль от продажи за каждый месяц
3. Определить страну, в которой изготовлены компоненты, вошедшие в товар, пользующийся наибольшей популярностью.
4. Определить самый дешевый компонент, поступающий без лицензии.
5. Определить товар, в состав которого входят компоненты с максимальными транспортными расходами.

Вариант 6.

1. Вычислить оплату труда каждого сотрудника как нижняя грань оплаты, если сотрудник работает меньше года, средняя оплата, если сотрудник работает от года до пяти лет, и верхняя грань - более пяти лет.
2. Для каждого сотрудника рассчитать его ежемесячный заработок.
3. Найти сотрудника, который работает дольше других.
4. Вычислить сумму налога, которую фирма платить каждый месяц.
5. Определить сотрудников, ежемесячная оплата которых оказалась больше средней.

Вариант 7.
1. Определить услугу, пользующую наибольшей популярностью.
2. Для каждого клиента рассчитать стоимость услуг с учетом социального положения и скидок.
3. Определить доход фирмы от предоставленных услуг за каждый месяц.
4. Определить, жители города или села чаще всего обращаются в фирму.
5. Рассчитать количество и сумму предоставленных населению услуг по категориям, определенным социальным происхождением клиентов.

Вариант 8.
1. Для каждого клиента вычислить сумму оплаты междугородних разговоров.
2. Определить город, с которым чаще всего разговаривают клиенты.
3. Определить клиента, который говорит по телефону чаще и дольше других.
4. Определить время суток, на которое приходится больше всего разговоров.
5. Определить день, в который телефонная линия была занята меньше всего.

Вариант 9.
1. Вычислить стоимость и калорийность каждого блюда.
2. Определить блюдо из супов с наименьшим содержанием жиров.
3. Определить компоненты самого дорогого блюда.
4. Найти компонент, который входит в большинство блюд.
5. Определить содержание жира, белков и углеводов в самом дорогом блюде самого дешевого в среднем типа блюд.

Вариант 10.
1. Определить тематику, по которой продается больше всего книг.
2. По каждому месяцу вычислить сумму продаж.
3. Определить, книги каких авторов пользуются наибольшей популярностью, авторов мужчин или авторов-женщин.
4. Определить дни, когда было продано книг больше, чем обычно (т.е. больше среднего).
5. Какие по тематике книги пишут молодые авторы.

Вариант 11.
1. Вычислить сумму продаж по каждому месяцу.
2. Определить страну, выпустившую самый долгозвучащий диск.
3. Определить песню, пользующуюся наибольшей популярностью.
4. Составить рейтинг исполнителей по каждому месяцу.
5. Определить автора слов, написавшего больше всех песен.

Вариант 12.
1. Определить сумму продаж по каждому месяцу.

2. Определить, какой тип покупателей чаще других покупает видеокассеты.
3. Какой самый старый фильм был продан за последний месяц.
4. Определить страну, завоевавшую своими фильмами больше сего Оскаров.
5. Какие по тематике фильмы смотрит молодежь.

Вариант 13.
1. Для каждой олимпиады рассчитать отношения числа завоеванных медалей к числу участников.
2. Определить команду, для которой отношение числа завоеванных золотых медалей к числу участников больше, чем аналогичный показатель олимпиады.
3. Определить команды, которые чаще других участвовали в олимпиадах.
4. Определить команды, которые по числу завоеванных медалей на протяжении всех олимпиад попадали в первую тройку.
5. Найти олимпиады, символы которых совпадали.

Вариант 14.
1. Определить предмет, по которому нет двоек.
2. Определить студентов, сдавших успешно экзамены и набравших в сумме часов больше, чем среднее число часов по всем студентам.
3. Определить блок дисциплин, средняя оценка по которым самая высокая.
4. Определить кафедру, по предметам которой получено больше всего двоек студентами младших курсов.
5. Найти студентов, сдавших все экзамены успешно, если их день рождения пришелся на период сдачи экзаменов.

Вариант 15.
1. Вычислить зарплату каждого преподавателя.
2. Определить блок дисциплин, которые читают самые квалифицированные преподаватели.
3. Определить кафедру, для которой отношение числа предметов к числу преподавателей самое большое.
4. Определить семестр и кафедры, на которые приходится учебная нагрузка в часах, больше средней по кафедрам.
5. Определить предмет, который читается в большинстве семестров.

Вариант 16.
1. Вычислить прибыль от каждого рейса.
2. Определить рейсы до заданного пункта, на которые остались свободные места.
3. Найти отношение количества рейсов дальнего следования, которые выполняют квалифицированные командиры экипажей, к общему числу рейсов дальнего следования.
4. Какой экипаж имеет больше всего налетов , по количеству и по продолжительности.
5. Какие пассажиры по своему социальному положению летают чаще других.

Вариант 17.
1. Определить водителей с плохим техническим состоянием автобуса6 чаще других отправляющихся в рейс.
2. Вычислить прибыль от поездок за каждый месяц.
3. Какие поездки пользуются наибольшей популярностью.
4. Для каждой организации рассчитать долг с учетом стоимости поездок и выплаченного аванса.

5. Определить водителя, совершившего больше всего поездок.

Вариант 18.
1. Для каждого вуза рассчитать объем свободных наличных средств.
2. Определить факультеты с самым большим отношением числа студентов к числу преподавателей.
3. Определить вуз с самой низкой средней стоимостью обучения одного студента.
4. Найти вуз с самым большим числом хоздоговорных студентов.
5. Определить, какая сумма приходится на каждого студента.

Вариант 19.
1. Определить, преступления какого вида раскрываются быстрее других.
2. Оружие какой страны наиболее часто используется в преступлениях.
3. Определить сколько преступлений и какого вида приходится на каждую возрастную группу.
4. В каком городе преступления раскрываются быстрее, чем в других городах.
5. В какой месяц было совершено больше всего преступлений.

Вариант 20.
1. В каком месяце была продана самая дорогая из старых машин.
2. Машины какой страны пользуются популярностью у молодежи.
3. Определить сумму продаж за каждый месяц.
4. Определить, какая возрастная группа покупает в среднем самые дорогие автомобили.
5. Какая группа по социальному положению предпочитает при расчете кредитные карточки.

**Контрольные вопросы:**

1. Для чего служат запросы? Назовите типы запросов.
2. Как создать в запросе вычисляемое поле?
3. Параметризированный запрос, назначение и порядок выполнения.
4. Как осуществляется выборка сотрудников по фамилии в диапазоне, например, от В до М?
5. Для чего используют запросы на добавление и удаление?
6. Каково назначение групповых операций при формировании запроса?
7. Для чего используют встроенные функции? Назовите перечень операций, позволяющий их использовать.

# Лабораторная работа №5

## *Введение ограничений целостности базы данных в СУБД Access*

**Цель работы:** изучение и закрепление на практике методов обеспечения целостности данных в реляционных базах данных.
**Результат:** mdb-файл (accdb-файл), содержащий базу данных в формате MS Access с выполненным заданием.
**Краткая теория:**

Целью обеспечения целостности данных является предотвращение появления непарных записей, ссылающихся на несуществующие записи. Обеспечение целостности данных включается для конкретного отношения между таблицами. В результате Access отменяет для этого отношения все действия, которые могут нарушить целостность данных. Это означает, что будет отменено как обновление, изменяющее целевой объект ссылки, так и удаление такого целевого объекта. Сведения о том, как настроить в Access распространение операций обновления и удаления таким образом, чтобы в результате изменялись и все связанные строки.

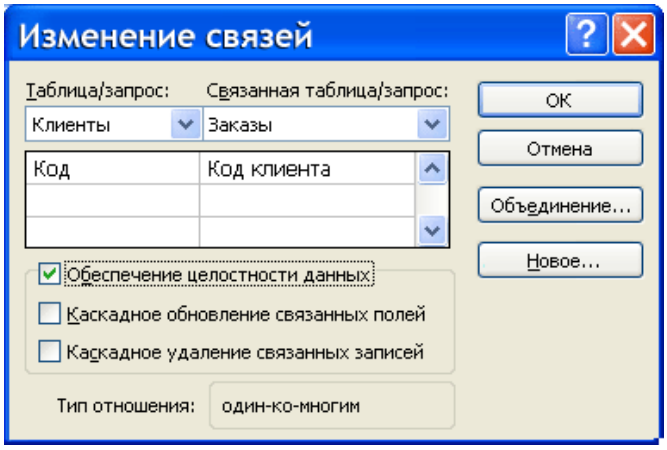



Рис.23. Диалоговое окно **Изменение связей**.

### Включение и отключение обеспечения целостности данных

1. На вкладке **Работа с базами данных** в группе **Отношения** нажмите кнопку **Схема данных**.
2. На вкладке **Конструктор** в группе **Связи** нажмите кнопку **Все связи**. Будут отображены все таблицы с отношениями, а также соответствующие линии связи. Обратите внимание на то, что скрытые таблицы (таблицы, для которых установлен флажок **скрытый** в диалоговом окне **Свойства**) и их отношения не отображаются, если в диалоговом окне **Параметры переходов**не выбран параметр "Показывать скрытые объекты".
3. Щелкните линию отношения, которое вы хотите изменить. При выделении линия связи становится толще.
4. Дважды щелкните линию связи. Откроется диалоговое окно **Изменение связей**.
5. Выберите или отмените параметр **Обеспечение целостности данных**.
6. Внесите в отношение необходимые изменения и нажмите кнопку **ОК**.

В режиме обеспечения целостности данных действуют перечисленные ниже правила.

- Не допускается ввод в поле внешнего ключа связанной таблицы значений, отсутствующих в поле первичного ключа главной таблицы, поскольку это приводит к появлению потерянных записей.
- Не допускается удаление записи из главной таблицы, если в связанной таблице существуют связанные с ней записи. Например, невозможно удалить запись из таблицы "Сотрудники", если в таблице "Заказы" имеются заказы, относящиеся к данному сотруднику. Однако можно удалить главную запись и все связанные записи одним действием, установив флажок Каскадное удаление связанных записей.
- Не допускается изменение значения первичного ключа в главной таблице, если это приведет к появлению потерянных записей. Например, нельзя изменить номер заказа в таблице "Заказы", если в таблице "Сведения о заказах" имеются строки, относящиеся к этому заказу. Однако можно обновить

главную запись и все связанные записи одним действием, установив флажок "Каскадное обновление связанных полей".

**Примечания:** Если при включении обеспечения целостности данных возникли трудности, обратите внимание на то, что должны выполняться условия, перечисленные ниже.

- Общее поле главной таблицы должно быть первичным ключом или иметь уникальный индекс.
- Общие поля должны иметь одинаковый тип данных. Единственное исключение — поле типа "Счетчик" можно связать с полем типа "Числовой", если его свойство Размер поля имеет значение Длинное целое.
- Обе таблицы существуют в одной базе данных Access. Обеспечение целостности данных нельзя включить для присоединенных таблиц. Однако если исходные таблицы имеют формат Access, можно открыть базу данных, в которой они хранятся, и включить обеспечение целостности данных в этой базе.

## Задание каскадных параметров

Иногда возникает ситуация, в которой требуется изменить значение только на стороне "один" отношения. В этом случае необходимо, чтобы Access автоматически обновил все затронутые строки в ходе выполнения одной операции. Тогда обновление будет полностью завершено, а база данных не будет находиться в несогласованном состоянии, когда некоторые строки обновлены, а другие — нет. Этой проблемы можно избежать с помощью параметра Access "Каскадное обновление связанных полей". Если при включении обеспечения целостности данных был включен параметр "Каскадное обновление связанных полей", то при последующем обновлении первичного ключа автоматически будут обновлены все связанные с ним поля.

Также может потребоваться удаление строки и всех связанных с ней записей — например, записи в таблице "Поставщики" и всех связанных с этим поставщиком заказов. Для этого в Access предназначен параметр "Каскадное удаление связанных записей". Если включить обеспечение целостности данных и установить флажок **Каскадное удаление связанных записей**, при удалении записи, содержащей первичный ключ, будут автоматически удалены все записи, связанные с этим первичным ключом.

### Включение и отключение каскадного обновления и каскадного удаления

1. На вкладке **Работа с базами данных** в группе **Отношения** нажмите кнопку **Схема данных**.
2. На вкладке **Конструктор** в группе **Связи** нажмите кнопку **Все связи**. Будут отображены все таблицы с отношениями, а также соответствующие линии связи. Обратите внимание на то, что скрытые таблицы (таблицы, для которых установлен флажок **скрытый** в диалоговом окне **Свойства**) и их отношения не отображаются, если в диалоговом окне **Параметры переходов** не выбран параметр "Показывать скрытые объекты".
3. Щелкните линию отношения, которое вы хотите изменить. При выделении линия связи становится толще.
4. Дважды щелкните линию связи. Откроется диалоговое окно **Изменение связей**.
5. Установите флажок **Обеспечение целостности данных**.
6. Установите флажок **Каскадное обновление связанных полей**, **Каскадное удаление связанных записей** или оба эти флажка.
7. Внесите в отношение необходимые изменения и нажмите кнопку **ОК**.

**Порядок выполнения работы:**

1. С помощью механизма каскадных изменений обеспечить поддержку ссылочной целостности Вашей базы данных.
2. Обеспечить семантическую поддержку целостности Вашей базы данных. Для каждого поля базы данных задать следующие (подходящие по смыслу) виды декларативных ограничений целостности:
   - ограничения целостности атрибута: значение по умолчанию, задание обязательности или необязательности значений (Null), задание условий на значения атрибутов;
   - задание значения по умолчанию.

**Кафедра цифровой экономики**

# БАЗЫ ДАННЫХ

**Методические указания**
**к лабораторным работам №№ 1,2,3,4,5 для студентов направления подготовки 080500 «Бизнес-информатика»**
**2 семестр**

Ульяновск
2018

# Лабораторная работа №6

## *Разработка информационной системы для работы с базой данных (PHP, Apache, MySQL)*

**Цель работы**: Приобретение навыков доступа к базам данных в сети Интернет, используя возможности PHP. Задачами лабораторной работы являются овладение навыками создания и заполнения таблиц баз данных, создания представлений, триггеров и хранимых процедур, освоение программных технологий доступа к базам данных MySQL с помощью серверных сценариев PHP.
**Задание:** Спроектировать базу данных формата MySQL. Разработать PHP-скрипт обработки данных.
**Формат результата:** Работающий скрипт и заполненная база данных.

**Теоретическая часть:**

**СУБД MySQL.** Первоначально сервер MySQL разрабатывался для управления большими базами данных с целью обеспечить более высокую скорость работы по сравнению с существующими на тот момент аналогами. И вот уже в течение нескольких лет данный сервер успешно используется в условиях промышленной эксплуатации с высокими требованиями. Несмотря на то, что MySQL постоянно совершенствуется, он уже сегодня обеспечивает широкий спектр полезных функций. Благодаря своей доступности, скорости и безопасности MySQL очень хорошо подходит для доступа к базам данных по Интернету.

**GPL.** MySQL - это программное обеспечение с открытым кодом. Это означает, что применять и модифицировать его может любой желающий. Такое ПО можно получать по Internet и использовать бесплатно. При этом каждый пользователь может изучить исходный код и изменить его в соответствии со своими потребностями. Использование программного обеспечения MySQL регламентируется лицензией GPL (GNU General Public License), в которой указано, что можно и чего нельзя делать с этим программным обеспечением в различных ситуациях. MySQL можно загрузить с веб-сайта http://www.mysql.com/.

**Учетные записи MySQL.** У MySQL есть собственный интерфейс для организации взаимодействия с клиентами, с помощью которого можно перемещать данные и изменять параметры баз данных. Чтобы иметь возможность работать с базой данных, необходимы учетная запись и пароль. Каждый сервер MySQL может содержать несколько баз данных, где группируются таблицы. Если MySQL установлен на локальном компьютере, то по умолчанию именем пользователя является root.

**Форматы таблиц MySQL.** В MySQL можно было выбирать из семи основных форматов таблиц, наиболее распространенными из которых являются ISAM и InnoDB.

Принятым по умолчанию типом таблиц в MySQL является MyISAM. Если попытаться воспользоваться таблицей, которая не была активизирована или добавлена при компиляции, MySQL вместо нее создаст таблицу типа MyISAM. Это очень полезная функция, когда необходимо произвести копирование таблиц с одного SQL-сервера на другой, а серверы поддерживают различные типы таблиц (например, при копировании таблиц на подчиненный компьютер, который оптимизирован для быстрой работы без использования транзакционных таблиц).

Таблицы InnoDB в MySQL снабжены обработчиком таблиц, обеспечивающим безопасные транзакции (уровня ACID) с возможностями фиксации транзакции, отката и восстановления после сбоя. Для таблиц InnoDB осуществляется блокировка на уровне строки, а также используется метод чтения без блокировок в команде SELECT. Перечисленные функции позволяют улучшить взаимную совместимость и повысить производительность в многопользовательском режиме. В InnoDB нет необходимости в расширении блокировки, так как блоки строк в InnoDB занимают очень мало места. Важно, что для таблиц InnoDB поддерживаются ограничивающие условия FOREIGN KEY.

**Инструментарий phpMyAdmin.** Инструмент phpMyAdmin позволяет администрировать MySQL с помощью обычного браузера. Все, что требуется для работы с этим инструментом, - это веб-сервер с установленным PHP и база данных MySQL, которую нужно администрировать. Инструмент администрирования позволяет увидеть параметры настройки базы данных и имеющиеся в ней объекты (например, таблицы), а также добавлять новые таблицы при помощи графического интерфейса. С помощью phpMyAdmin можно создавать новые базы данных и таблицы, запускать запросы и просматривать статистику работы сервера.

**Ограничения в MySQL.** Важной особенностью (и недостатком) MySQL является отсутствие поддержки ограничений уровня столбцов таблиц (например, даже такое простое ограничение, как проверка принадлежности числа заданному диапазону). При этом реализация ограничений с помощью программного

кода (запросов check) возможна, но не действенна: написанные запросы будут без проблем выполнены СУБД, но проигнорированы при работе с таблицами.

Рассмотренные ранее СУБД Access и SQL Server решали задачу создания ограничений целостности с помощью соответствующих ограничений (типа Check), доступных как в графическом, так и программном (с помощью непосредственного ввода кода) режимах. Те же самые задачи в MySQL решаются намного сложнее и требуют написания соответствующих программных триггеров.

**Триггеры в MySQL.** Поддержка для триггеров включена, начиная с MySQL 5.0.2. Триггер представляет собой именованный объект базы данных, который связан с таблицей, и он будет активизирован, когда специфическое событие происходит для таблицы.

В общем виде программный код создания триггера имеет следующий вид:

```
CREATE  TRIGGER trigger_name trigger_time trigger_event
ON tbl_name FOR EACH ROW trigger_stmt
```

В приведенном фрагменте параметр trigger_time задает время действия. Это может быть BEFORE или AFTER, чтобы задать, что триггер активизируется прежде или после инструкции, которая активизировала это.

Следующий параметр - trigger_event - указывает вид инструкции, которая активизирует триггер. Здесь trigger_event может быть одним из следующего:

- INSERT: всякий раз, когда новая строка вставлена в таблицу. Например, через команды INSERT, LOAD DATA или REPLACE.
- UPDATE: всякий раз, когда строка изменяется. Например, через инструкцию UPDATE.
- DELETE: всякий раз, когда строка удалена из таблицы. Например, через инструкции DELETE и REPLACE.

Важно, что не может быть двух триггеров для данной таблицы, которые имеют те же самые время действия и событие. Например, не может быть два триггера BEFORE UPDATE для таблицы, но возможны BEFORE UPDATE и BEFORE INSERT или BEFORE UPDATE и AFTER UPDATE.

Следующий параметр - trigger_stmt - задает инструкцию, которая будет выполнена, когда триггер активизируется. Если нужно выполнить много инструкций, используется операторная конструкция BEGIN ... END. Это также дает возможность использовать те же самые инструкции, которые являются допустимыми внутри сохраненных подпрограмм.

Схема взаимодействия серверного php-приложения с базой данных MySQL по приведенному алгоритму проиллюстрирована на рис. 1.

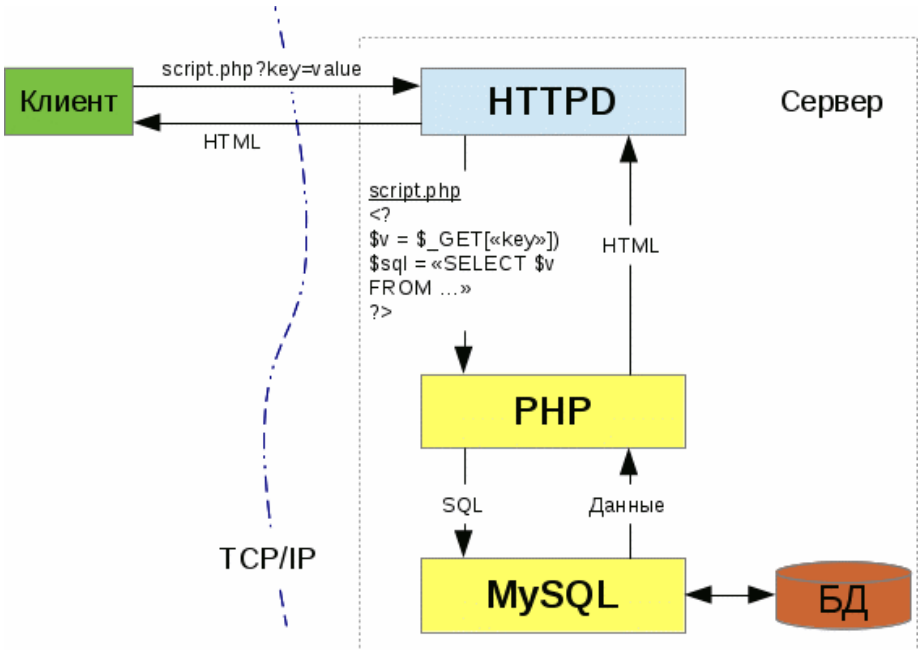



Рис. 24. Взаимодействие php-скрипта и БД MySQL

**MySQL и PHP.** Язык PHP (PHP: Hypertext Preprocessor) представляет собой язык сценариев, которые внедряются в страницы HTML для исполнения на стороне сервера. Как правило, для отделения PHP-кода от кода HTML используются символы <?php код ?>, реже - <? код ?> и <script language="php">код</script>, а также инструкции «в стиле ASP» - <% код %>.

Web-страница PHP имеет расширение .php и состоит из трех разделов:

- - директивы страницы - используются для настройки и определяют, как должна обрабатываться страница. Например, так можно задать подключение внешних файлов.
- - код - программный код, реализующий выполняемые на сервере операции.
- - разметка страницы - это HTML-код страницы, включающий тег body и его содержимое.

Поддержка PHP обеспечивается многими серверами, но традиционно такие Web-приложения функционируют на Web-сервере Apache.

Приложения PHP, использующие для хранения информации базу данных, как правило, работают с системой управления базами данных MySQL. Для работы с базами данных PHP располагает достаточно широким набором функций (табл. 2) - от установки соединения с базой данных до извлечения отдельных значений, полученных в результате выполнения запроса.

Таблица 2

Функции взаимодействия с базами данных в PHP-приложениях (СУБД MySQL)

| Название класса | Описание класса |
| --- | --- |
| mysql_connect | Соединение с источником данных |
| mysql_select_db | Выбор базы данных для дальнейшей работы |
| mysql_query | Отправка SQL-запросов серверу |
| mysql_fetch_array | Представление результата запроса в виде ассоциативного массива |
| mysql_result | Доступ к отдельному полю записи результата запроса |

**А. Создание и заполнение базы данных**

1. Запустить СУБД MySQL:
   Рабочий стол | ярлык Start Denwer ►окно запуска Denwer; дождаться закрытия окна
2. Открыть среду PHPMyAdmin:
   Запустить браузер ►адресная строка ← http://localhost ►окн. «Ура, заработало!» | выбр. http://localhost/Tools/phpMyAdmin ► окн. phpMyAdmin; Server: localhost
3. Создать новую базу данных *Education*:
   ст. MySQL localhost | Create database: | ввести *Education* | Collation ← utf8_unicode_ci |
   кн. Create ►окн. Server: localhost; Database: Education |
   Database Education has been created.
4. В базе данных с помощью конструктора таблиц создать структуру таблицы *Students* (справочник студентов):

- создать новую таблицу *Students* с 11 полями:
  окн. Server: localhost; Database: Education | Create new table on database Education: | Name ← *Students*; Number of fields ← 11 | кн. Go ► окн. Server: localhost; Database: Education; Table: Students;
- задать поле идентификатора студента (счетчик, первичный ключ):
  ст. Field ← *stud_ID*; Type ← *int*; уст. *A_I*; Index ← *Primary*; перейти на след. строку;
- задать поле номера зачетной книжки (уникальное, 6 цифр):
  ст. Field ← *no_zk*; Type ← *varchar*; Length / Values[1] ← *6*;
  Index ← *Unique*; перейти на след. строку;
- задать поля фамилии, имени, отчества (строковые):
  ст. Field ← *fam*; Type ← *varchar*; Length / Values[1] ← *20*; перейти на след. строку;

аналогично - для имени (*im*) и отчества (*ot*);

- создать уникальный индекс *ФИО* из полей *fam*, *im*, *ot*:
  ст. Field *fam* ← Index ← *Index*; перейти на след. строку;
  ст. Field *im* ← Index ← *Index*; перейти на след. строку;
  ст. Field *ot* ← Index ← *Index*;
- задать поле кода специальности (до 7 букв):
  ст. Field ← *spec*; Type ← *varchar*; Length / Values[1] ← *7*; перейти на след. строку;
- задать поле курса (одна цифра от 1 до 6):
  ст. Field ← *kurs*; Type ← *varchar*; Length / Values[1] ← *1*; перейти на след. строку;
- задать поле номера группы (две цифры):
  ст. Field ← *gr*; Type ← *varchar*; Length / Values[1] ← *2*; перейти на след. строку;
- задать поле даты рождения:
  ст. Field ← *data_r*; Type ← *date*; перейти на след. строку;
- задать поле биографии (мемо):
  ст. Field ← *biogr*; Type ← *longtext*; уст. *Null*; перейти на след. строку;
- задать поле фотографии (объект):
  ст. Field ← *foto*; Type ← *blob*; уст. *Null*; перейти на след. строку;

кн. Save ► окн. Table `Education`.`Students` has been created.

3. Аналогичным образом создать структуру таблицы *Subjects* (справочник предметов):

- создать новую таблицу базы данных *Education*:
  пан. phpMyAdmin | Database ← *Education (1)*;
  пан. Create new table on database Education: | Name ← *Subjects*; Number of fields ← 5 |
  кн. Go ► окн. Server: localhost; Database: Education; Table: Subjects.
- задать поле идентификатора предмета (счетчик, первичный ключ):
  ст. Field ← *predm_ID*; Type ← *int*; уст. *A_I*; Index ← *Primary*; перейти на след. строку;
- аналогичным образом создать остальные поля таблицы:
  - *name* (название предмета (строковое));
  - *cycle* (цикл дисциплин, к которым относится предмет (строковое));
  - *hours* (количество часов (числовое));
  - *dep* (название кафедры, на которой ведется преподавание предмета (строковое));
- сохранить структуру таблицы:
  кн. Save ► окн. Table `Education`.`Subjects` has been created.

4. Аналогичным образом создать структуру таблицы *Uspev* (успеваемость студентов по предметам) с полями:

- *stud* (идентификатор студента (тип такой же, как в таблице *Студенты*, но не счетчик));
- *predm* (идентификатор предмета (тип такой же, как в таблице *Предметы*, но не счетчик));
- *ocenka* (оценка (символ));
- *data* (дата (дата)).

- ввести составной первичный ключ (поля идентификаторов студента и предмета):
  Поле *stud* | Index ← *Primary*; перейти на след. строку;
  Поле *predm* | Index ← *Primary*;
- сохранить структуру таблицы:
  кн. Save ► окн. Table `Education`.`Uspev` has been created.

5. Задать ограничения для столбцов таблицы *Students* в форме триггера *students_constraints*, срабатывающего при попытке вставки новой записи в таблицу:

- создать новый SQL-запрос для таблицы *Students*:
  пан. phpMyAdmin | Database: ← *Education (1)*; выбр. *Students*; вкл. SQL ►
  окн. Run SQL query/queries on database Education | удалить текст «SELECT * FROM `Students` WHERE 1»;
- ввести SQL-запрос на удаление триггера *students_constraints_insert* в случае его существования (для возможности редактирования текста триггера):

```
DROP TRIGGER IF EXISTS `students_constraints_insert` ;
```

- ввести «заголовочный» код создания триггера:

```
delimiter ||
CREATE TRIGGER `students_constraints_insert` BEFORE INSERT ON `Students`
FOR EACH
ROW BEGIN
```

- ввести код, проверяющий, что значение поля *kurs* лежит в пределах от 1 до 6 и заменяющий его на 0 в противном случае:

```
IF not(NEW.kurs >=1 and NEW.kurs <= 6) THEN
SET NEW.kurs = 0;
END IF;
```

- ввести код, проверяющий, что значение поля *spec* вводится русскими буквами и заменяющий его на 0 в противном случае:

```
IF not(NEW.spec >= 'A' AND NEW.spec <= 'я') THEN
SET NEW.spec = 0;
END IF;
```

- ввести код, проверяющий, что значение поля *gr* лежит в пределах от 1 до 99 и заменяющий его на 0 в противном случае:

```
IF not(NEW.gr >='01' AND NEW.gr <= '99') THEN
SET NEW.gr = 0;
END IF;
```

- ввести завершающий код триггера:

```
END ||
delimiter ;
```

6. Аналогичным образом запрограммировать триггер *students_constraints_update*, срабатывающий при попытке изменения записи в таблице.

7. Для таблиц *Subjects* и *Uspev* написать по два триггера, проверяющие целостность данных при вставке и изменении данных.

8. Установить связи между таблицами и указать правила ссылочной целостности:

- изменить тип таблицы *Student*:

пан. phpMyAdmin | Database: ← *Education (1)*; выбр. *Students*; вкл. Operations ►
Storage Engine ← *InnoDB*;

аналогично для всех остальных таблиц.

- указать поле *stud* в качестве внешнего ключа таблицы *Students*, ссылающегося на первичный ключ таблицы *Students*:

пан. phpMyAdmin | Database: ← *Education (3)*; выбр. *Uspev*; вкл. Operations ►
выбр. Relation View ► ф. Links to | стр. *stud*; Foreign key (InnoDB) ← `Education`.`Students`; onDelete ← cascade; onUpdate ← cascade;

аналогичным образом задать внешний ключ, ссылающийся на первичный ключ таблицы *Subjects*.

8. Наполнить базу данных сведениями о студентах (не менее 5), предметах (не менее 3) и оценках (не менее 10):

пан. phpMyAdmin | Database: ← *Education (3)*; выбр. таблицу; вкл. Insert ►заполнить столбец *Value*.

Освоить приемы изменения и удаления полей и записей. Проверить работоспособность ограничений значений полей, уникальности и др., предусмотренные при задании структуры базы данных. Проверить работоспособность ссылочной целостности, удаляя, изменяя и вставляя данные.

**Б. Представления и хранимые процедуры**

1. Создать просмотр для вывода кратких сведений о студентах (идентификатор, номер зачетки, фамилия и инициалы, идентификатор группы):

- создать новый SQL-запрос для таблицы *Students*:

пан. phpMyAdmin | Database: ← *Education (3)*; выбр. *Students*; вкл. SQL ►
окн. Run SQL query/queries on database Education | удалить текст «SELECT * FROM `Students` WHERE 1»;

- ввести программный код запроса на извлечение данных:

```
SELECT `stud_ID`,`no_zk`,CONCAT(fam,' ',SUBSTRING(im,1),'.',SUBSTRING(ot,1),'.') AS FIO,
CONCAT(spec,'-',kurs,gr) AS Gruppa FROM `Students` WHERE 1
```

- выполнить запрос

окн. Run SQL query/queries on database Education | кн. Go ► результаты на экране;

- сформировать представление по результатам запроса:

ф. Query results operations | CREATE VIEW | VIEW name ← *Students_info* | кн. Go.

2. Аналогичным образом создать представление для вывода сведений об успеваемости студентов из таблицы успеваемости с указанием сведений о студенте из запроса *Students_info* и сведений о предмете из таблицы предметов. Результат должен содержать следующие поля: ФИО студента, Группа, Предмет, Дата, Оценка. Назвать представление как *Students_uspev*. Программный код запроса:

```
SELECT `no_zk`,`FIO`,`Gruppa`,`ocenka`,`data` FROM `students_info`,`uspev`,`subjects` WHERE
`students_info`.`stud_ID`=`uspev`.`stud ` AND
`subjects`.`predm_ID`=`uspev`.`predm `
```

3. Создать хранимую процедуру для вывода кратких сведений о студенте (идентификатор, фамилия и инициалы, идентификатор группы) по номеру его зачетной книжки:

- создать новый SQL-запрос для базы данных *Education*:

пан. phpMyAdmin | Database: ← *Education (3)*; вкл. SQL ►
окн. Run SQL query/queries on database Education;

- ввести программный код хранимой процедуры:

```
DELIMITER $$
CREATE PROCEDURE studs(IN IN_ZK VARCHAR(6),OUT V_ID INT, OUT V_ZK
VARCHAR(6),OUT V_FIO VARCHAR(25),OUT V_GR VARCHAR(25))
BEGIN
    SELECT stud_id, no_zk, concat(fam,' ',substring(im,1),'.',substring(ot,1),'.'),
    concat(spec,'-',kurs,gr)
    into V_ID, V_ZK, V_FIO, V_GR
    FROM Students
    WHERE `no_zk` = IN_ZK;
  END;
```

```
$$
DELIMITER ;
```

- проверить работоспособность хранимой процедуры:
  пан. phpMyAdmin | Database: ← *Education (3)*; вкл. SQL ►
  окн. Run SQL query/queries on database Education; ввести код:

```
CALL studs('номер зачетной книжки',@V_ID,@V_ZK,@V_FIO,@V_GR);
SELECT @V_ID,@V_ZK,@V_FIO,@V_GR;
```

  кн. Go ► Результат на экране.

**B. PHP и MySQL**

1. Запустить программную оболочку Denwer:
   Рабочий стол | ярлык Start Denwer ►окно запуска Denwer; дождаться закрытия окна
2. Проверить корректность работы Denwer:
   Запустить браузер ►адресная строка ← http://localhost ►окн. «Ура, заработало!»
3. Создать виртуальную директорию для хранения файлов веб-приложения:
   Мой компьютер | Локальный диск D | WebServers | denwer | www | denwer |
   создать папку под именем lab3_группа (например, lab3_asoi338).
   При наличии уже существующей папки с таким именем добавить постфикс «2»,
   например, lab3_asoi338_2.
4. Создать файл index.php начальной страницы веб-приложения:
   Директория lab3_группа | Создать текстовый документ |
   Переименовать файл как index.php
5. Открыть файл index.php с помощью редактора Notepad++:
   Директория lab3_группа | Файл index.php | пр. кн. мыши;
   выбр. Edit with Notepad++ ►окно редактора Notepad++
6. Ввести HTML-разметку страницы index.php:

```
<HTML>
  <HEAD><TITLE>Лабораторная работа 3</TITLE></HEAD>
  <BODY>
        <H2>Информация о студентах</H2>
        <FORM id="form" method="POST" action="index.php">
              <TABLE border="1" width="60%">
                    <TR>
                          <TH width="10%">Код</TH>
                          <TH width="20%">Зачетная книжка</TH>
                          <TH width="40%">ФИО</TH>
                          <TH width="30%">Группа</TH>
                    </TR>
                    <TR align="center">
                          <TD width="10%">Значение кода</TD>
                          <TD width="20%">Значение зачетки</TD>
                          <TD width="40%">Значение ФИО</TD>
                          <TD width="30%">Значение группы</TD>
                    </TR>
              </TABLE>
              <BR/>   Номер зачетной книжки: <input name="zk" type="text"/>
              <input type="submit" value="Запрос"/>
        </FORM>
  </BODY>
</HTML>
```

7. Проверить работоспособность созданной страницы:
   Браузер со стартовой страницей Denwer | строка адреса «http://localhost/denwer/» |
   дополнить названием виртуальной директории (см. п. 3),
   например, как  http://localhost/denwer/ lab3_asoi338 ►страница на экране
8. Создать PHP-сценарий соединения с базой данных Education:

- создать в виртуальной директории и открыть файл connection.php:
  Директория lab3_группа | Создать текстовый документ |
  Переименовать файл как connection.php | пр. кн. мыши;
  выбр. Edit with Notepad++ ►окно редактора Notepad++
- ввести программный код сценария:

- ввести открывающий тег сценария:
<?php
- ввести программный код соединения с локальным сервером:
$link = @mysql_connect("localhost", "root")
or die("Невозможно соединиться с сервером");
- ввести программный код соединения с базой данных Education:
$db=@mysql_select_db("Education") or die("Нет такой базы данных");
- ввести закрывающий тег сценария:
?>

- сохранить сценарий.

9. Дополнить файл index.php PHP-инструкциями:

- подключить сценарий соединения с базой данных:
окно редактора Notepad++ | вкл. Index.php | поставить курсор
до первой строки <HTML> | ввести код:

```
<?php
include("connection.php");
?>
```

- проверить работоспособность сценария, обновив в браузере страницу index.php.
- выполнить запрос к представлению Students_info базы данных:
окно редактора Notepad++ | вкл. Index.php | поставить курсор
после строки include("connection.php"); | ввести код:

```
$sql = "SELECT * FROM `students_info`";
$query = mysql_query($sql);
```

- организовать цикл по строкам таблицы:
окно редактора Notepad++ | вкл. Index.php | поставить курсор
перед второй строкой <TR> | ввести код:

```
<?php
        for($i=0;$i<$count;$i++)
        {
?>
```

поставить курсор после второй строки </TR> | ввести код:

```
<?php
        }
?>
```

- выполнить подстановку результатов запроса в строки таблицы:
окно редактора Notepad++ | вкл. Index.php | поставить курсор
перед второй строкой с текстом «Значение кода» | заменить текст «Значение кода» на:

```
<?php echo mysql_result($query,$i,stud_id);?>
```

аналогичным образом заменить фрагменты текста «Значение зачетки»,
«Значение ФИО» и «Значение группы» на фрагменты кода:

```
<?php echo mysql_result($query,$i,no_zk);?>
<?php echo mysql_result($query,$i,FIO);?>
<?php echo mysql_result($query,$i,Gruppa);?>
```

- сохранить файл index.php.
- проверить работоспособность сценария, обновив в браузере страницу index.php.
- добавить инструкции фильтрации данных по номеру зачетной книжки:
  - добавить условие, проверяющее, был ли запрос на фильтрацию:
  окно редактора Notepad++ | вкл. Index.php | поставить курсор
  перед строкой с текстом «sql = "SELECT * FROM `students_info`";» |
  ввести код:

```
if(!($_POST['zk']) or $_POST['zk']=='')
{
```

  - закрыть условный блок (если запроса не было, то будут показаны все записи):
  окно редактора Notepad++ | вкл. Index.php | поставить курсор
  после строки с текстом «$count = mysql_num_rows($query);» | ввести «}»
  - добавить код, выполняющий запрос на фильтрацию:
  окно редактора Notepad++ | вкл. Index.php | поставить курсор
  после строки с текстом «};» | ввести код

```
else
```

```
{
    $sql = "SELECT * FROM `students_info` where
`no_zk`='".$_POST['zk']."'";
    $query = mysql_query($sql);
    $count = mysql_num_rows($query);
}
```

- сохранить файл index.php.
- проверить работоспособность сценария, обновив в браузере страницу index.php.

**5. Контрольные вопросы**

1. Что представляет собой СУБД MySQL?
2. Каковы преимущества СУБД MySQL?
3. Как реализуются триггеры в MySQL?
4. Что представляют собой хранимые процедуры в MySQL?
5. Как реализуется взаимодействие MySQL и PHP?