Министерство образования и науки Самарской области

государственное автономное профессиональное

образовательное учреждение Самарской области

"Жигулевский государственный колледж"

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ**

**ПРАКТИЧЕСКИХ ЗАНЯТИЙ**

По дисциплине Основы проектирования баз данных

Курс 2

Для специальности 09.02.07 Информационные системы и программирование

Жигулевск, 2021г.

|  |  |
| --- | --- |
| РАССМОТРЕНО  на заседании предметной (цикловой) комиссииЭВЭТП  Протокол № 1 от 31.08.2021 г.  Председатель Л.В. Форсюк | Составлено в соответствии с требованиями ФГОС СПО к реализации программы подготовки специалистов среднего звена по специальности09.02.07 Информационные системы и программирование |
|  | УТВЕРЖДЕНО  на заседании научно-методического  совета  Протокол № 1 от 31.08.2021 г.  Председатель НМС М.Н. Тусинова |

Составитель: Ханмурзина Е.В., преподаватель ГАПОУ СО «ЖГК»

Эксперт: Скворцова Н.А., преподаватель ГАПОУ СО «ЖГК»

**Введение**

Методические указания для выполнения практическихзанятийявляются частью программы подготовки специалистов среднего звена ГАПОУ СО «Жигулевский государственный колледж» по специальности СПО 09.02.07 Информационные системы и программирование в соответствии с требованиями ФГОС СПО.

Методические указания по выполнению практическихзанятийадресованы студентам очнойформы обучения.

Методические указания включают в себя учебную цель, вопросы для закрепления теоретического материала, задания для работы студентов и инструкцию по ее выполнению.

Практические занятия проходят в форме практических работ.

Отчет о проделанной работе студенты выполняют в тетради для практических работ по дисциплине.

В результате освоения учебной дисциплины обучающийся должен уметь:

- проектировать реляционную базу данных;

- использовать язык запросов для программного извлечения сведений из баз данных.

В результате освоения учебной дисциплины обучающийся должен знать:

- основы теории баз данных;

- модели данных;

- особенности реляционной модели и проектирование баз данных;

- изобразительные средства, используемые в ER- моделировании;

- основы реляционной алгебры;

- принципы проектирования баз данных;

- обеспечение непротиворечивости и целостности данных;

- средства проектирования структур баз данных;

- язык запросов SQL.

В результате освоения учебной дисциплины должны формироваться общие компетенции (ОК):

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК 09. Использовать информационные технологии в профессиональной деятельности.

ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языке.

В результате освоения учебной дисциплины должны формироваться профессиональные компетенции (ПК):

ПК 5.4. Производить разработку модулей информационной системы в соответствии с техническим заданием.

Результат выполнения практической работы оценивается по пятибалльной шкале (табл. 1).

**КРИТЕРИИ ОЦЕНКИ ВЫПОЛНЕНИЯ СТУДЕНТАМИ ПРАКТИЧЕСКИХ РАБОТ**

**Таблица 1 – Оценивание практических работ студентов**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Оцениваемые навыки | Метод оценки | Критерии оценки | | | |
| Отлично | Хорошо | Удовлетворительно | Неудовлетвори  тельно |
| 1. | Отношение к работе | Наблюдение преподавателя | Внимательность при изучении методических рекомендаций | Не достаточно внимательно изучает ход работы | Не достаточно внимательно изучает ход работы | Не изучает методические рекомендации |
| 2. | Способность применять приемы работы в программе для выполнения задания.  Способность самостоятельно выполнять работу | Просмотр на компьютере выполненных заданий | Полное выполнение работы в назначенное время. Соответствие выполненной работы требованиям методических рекомендаций | Допускает одну ошибку (неточность) при выполнении работы | Допускает две, три ошибки при выполнении работы | Допускает более трех ошибок при выполнении работы |
| 3. | Умение отвечать на вопросы, пользоваться профессиональной лексикой | Собеседование (защита) при сдаче работы | Грамотно отвечает на поставленные вопросы. | Допускает незначительные ошибки в изложении приемов обработки изображений | Допускает ошибки в изложении приемов обработки изображений. Имеет ограничен  ный словарный запас. | Не отвечает на поставленные вопросы. |

ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ РАБОТ

|  |  |
| --- | --- |
| Практическая работа 1. Нормализация реляционной БД, освоение принципов проектирования БД | 7 |
| Практическая работа 2. Преобразование реляционной БД в сущности и связи | 13 |
| Практическая работа 3. Проектирование реляционной БД. Нормализация таблиц | 17 |
| Практическая работа 4. Задание ключей. Создание основных объектов БД. | 22 |
| Практическая работа 5. Создание проекта БД. Создание БД. Редактирование и модификация таблиц | 28 |
| Практическая работа 6. Редактирование, добавление и удаление записей в таблице. Применение логических условий к записям. Открытие, редактирование и пополнение табличного файла | 33 |
| Практическая работа 7. Создание ключевых полей. Задание индексов. Установление и удаление связей между таблицами | 37 |
| Практическая работа 8. Проведение сортировки и фильтрации данных. Поиск данных по одному и нескольким полям. Поиск данных в таблице | 42 |
| Практическая работа 9. Работа с переменными. Написание программного файла и работа с табличными файлами. Заполнение массива из табличного файла. Заполнение табличного файла из массива. | 47 |
| Практическая работа 10. Создание меню различных видов. Модификация и управление меню. | 51 |
| Практическая работа 11. Создание рабочих и системных окон. Добавление элементов управления рабочим окном. | 55 |
| Практическая работа 12. Создание файла проекта базы данных. Создание интерфейса входной формы. Использование исполняемого файла проекта БД, приемы создания и управления. | 60 |
| Практическая работа 13. Создание формы. Управление внешним видом формы. | 65 |
| Практическая работа 14. Задание значений и ограничений поля. Проверка введенного в поле значения. Отображение данных числового типа и типа дата. | 68 |
| Практическая работа 15. Создание и модификация таблиц БД. Выборка данных из БД. Модификация содержимого БД. | 72 |

## Практическая работа 1. Нормализация реляционной БД, освоение принципов проектирования БД

**Порядок выполнения работы**

1. Для указанного варианта задания выполнить нормализацию отношений методом декомпозиции с учетом выделенных функциональных зависимостей.
2. Представить результат в виде *нормализованной реляционноймодели*.
3. Оформить отчет по работе.

**Содержание отчета**

1. Титульный лист.
2. Реляционная таблица для заданного варианта.
3. Описание первичных ключей и функциональных зависимостей для заданного отношения (таблицы).
4. Нормализованная реляционная модель до уровня 3НФ с описанием всех этапов проведения анализа исходной таблицы и выполняемых декомпозиций. Для каждой полученной в результате декомпозиции таблицы должны быть описаны все функциональные зависимости.
5. Вывод по результатам работы.

**Теоретическиесведения**

*Нормализация* – метод создания набора отношений с заданными свойствами на основе некоторых требований к данным. Процесс нормализации – формальный метод для оптимизации столбцов отношений и устранения аномалий.

**Избыточность данных и аномалииобновления**

Основная цель проектирования реляционной БД – группирование атрибутов в отношениях таким образом, чтобы минимизировать избыточность данных (сокращение объема вторичной памяти для хранения БД) и повышение надежности при работе с данными.

Обычно процесс проектирования отношений реляционной БД ведется на основе разработанной ER-диаграммы или на основе просто здравого смысла разработчика. В общем случае при таком подходе расположение атрибутов в отношениях *неоптимальное*. При работе с отношениями, содержащими избыточные данные, могут возникать проблемы – *аномалии обновления*.

Аномалии обновления делят на три вида:

* + - * *аномалии вставки* – возникают при добавлении новых несогласованных данных (нарушающих целостность данных вотношении);
      * *аномалии изменения* – возникают при изменении части ранее введенных данных; частичное обновление сведений приведет к нарушению целостности данныхотношения;
      * *аномалии удаления* – возникают при удалении строк изотношений.

Обычно для решения проблем избыточности и аномалий выполняется деление отношения на такие отношения, в которых избыточности не будет. Для выполнения такого процесса необходимо выявить все зависимости между атрибутами отношения (потеря одной такой зависимости меняет модель внешнего мира).

**Функциональныезависимости**

Выявление смысловой зависимости между данными – один из способов формализации смысловой информации о данных.

*Функциональная зависимость* описывает связь типа «многие-к-одному» между атрибутами отношения, где «много» – детерминант функциональной зависимости. Функциональная зависимость является семантическим свойством атрибутов отношения.

Если в отношении *R*, содержащем атрибуты *A* и *B*, атрибут*B* функционально зависит от атрибута *A* (*А* является детерминантом атрибута *B*) *A* *B*, то в каждом кортеже этого отношения каждое конкретное значение атрибута *A* всегда связано только с одним значением атрибута *B*.

Особенности функциональных зависимостей, лежащие в основе процесса нормализации:

* + - * функциональная зависимость является специализированным правилом целостности – она накладывает ограничения на допустимые значения атрибутов отношений; эту особенность можно использовать при обновлении БД, т.к. зная, какие функциональные зависимости есть в отношении, можно понять, нарушат ли новые данные целостность данныхотношения;
      * функциональная зависимость является обобщением понятия потенциального ключа; функциональные зависимости позволяют определить все потенциальные ключи отношения (и соответственно – первичный ключ): все атрибуты отношения, которые не являются частью первичного (или потенциального) ключа, должны функционально зависеть от этого ключа; если не все остальные атрибуты отношения зависят от некоторого детерминанта, то этот детерминант не является потенциальным ключом этогоотношения.

**Нормальные формы и нормализация методомдекомпозиции**

*Нормализация* – это формальный метод анализа отношений на основе их первичного ключа и существующих функциональных зависимостей.

Суть процесса нормализации:

* + - * в нормализованных отношениях не разрешаются никакие функциональные зависимости, кроме функциональных зависимостей вида *K* *A*, где *K* – потенциальный ключ отношения *R*, а *A* – неключевой атрибут;
      * еслижеотношение*R*имеетфункциональныезависимости *B* *A* , где B не является потенциальным ключом, то в отношении *R* будет наблюдаться избыточностьданных.

В процессе нормализации реляционных отношений применяются концепции *нормальных форм*. Говорят, что отношение находится в определенной нормальной форме, если оно удовлетворяет правилам этой нормальной формы. В настоящее время используется шесть нормальных форм, которые зависят друг от друга путем усложнения (вложенности) набора правил:

1*НФ*  2*НФ*  3*НФ* *НФБК*  4*НФ*  5*НФ* .

Каждая нормальная форма, таким образом, **удовлетворяет всем предыдущим нормальным формам**. Более высокая нормальная форма приводит к более строгому формату отношения (меньшее число аномалий обновления).

**Примечание.** БД можно построить и на отношениях, находящихся в первой нормальной форме, но такая БД будет сильно подвержена аномалиям и избыточности данных.

**На практике желательно использовать, как минимум, *3НФ***, чтобы устранить большинство аномалий обновления.

1. *1НФ*. Отношение находится в *1НФ* тогда и только тогда, когда в любом допустимом значении этого отношения каждый кортеж содержит только одно значение для каждого из атрибутов, т.е. это значение не имеет внутренней структуры (множество, таблица и т.п.). Отношения в *1НФ* имеют большое количество аномалийобновления.
2. *2НФ*. Отношение находится в *2НФ* тогда и только тогда, когда оно находится в *1НФ*, и каждый атрибут отношения, не входящий в состав первичного ключа, характеризуется полной функциональной зависимостью от этого первичного ключа.

**Полной функциональной зависимостью**называется такая зависимость *A* *B* , когда *B* функционально зависит от *A* и не зависит ни от какого подмножества *A* (т.е. удаление какого-либо атрибута из *A* приведет к утрате этой функциональной зависимости). *2НФ* устраняет в отношении частичные функциональные зависимости неключевых атрибутов от первичного ключа, которые выносятся в отдельное отношение вместе с копиями своих детерминантов (частей первичного ключа, от которого они зависят).

1. *3НФ*. Отношение находится в *3НФ* тогда и только тогда, когда оно находится в *2НФ* и не имеет не входящих в первичный ключ атрибутов, которые находились бы в транзитивной функциональной зависимости от этого первичного ключа.

Транзитивной функциональной зависимостьюназывается зависимость *A* *C* , если существуют зависимости *A* *B* и *B* *C*(говорят, что атрибут *C* транзитивно зависит от *A* через атрибут *B*), при условии, что атрибут A функционально не зависит ни от атрибута B, ни от атрибута *C*.

*3НФ* устраняет в отношении транзитивные функциональные зависимости неключевых атрибутов от первичного ключа, которые выносятся в отдельное отношение вместе с копиями своих детерминантов. В *3НФ* устранено большинство аномалий от первичного ключа, но отношение в этой форме имеет аномалии в случае наличия более чем одного потенциального ключа.

*Декомпозиция* – формирование отношений БД путем разделения их на более мелкие, если эти отношения не выполняют правила необходимой нормальной формы.

**Рекомендации по выполнениюработы**

*Этап* 1. Выделить функциональные зависимости для каждого отношения исходной реляционной схемы. Проверить практический смысл выделенных функциональных зависимостей.

*Этап* 2. Для каждого отношения (включая и вновь создаваемые) последовательно применить правила нормальных форм. При несоблюдении текущего правила в отношении выполнить его декомпозицию (удалить проблемный атрибут из отношения с образованием нового отношения, первичным ключом которого будет детерминант рассматриваемой функциональной зависимости (этот атрибут только копируется в новое отношение)). Нормализованное отношение должно удовлетворять как минимум *3НФ*.

*Этап* 3. Для полученной нормализованной реляционной схемы проверить смысл ссылок.

*Этап* 4. Реализовать полученные реляционные отношения в виде таблиц в среде целевойСУБД.

*Этап* 5. Оформить отчет по работе.

**Пример приведения отношения к 3НФ**

Рассмотрим отношение «Экзаменационная ведомость»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код студента | Фамилия | Код экзамена | Предмет и дата | Оценка |
| 1 | Иванов | 1 | Математика, 05.06.2019 | 4 |
| 2 | Петров | 1 | Математика, 05.06.2019 | 5 |
| 1 | Иванов | 2 | Физика, 10.06.2019 | 5 |
| 2 | Петров | 2 | Физика, 10.06.2019 | 5 |

Первичный ключ таблицы состоит из атрибутов: Код студента, Код экзамена

**Отношение находится в первой нормальной форме (1НФ), если все атрибуты отношения принимают простые значения (атомарные или неделимые), не являющиеся множеством или кортежем из более элементарных составляющих.**

Наше отношение не находится в 1НФ.

Приведем отношение к 1НФ:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Код студента | Фамилия | Код экзамена | Предмет | Дата | Оценка |
| 1 | Иванов | 1 | Математика | 05.06.2019 | 4 |
| 2 | Петров | 1 | Математика | 05.06.2019 | 5 |
| 1 | Иванов | 2 | Физика | 10.06.2019 | 5 |
| 2 | Петров | 2 | Физика | 10.06.2019 | 5 |

Для исследования наличия 2НФ следует проанализировать функциональные зависимости между атрибутами отношения.

Единственный способ определить функциональные зависимости – внимательно проанализировать семантику (смысл) атрибутов.

Примеры функциональных зависимостей для отношения ЭКЗАМЕНАЦИОННАЯ ВЕДОМОСТЬ:

Код студента → Фамилия

Код студента, Код экзамена → Оценка

Код экзамена → Дата

Код экзамена → Предмет

**Отношение находится в 2НФ, если оно находится в 1НФ и каждый неключевой атрибут зависит от всего первичного ключа (не зависит от части ключа).**

**Отношение находится в 3НФ, если оно находится в 2НФ и каждый ключевой атрибут нетранзитивно зависит от первичного ключа. Отношение находится в 3НФ в том и только том случае, если все неключевые атрибуты отношения взаимно независимы и полностью зависят от первичного ключа.**

Продолжим рассмотрение примера с отношением ЭКЗАМЕНАЦИОННАЯ ВЕДОМОСТЬ. Для более краткой записи процесса нормализации введем следующие обозначения: КС – код студента, КЭ – код экзамена, Ф – фамилия, П – предмет, Д – дата, О – оценка.

Наше отношение примет вид: R=( **КС, КЭ**, Ф, П, Д, О )

Выпишем функциональные зависимости:

КС, КЭ → Ф, П, Д, О (КС, КЭ - первичный ключ отношения, все неключевые атрибуты зависят от первичного ключа)

При этом некоторые атрибуты зависят не от всего ключа в целом:

КЭ → П

КЭ → Д (предмет и дата зависят только от кода экзамена)

КС → Ф (фамилия студента зависит только от кода студента)

В соответствии с определением, **отношение находится во второй нормальной форме (2НФ), если оно находится в 1НФ и каждый неключевой атрибут зависит от первичного ключа и не зависит от части ключа**. Здесь атрибуты П, Д, Ф зависят от части ключа. Чтобы избавиться от этих зависимостей необходимо произвести декомпозицию отношения.

Выделим неполные зависимости в отдельные отношения. Если какие-то атрибуты зависят от одной части ключа, объединяем их в одну таблицу.

Получим отношение R1(**КС**, Ф) - это отношение находится в 2 НФ, так как ключ отношения простой и частичной зависимости быть не может. Так как в этом отношении нет транзитивных зависимостей, отношение R1(КС, Ф) находится в 3НФ.

Второе отношение R2(**КЭ**, П, Д) - зависимости неключевых атрибутов от части ключа нет, следовательно отношение находится в 2НФ. Транзитивных зависимостей в этом отношении также нет, следовательно отношение находится в 3НФ.

Исходное отношение приведено к виду: R(**КС, КЭ**, О). Из него выведены неключевые атрибуты, зависящие от части ключа. Неключевой атрибут О зависит от ключа КС, КЭ в целом, а не от его части. Значит, это отношение находится в 2НФ. Транзитивные зависимости отсутствуют, то есть отношение находится в 3НФ.

Таким образом все полученные отношения находятся в 3НФ.

Между таблицами установлены связи, как показано на рис. 1.

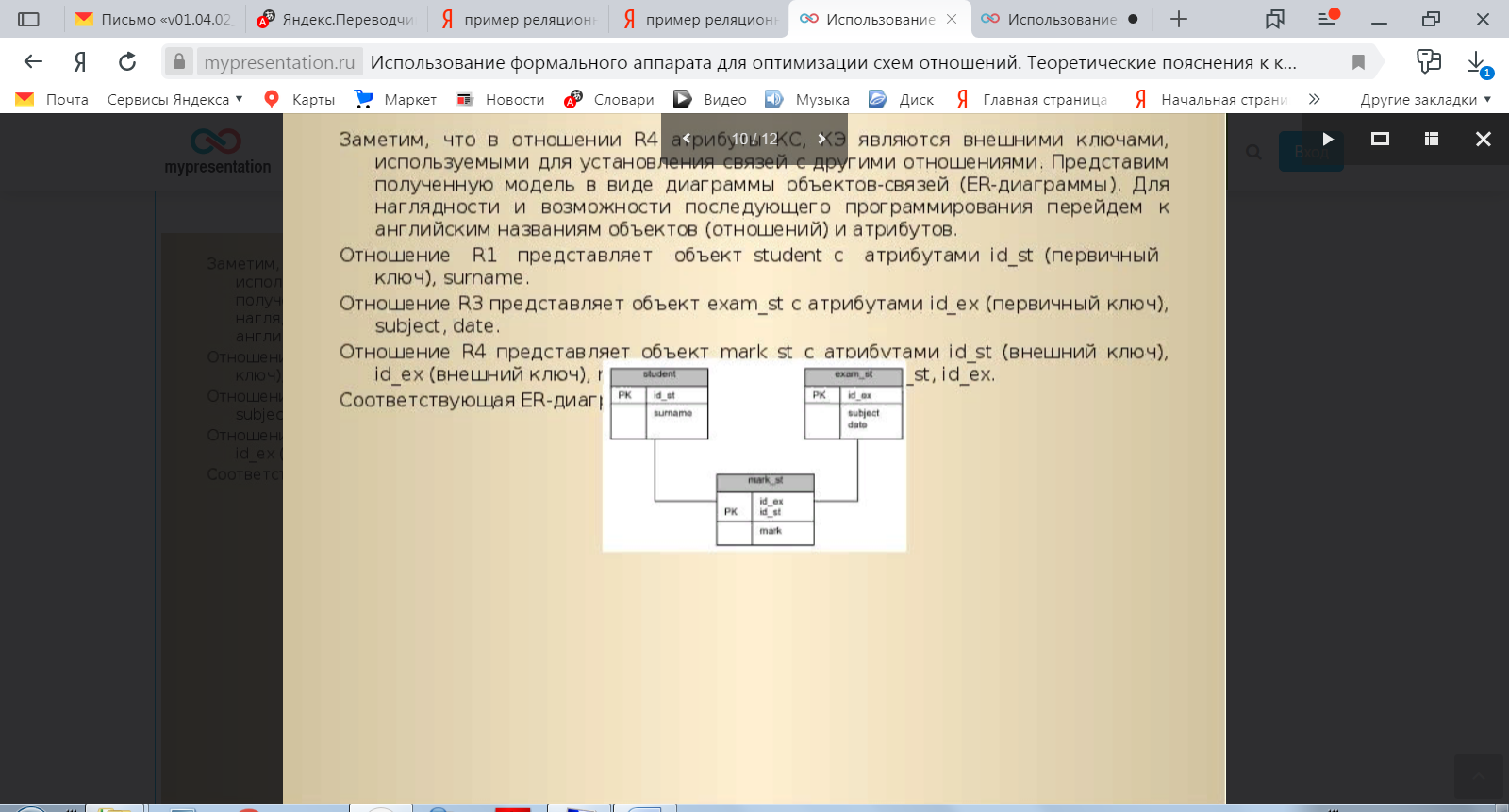


Рис. 1. Полученная реляционная модель (нормализованная)

Использованы следующие обозначения:

id\_st – код студента;

surname – фамилия;

id\_ex – код экзамена;

subject – предмет;

date – дата;

mark – оценка.

**Варианты заданий**

*Вариант 1 – отношение «Морские перевозки»*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер судна | Название | Номер рейса | Дата погрузки | Порт погрузки | Дата прибытия | Порт прибытия | Ф.И.О.  капитана | Вид судна | Грузо  подъем  ность, тонны |
| 526 | Japan Bear | 9201W | 5/31/92 | SFO | 6/6/92 | HNL | Емелин А.О. | Сухогруз | 500 |
| 603 | Korea Bear | 9202W | 5/05/92 | OAK | 6/19/92 | OSA | Крылов О.Б. | Ролкер | 1000 |
| 531 | China Bear | 9203W | 6/20/92 | LAX | 7/10/92 | PAP | Мухин Е.А. | Универсал | 1500 |
| 526 | Japan Bear | 9204W | 8/20/92 | SFO | 8/27/92 | HNL | Емелин А.О. | Сухогруз | 500 |

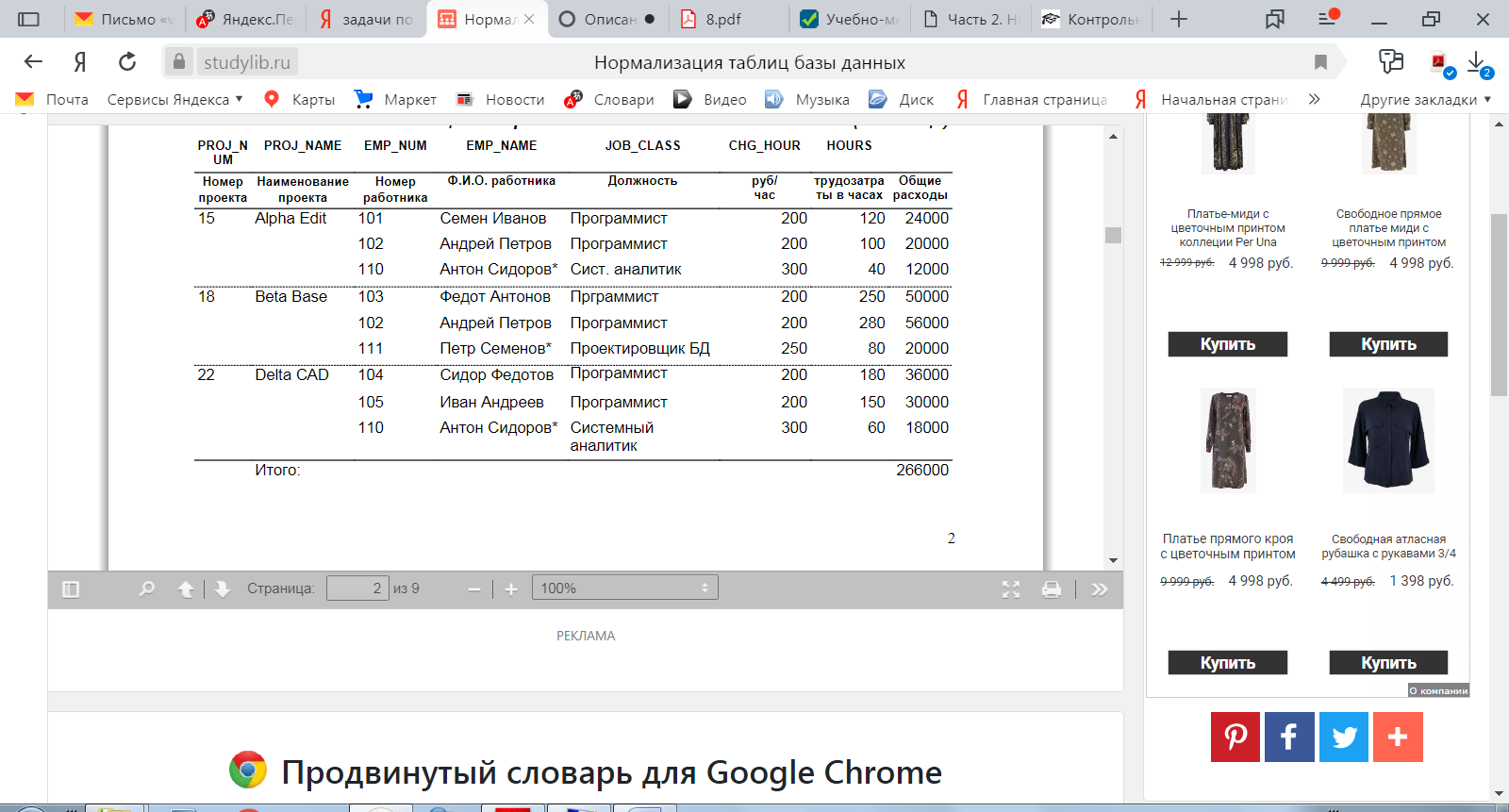
*Вариант 2 – отношение «Контрагенты»*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Наименование контрагента | Город | Адрес | Вид контрагента | Должность контактного лица | Ф.И.О.  контактного лица | Код города | Телефон |
| Поршневой завод | Владимир | ул. Кольцевая, 17 | Поставщик | зам. дир. | Иванов И.И. | 3254 | 76-15-95 |
| Поршневой завод | Владимир | ул. Кольцевая, 17 | Поставщик | нач. отд. сбыта | Петров П.П. | 3254 | 76-15-35 |
| ООО «Вымпел» | Курск | ул. Гоголя, 25 | Клиент, Поставщик | директор | Сидоров С.С. | 7634 | 66-65-38 |
| ИП «Альфа» | Владимир | ул.Пушкинская, 37 | Клиент, Поставщик | директор | Васильев В.В. | 3254 | 74-57-45 |

*Вариант 3 – отношение «Отдел кадров»*

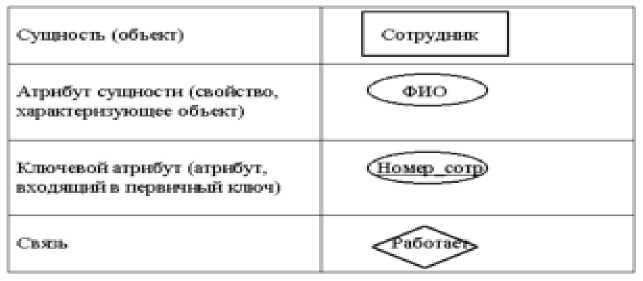
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Код сотрудника | ФИО | Должность | Номер отдела | Наименование отдела | Квалификация |
| 7513 | Иванов И.И | Программист | 120 | Отдел проектирования | С, Java |
| 9842 | Петров А.А. | Администратор БД | 30 | Финансовый отдел | MS SQL Server |
| 6651 | Сорокин А.П. | Прогрсммист | 120 | Отдел проектирования | VB, Java |
| 9006 | Ворнов Г.Р. | Системный администраторо | 120 | Отдел проектирования | Windows, Linux |

*Вариант 4 – отношение «Ведомость расходов»*



**Практическая работа 2. Преобразование реляционной БД в сущности и связи**

**Цель работы:** выработать практические навыки моделирования предметной области и построении ER-модели данных, закрепить технологию проектирования БД, закрепить основные понятия теории реляционных баз данных, освоить технологию построения ER-диаграмм, научиться получать реляционные БД из ER-диаграмм Сущность-связь



**Теоретические сведения**

Работа с базой данных начинается с построения модели. Наиболее распространенной является ER-модель (entity-relationshipmodel) - модель "Сущность-связь". Для "ручного" построения ER-модели на практике будем использовать простую систему обозначений, предложенную Питером Ченом (обозначения, встречающиеся в разных источниках, могут отличаться от нижеприведенных):

Первичный ключ - атрибут или группа атрибутов, однозначно идентифицирующих объект. Первичный ключ может состоять из нескольких атрибутов, тогда подчеркивается каждый из них.

Связи между объектами могут быть 3-х типов: Один - к одному. Этот тип связи означает, что каждому объекту первого вида соответствует не более одного объекта второго вида, и наоборот. Например: сотрудник может руководить только одним отделом, и у каждого отдела есть только один руководитель. Один - ко многим. Этот тип связи означает, что каждому объекту первого вида может соответствовать более одного объекта второго вида, но каждому объекту второго вида соответствует не более одного объекта первого вида. Например: в каждом отделе может быть множество сотрудников, но каждый сотрудник работает только в одном отделе. Многие - ко многим. Этот тип связи означает, что каждому объекту первого вида может соответствовать более одного объекта второго вида, и наоборот. Например: каждый счет может включать множество товаров, и каждый товар может входить в разные счета.

Реляционная структура данных

В конце 60-х годов появились работы, в которых обсуждались возможности применения различных табличных даталогических моделей данных. Наиболее значительной из них была статья сотрудника фирмы IBMд-ра Эдварда Кодда (CoddE.F., ARelationalModelofDataforLargeSharedDataBanks. CACM 13: 6, June 1970), где впервые был применен термин «реляционная модель данных».

Будучи математиком по образованию, Э. Кодд предложил использовать для обработки данных аппарат теории множеств (объединение, пересечение, разность, декартово произведение). Он показал, что любое представление данных сводится к совокупности двумерных таблиц особого вида, известного в математике как отношение - relation.

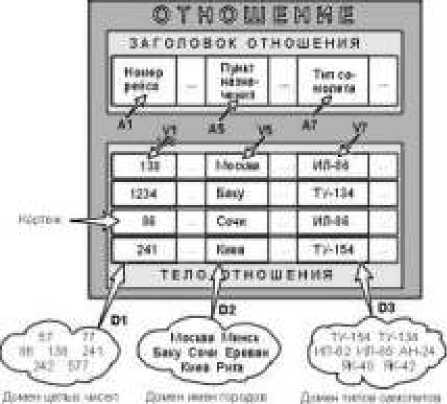
Наименьшая единица данных реляционной модели - это отдельное атомарное (неразложимое) для данной модели значение данных.Так, в одной предметной области фамилия, имя и отчество могут рассматриваться как единое значение, а в другой - как три различных значения.

Доменом называется множество атомарных значений одного и того же типа. Так, на рис. домен пунктов отправления (назначения) - множество названий населенных пунктов, а домен номеров рейса - множество целых положительных чисел



Смысл доменов состоит в следующем. Если значения двух атрибутов берутся из одного и того же домена, то, вероятно, имеют смысл сравнения, использующие эти два атрибута (например, для организации транзитного рейса можно дать запрос «Выдать рейсы, в которых время вылета из Москвы в Сочи больше времени прибытия из Архангельска в Москву»). Если же значения двух атрибутов берутся из различных доменов, то их сравнение, вероятно, лишено смысла: стоит ли сравнивать номер рейса со стоимостью билета?

Отношение на доменах D1, D2, ..., Dnсостоит из заголовка и тела. На рис. 3.4 приведен пример отношения для расписания движения самолетов (таблица 1). Ai - атрибуты, Vi - значения атрибутов.



6

Рисунок 1.2. Отношение с математической точки зрения

Заголовок отношения состоит из такого фиксированного множества атрибутов A1, A2, ..., An, что существует взаимно однозначное соответствие между этими атрибутами Ai и определяющими их доменами Di (i=1,2,...,n).

Тело отношения состоит из меняющегося во времени множества кортежей, где каждый кортеж состоит в свою очередь из множества пар атрибут-значение (Ai:Vi), (i=1,2,...,n), по одной такой паре для каждого атрибута Ai в заголовке.

Для любой заданной пары атрибут-значение (Ai:Vi) Vi является значением из единственного домена Di, который связан с атрибутом Ai.

Отношение степени один называют унарным, степени два - бинарным, степени три - тернарным, ..., а степени n - n-арным. Степень отношения «Рейс» (таблица 1) равна 8.

*Кардинальное число или мощность отношения - это число его кортежей.*

Мощность отношения «Рейс» равна 10. Кардинальное число отношения изменяется во времени в отличие от его степени.

Поскольку отношение - это множество, а множества по определению не содержат совпадающих элементов, то никакие два кортежа отношения не могут быть дубликатами друг друга в любой произвольно заданный момент времени.

*Пусть R- отношение с атрибутами A1, A2, An. Говорят, что множество атрибутов K=(Ai, Aj, ..., Ak) отношения Rявляется возможным ключом Rтогда и только тогда, когда удовлетворяются два независимых от времени условия:*

Уникальность: в произвольный заданный момент времени никакие два различных кортежа R не имеют одного и того же значения для Ai, Aj, ..., Ak.

Минимальность: ни один из атрибутов Ai, Aj, ..., Ak не может быть исключен из K без нарушения уникальности.

Каждое отношение обладает хотя бы одним возможным ключом, поскольку по меньшей мере комбинация всех его атрибутов удовлетворяет условию уникальности. Один из возможных ключей (выбранный произвольным образом) принимается за его первичный ключ. Остальные возможные ключи, если они есть, называются альтернативными ключами.

Вышеупомянутые и некоторые другие математические понятия явились теоретической базой для создания реляционных СУБД, разработки соответствующих языковых средств и программных систем, обеспечивающих их высокую производительность, и создания основ теории проектирования баз данных.

Также на практике широко используются неформальные эквиваленты этих понятий: Отношение - Таблица, Кортеж - Строка таблицы или Запись, Атрибут - Столбец Таблицы или Поле.

При этом принимается, что «запись» означает «экземпляр записи», а «поле» означает «имя и тип поля».

**Задание для практической работы**

**Задание 1. Проектирование реляционных Баз Данных.**

Вариант 1.

Построить реляционную таблицу Базы Данных имен родственников студентов вашей группы, содержащую данные об именах родителей, братьев и сестер студентов.

Вариант 2.

Построить реляционную таблицу Базы Данных домов, где живут студенты вашей группы, содержащую данные о районе расположения дома, количестве этажей в нем и номере этажа, где живет студент.

Вариант 3.

Построить реляционную таблицу Базы Данных дней рождения студентов вашей группы, содержащую данные о дате рождения, знаке зодиака и годе по Китайскому календарю.

Вариант 4.

Построить реляционную таблицу Базы Данных Сотовых телефонов студентов вашей группы, содержащую данные о модели телефона, типе корпуса, операторе.

**Технология выполнения работы и оформление отчета**

1. Придумайте заголовок отношения и запишите его в отчет.
2. Определите атрибуты отношения. Начертите сетку таблицы в отчет и занесите в нее атрибуты.
3. Опросите студентов вашей группы и занесите полученные данные в таблицу.

На чертеже таблицы укажите чему соответствуют понятия: Заголовок отношения, тело отношения, атрибут отношения, кортеж отношения.

1. Определите и запишите в отчет степень отношения и мощность отношения.
2. Дайте определение первичного ключа. Укажите Первичный ключ получившегося отношения
3. Докажите, что у вас получилась реляционная таблица, для этого укажите типы данных всех атрибутов.

**Задание 2. Проектирование Баз Данных. ER-диаграммы.**

Формулировка задания. По описанию предметной области построить логическую модель БД методом ER-диаграмм, на основании которой построить набор таблиц БД.

Вариант 1.

Описание предметной области (Ресторан).

Посетители ресторана обслуживаются за столиками. За одним столом может располагаться не более 4 посетителей, каждый из которых может сделать заказ тех или иных блюд. Столики обслуживают официанты. У одного официанта в обслуживании несколько столов.

Задачи для БД:

* Есть ли свободные столы?
* Сколько посетителей обслужил официант за смену?
* Сколько каких блюд было реализовано?

Вариант 2.

Описание предметной области (Колледж).

Студенты колледжа объединены в группы. Набор дисциплин, изучаемых студентом, зависит от номера группы в которой он учится. Преподаватели читают дисциплины и выставляют зачеты студентам. Один преподаватель может читать несколько дисциплин, но каждую дисциплину ведет один преподаватель.

Задачи для БД:

* Какие дисциплины изучает студент?
* Какая оценка у студента по данной дисциплине?
* Кто выставил эту оценку?

Вариант3.

Описание предметной области (Театральная касса).

В театральной кассе продаются билеты на спектакли. Стоимость билета зависит от ряда, театра и спектакля. Каждый день в театре может идти не более одного спектакля. Спектакль характеризуется названием и автором. Каждый покупатель может купить сколько угодно билетов на любые спектакли.

Задачи для БД:

* Какие спектакли идут в определенный день?
* Есть ли билеты на конкретный спектакль?
* Сколько стоит конкретный билет?

Вариант 4.

Описание предметной области (Грузоперевозки).

АТП имеет грузовые автомобили с гос. номерами и организует перевозки для своих заказчиков. Стоимость перевозки зависит от расстояния и грузоподъемности автомобиля, который ее выполняет. Каждый заказчик может сделать заказ нескольких перевозок. Одну перевозку выполняет один грузовик.

Задачи для БД:

* Какие грузовики свободны?
* Какой заказчик сделал самый дорогой заказ?
* Какой грузовик выполнил наибольшее количество заказов?

**Технология выполнения работы**

Построение ER-диаграммы

* 1. Выберите из описания предметной области все существительные. Продумайте, какие из них будут соответствовать сущностям, а какие атрибутам сущностей. Зарисуйте в отчет все сущности с их атрибутами согласно обозначениям, принятым в ER-диаграммах.
  2. На рисунке подчеркиванием атрибутов обозначьте для каждой сущности уникальный идентификатор (Ключ). При необходимости добавьте сущностям атрибуты, которые помогут однозначно отличить каждый экземпляр сущности.
  3. Определите и включите в схему связи сущностей. Подпишите названия связей и пронумеруйте связи. Для первой связи укажите тип и модальность. Для всех связей запишите их прочтение слева направо и справа налево.
  4. Если в схеме присутствуют связи типа «много-со-многими» уберите их путем ввода дополнительной сущности. Измененную схему зарисуйте в отчет.

1. Получение реляционной схемы из ER-диаграммы.
   1. Каждая сущность превращается в таблицу. Имя сущности - имя таблицы. Набор всех таблиц - БД. Вспомните, что такое схема БД. Запишите схему вашей БД в отчет.
   2. Зарисуйте все полученные таблицы с их заголовками и названиями столбцов. Выделите потенциальные и внешние ключи (если есть) для каждой таблицы. Укажите столбцы, допускающие неопределенные значения.

Докажите, что полученные отношения находятся в Первой нормальной форме

**Практическая работа 3. Проектирование реляционной БД. Нормализация таблиц**

**Цель:**Научиться проектировать базы данных и приводить таблицы БД к требуемой нормальной форме

**Задание:**

1. Изучить способы нормализации таблиц.
2. Спроектировать базу данных в соответствии с индивидуальным заданием
3. Составить описание полей всех таблиц
4. Создать логическую модель базы данных
5. Описать ход работы.
6. Оформить отчет

**Теоретические сведения:**

**Нормализация таблиц**

Нормализация таблицы представляет собой последовательное изменение структуры таблицы до тех пор, пока она не будет удовлетворять требованиям последней формы нормализации.

**Функциональной зависимостью** между полями А и В называется зависимость, при которой каждому значению А в любой момент времени соответствует единственное значение В из всех возможных. Примером функциональной зависимости может служить связь между идентификационным номером налогоплательщика и номером его паспорта.

**Полной функциональной зависимостью между составным полем А и полем В** называется зависимость, при которой поле В зависит функционально от поля А и не зависит функционально от любого подмножества поля А.

**Многозначная функциональная зависимость между полями** определяется следующим образом. Поле А многозначно определяет поле В, если для каждого значения поля А существует «хорошо определенное множество» соответствующих значений поля В. Например, если рассматривать таблицу успеваемости учащихся в школе, включающую в себя поля «Предмет» (поле А) и «Оценка» (поле В), то поле В имеет «хорошо определенное множество» допустимых значений: 1, 2, 3, 4, 5, т.е. для каждого значения поля «Предмет» существует многозначное «хорошо определенное множество» значений поля «Оценка».

**Первая нормальная форма.** Таблица находится в первой нормальной форме тогда и только тогда, когда ни одно из полей не содержит более одного значения и любое ключевое поле не пусто.

Первая нормальная форма является основой реляционной модели данных. Любая таблица в реляционной базе данных автоматически находится в первой нормальной форме, иное просто невозможно по определению. В такой таблице не должно содержаться полей (признаков), которые можно было бы разделить на несколько полей (признаков).

**Вторая нормальная форма.** Таблица находится во второй нормальной форме , если она удовлетворяет требованиям первой нормальной формы и все ее поля, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключем.

Если таблица имеет простой первичный ключ, состоящий только из одного поля, то она автоматически находится во второй нормальной форме. Если же первичный ключ составной, то таблица необязательно находится во второй нормальной форме. Тогда ее необходимо разделить на две или более таблиц таким образом, чтобы первичный ключ однозначно идентифицировал значение в любом поле.

**Третья нормальная форма**. Таблица находится в третьей нормальной форме, если она удовлетворяет определению второй нормальной формы и ни одно из ее не ключевых полей не зависит функционально от любого другого не ключевого поля.

**Типы данных в приложении MSAccess**

Для каждого поля таблиц базы данных необходимо указывать тип данных. Тип данных определяет вид и диапазон допустимых значений, которые могут быть введены в поле, а также объем памяти, выделяющийся для этого поля.

|  |  |
| --- | --- |
| **Тип данных** | **Содержимое типа данных** |
| Текстовый | Текст и числа, например, имена и адреса, номера теле­фонов и почтовые индексы. Текстовое поле может содер­жать до 255 символов |
| Поле Memo | Длинный текст и числа, например комментарии и пояснения. Поле Memo может содержать до 64 000 символов |
| Числовой | Числовые данные, допускающие проведение математи­ческих расчетов, за исключением денежных. Свойство *Размер поля* позволяет указывать различные типы числовых данных |
| Дата/время | Значения даты и времени. Пользователь имеет возмож­ность выбора одного из многочисленных стандартных форматов или создания специального формата |
| Денежный | Денежные значения (не рекомендуется использовать для проведения денежных расчетов значения, принадлежа­щие к числовому типу данных, так как последние могут округляться при расчетах), которые всегда выводятся с указанным числом десятичных знаков после запятой |
| Счетчик | Автоматически вставляющиеся последовательные номера. Нумерация начинается с единицы. Поле счетчика, удоб­ное для создания ключа, является совместимым с полем числового типа, для которого в свойстве *Размер поля* указано значение Длинное целое |
| Логический | Значения Да/Нет, Истина/Ложь, Вкл./Выкл. |
| Поле объекта OLE | Объекты, созданные в других программах, поддерживаю­щих протокол OLE, которые связываются или внедряют­ся в базу данных Microsoft Access через элемент управле­ния в форме или отчете |

**Ключи**

В каждой таблице реляционной модели должен быть столбец (или совокупность столбцов), значение которого однозначно идентифицирует каждую ее строку. Этот столбец (или совокупность столбцов) и называется **первичным ключом таблицы**.

Если таблица удовлетворяет требованию уникальности первичного ключа, она называется отношением. В реляционной модели все таблицы должны быть преобразованы в отношения. Отношения реляционной модели связаны между собой. Связи поддерживаются внешними ключами. **Внешний ключ** — это столбец (совокупность столбцов), значение которого однозначно характеризует значения первичного ключа другого отношения (таблицы).

**Логическая структура данных**

Каждая информационная система в зависимости от назначения имеет дело с той или иной частью конкретного мира, которую принято называть ее предметной областью. Анализ предметной области является необходимым начальным этапом разработки любой информационной системы. Именно на этом этапе определяются информационные потребности всей совокупности пользователей будущей системы, которые, в свою очередь, предопределяют содержание ее базы данных. Предметная область конкретной информационной системы рассматривается, прежде всего, как некоторая совокупность реальных объектов, которые представляют интерес для ее пользователей. Примерами объектов предметной области могут служить персональные ЭВМ, программные продукты и их пользователи. Каждый из этих объектов обладает определенным набором свойств (атрибутов). Так, например, компьютер характеризуется названием фирмы-производителя, идентификатором модели, типом микропроцессора, объемом оперативной и внешней памяти, типом графической карты и т.д.

Информационный объект — это описание некоторой сущности предметной области, т. е. реального объекта, процесса, явления или события. Информационный объект (сущность) образуется совокупностью логически взаимосвязанных атрибутов (свойств), представляющих собой качественные и количественные характеристики объекта (сущности).

Информационно-логическая модель (ИЛМ) — это совокупность информационных объектов (сущностей) предметной области и связей между ними.

Процесс создания информационной модели начинается с определения концептуальных требований будущих пользователей БД. Требования отдельных пользователей интегрируются в едином обобщенном представлении, которое называют концептуальной моделью данной предметной области. Такая модель отображает предметную область в виде взаимосвязанных объектов без указания способов их физического хранения.

Возможно, что отраженные в концептуальной модели взаимосвязи между объектами окажутся впоследствии нереализуемыми средствами выбранной СУБД, что потребует ее изменения. Версия концептуальной модели, которая может быть реализована конкретной СУБД, называется **логической моделью**.

Логическая модель, отражающая логические связи между атрибутами объектов вне зависимости от их содержания и среды хранения, может быть реляционной, иерархической или сетевой. Таким образом, логическая модель отображает логические связи между информационными данными в данной концептуальной модели.

**Ход работы:**

1. Описать предметную область, выбранную для индивидуального проектирования базы данных.
2. Привести таблицы к третьей нормальной форме.
3. Спроектировать базу данных, разработать таблицы, выбрать тип данных и ключевые поля. Составить подробное описание всех таблиц базы данных.
4. Составить логическую структуру базы данных.

**Пример:**

Приведем пример проектирования базы данных, содержащей информацию об успеваемости студентов.

1. Описание предметной области.

Каждый семестр учебная часть обрабатывает зачетные и экзаменационные ведомости. Каждый преподаватель в ведомости выставляет оценки студентов по соответствующей дисциплине. Все сведения о студентах являются справочными, при этом каждый студент находится в определенной группе. Целью проектирования базы данных является систематизация информации о студентах, группах, предметах и оценках по этим предметам.

2. Нормализация таблиц

В БД можно выделить 4 таблицы:

Студенты (номер студента, фамилия, имя, отчество, пол, дата рождения, адрес, телефон, номер группы)

Группы (номер группы, название группы, классный руководитель)

Предметы (код дисциплины, название, преподаватель)

Оценки (номер оценки, номер студента, код предмета, дата оценки, оценка)

Для соответствия 1НФ поле ФИО студента было разделено на три отдельных поля.

Для соответствия 2НФ проведены следующие действия: Каждое значение ключевого поля любой из таблиц однозначно определяет всю запись, т.е. все остальные не ключевые поля. Для устранения избыточности первичного ключа в некоторых таблицах было добавлено дополнительное поле (например, «номер оценки в таблице «Оценки»).

Таблицы так же соответствуют 3НФ, т.к. ни одно из не ключевых полей не зависит от других не ключевых полей.

3. Проектирование базы данных

Таблица Студенты

|  |  |  |
| --- | --- | --- |
| **Логическое имя поля** | **Тип данных** | **Свойства поля** |
| Номер студента | Числовой (счетчик) | Длина 5, индекс без повторений, первичный ключ |
| Номер группы | Числовой | Внешний ключ, список групп |
| Фамилия | Текстовый |  |
| Имя | Текстовый |  |
| Отчество | Текстовый |  |
| Пол | Текстовый | Длина 1, список «м», «ж» |
| Дата рождения | Дата/время | Короткий формат даты |
| Адрес | Текстовый |  |
| Телефон | Текстовый | Необязательное поле |

Таблица Группы

|  |  |  |
| --- | --- | --- |
| **Логическое имя поля** | **Тип данных** | **Свойства поля** |
| Номер группы | Числовой (счетчик) | Длина 5, индекс без повторений, первичный ключ |
| Название | Текстовый |  |
| Классный руководитель | Текстовый |  |

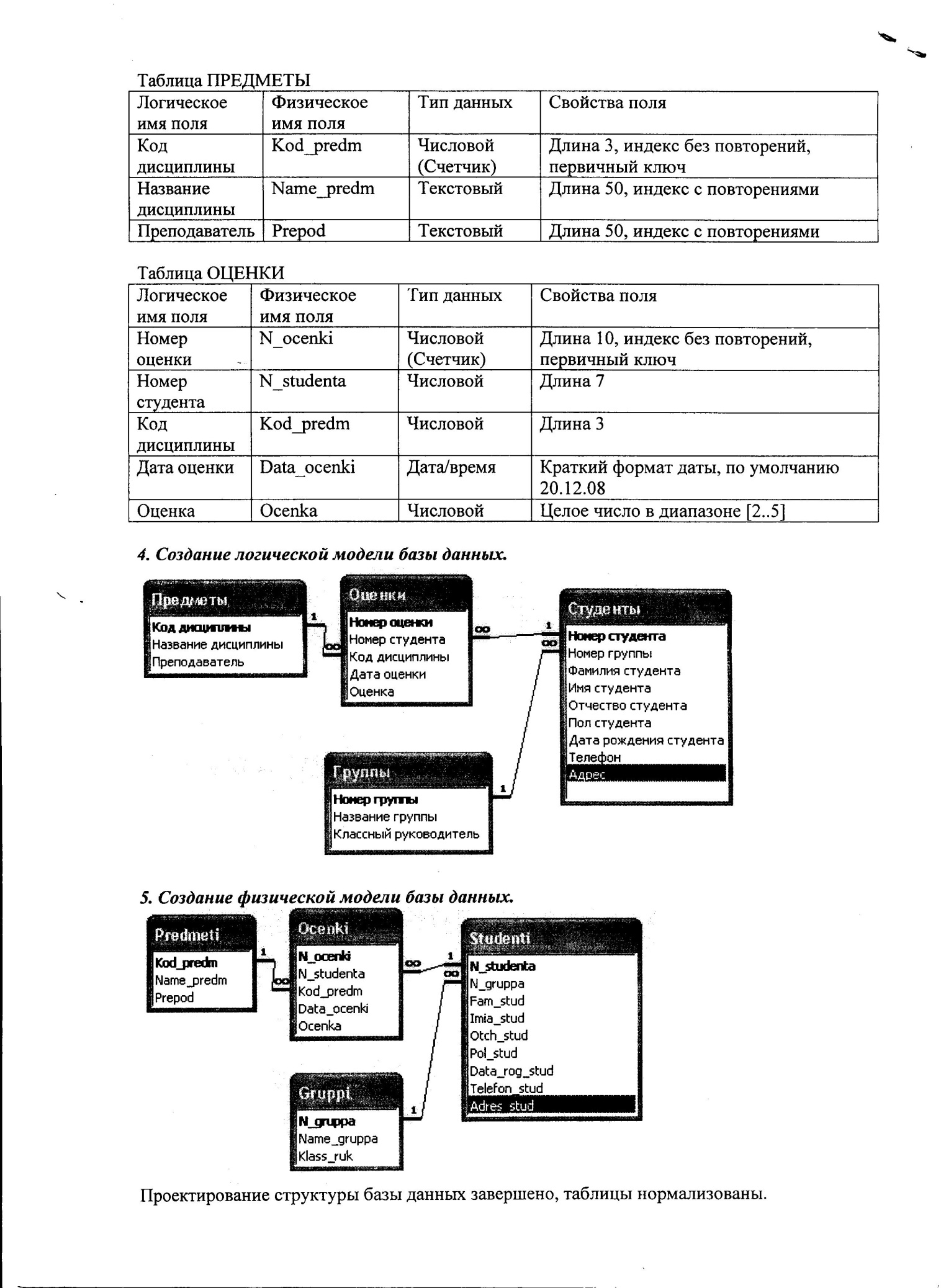
Таблица Предметы

|  |  |  |
| --- | --- | --- |
| **Логическое имя поля** | **Тип данных** | **Свойства поля** |
| Код дисциплины | Числовой (счетчик) | Длина 5, индекс без повторений, первичный ключ |
| Название дисциплины | Текстовый |  |
| Преподаватель | Текстовый |  |

Таблица Оценки

|  |  |  |
| --- | --- | --- |
| **Логическое имя поля** | **Тип данных** | **Свойства поля** |
| Номер оценки | Числовой (счетчик) | Длина 5, индекс без повторений, первичный ключ |
| Номер студента | Числовой | Внешний ключ |
| Код дисциплины | Числовой | Внешний ключ |
| Дата оценки | Дата/время |  |
| Оценка | Числовой | Число в диапазоне [2…5] |

4. Логическая структура данных

****

**Контрольные вопросы:**

1. Что такое нормализация таблиц?
2. При каких условиях таблица находится в первой нормальной форме?
3. При каких условиях таблица находится во второй нормальной форме?
4. При каких условиях таблица находится в третьей нормальной форме?
5. Какие типы данных существуют в приложении MSAccess?
6. Что такое первичный и внешний ключ?
7. Что такое логическая модель данных?

**Практическая работа 4. Задание ключей. Создание основных объектов БД**

**Цель:** научиться создавать таблицы баз данных, задавать их структуру, выбирать типы полей и управлять свойствами таблиц. Освоить приёмы наполнения таблиц

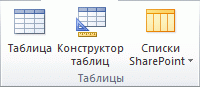
**Теоретические сведения**

В базе данных можно хранить данные в виде таблиц — тематических списков строк и столбцов.

**1.1 Создание базы данных**

1. Откройте Access. Если приложение Access уже открыто, на вкладке **Файл** выберите пункт **Создать**.
2. Выберите **пустую базу данных** или шаблон.
3. Введите имя базы данных, выберите расположение, а затем нажмите кнопку **Создать**. Когда база данных откроется, при необходимости нажмите кнопку **Включить содержимое** на желтой панели сообщений.

**1.2 Создание таблиц** осуществляется командами группы **Таблицы** на вкладке **Создание**.



**1.2.1 Создание таблицы в режиме таблицы.** В режиме таблицы можно начать ввод данных сразу, структура таблицы при этом будет создаваться автоматически. Полям присваиваются имена с последовательными номерами ("Поле1", "Поле2" и т. д.), а тип данных автоматически задается с учетом вводимых данных.

1. На вкладке **Создание** в группе **Таблицы** нажмите кнопку **Таблица**Выноска 4

Access создаст таблицу и выделит первую пустую ячейку в столбце **Щелкните для добавления**.

1. На вкладке **Поля** в группе **Добавление и удаление** выберите нужный тип поля. Если нужный тип поля не отображается, нажмите кнопку **Другие поля**Изображение кнопки.
2. Откроется список часто используемых типов полей. Выберите необходимый тип поля, и Access добавит в таблицу новое поле в точке вставки.

Поле можно переместить путем перетаскивания. При этом в таблице появляется вертикальная полоса вставки, указывающая место, где будет расположено поле.

1. Чтобы добавить данные, начните вводить их в первую пустую ячейку.
2. Для переименования столбца (поля) дважды щелкните его заголовок и введите новое название.

Присвойте полям значимые имена, чтобы при просмотре области **Список полей** было понятно, что содержится в каждом поле.

1. Чтобы переместить столбец, щелкните его заголовок для выделения столбца и перетащите столбец в нужное место. Можно выделить несколько смежных столбцов и перетащить их одновременно. Чтобы выделить несколько смежных столбцов, щелкните заголовок первого столбца, а затем, удерживая нажатой клавишу SHIFT, щелкните заголовок последнего столбца.

**1.2.2 Создание таблицы в режиме конструктора.** В режиме конструктора сначала следует создать структуру таблицы. Затем можно переключиться в режим таблицы для ввода данных.

1. На вкладке **Создание** в группе **Таблицы** нажмите кнопку **Конструктор таблиц** Изображение кнопки
2. Для каждого поля в таблице введите имя в столбце **Имя поля**, а затем в списке **Тип данных** выберите тип данных.
3. При желании можно ввести описание для каждого поля в столбце **Описание**. Это описание будет отображаться в строке состояния, когда в режиме таблицы курсор будет находиться в данном поле. Описание также отображается в строке состояния для любых элементов управления в форме или отчете, которые создаются путем перетаскивания этого поля из области **списка полей**, и любых элементов управления, которые создаются для этого поля при использовании мастера отчетов или мастера форм.

**Задание свойств полей в режиме конструктора.**    Независимо от способа создания таблицы рекомендуется проверить и задать свойства полей. Хотя некоторые свойства доступны в режиме таблицы, другие можно настроить только в режиме конструктора. Чтобы перейти в режим конструктора, в области навигации щелкните таблицу правой кнопкой мыши и выберите пункт **Конструктор**. Чтобы отобразить свойства поля, щелкните его в сетке конструктора. Свойства отображаются под сеткой конструктора в области **Свойства поля**.

Щелкните свойство поля, чтобы просмотреть его описание рядом со списком **Свойства поля**. Более подробные сведения можно получить, нажав кнопку справки.

В следующей таблице описаны некоторые наиболее часто изменяемые свойства полей.

| **Свойство** | **Описание** |
| --- | --- |
| **Размер поля** | Для текстовых полей это свойство указывает максимально допустимое количество знаков, сохраняемых в поле. Максимальное значение: 255. Для числовых полей это свойство определяет тип сохраняемых чисел ("Длинное целое", "Двойное с плавающей точкой" и т. д.). Для более рационального хранения данных рекомендуется выделять для хранения данных наименьший необходимый размер памяти. Если потребуется, это значение позже можно изменить. |
| **Формат поля** | Это свойство определяет формат отображения данных. Оно не влияет на фактические данные, сохраняемые в этом поле. Вы можете выбрать встроенный формат или задать собственный. |
| **Маска ввода** | Это свойство используется для определения общего шаблона для ввода любых данных в поле. Это позволяет обеспечить правильный ввод и нужное количество знаков для всех данных. Для получения справки по созданию маски ввода нажмите кнопку Кнопка "Построитель" справа от поля свойства. |
| **Значение по умолчанию** | Это свойство позволяет задать стандартное значение, которое будет отображаться в этом поле при добавлении новой записи. Например, для поля "Дата/время", в котором необходимо записывать дату добавления записи, в качестве значения по умолчанию можно ввести "Date()" (без кавычек). |
| **Обязательное поле** | Это свойство указывает, обязательно ли вводить значение в поле. Если для него задано значение **Да**, невозможно будет добавить запись, если в это поле не введено значение. |

1. **Добавление первичного ключа в таблицу**

Создайте первичный ключ для связывания данных между таблицами.

1) в области навигации щелкните таблицу правой кнопкой мыши и выберите пункт **Конструктор**;

2) выберите одно или несколько полей, которые требуется использовать в качестве первичного ключа.

3) На вкладке Конструктор нажмите кнопку **Ключевое поле**.

1. Когда все необходимые поля будут добавлены, сохраните таблицу (на вкладке **Файл** щелкните **Сохранить)**.
2. Вы можете в любой момент ввести данные в таблицу, переключив ее в режим таблицы и щелкнув первую пустую ячейку.

**1.2.3 Создание схемы данных**

Структура реляционной базы данных в Access задаётся схемой данных, которая имеет иерархическую структуру и называется канонической реляционной моделью предметной области.

Схема данных графически отображается в отдельном окне, где таблицы представлены списками полей, а связи – линиями между полями разных таблиц.

При построении схемы данных Access автоматически определяет по выбранному полю тип связи между таблицами. Если поле, по которому нужно установить связь, является уникальным ключом как в главной таблице, так и в подчинённой, Access устанавливает связь «один к одному». Если поле связи является уникальным ключом в главной таблице, а в подчинённой таблице является не ключевым или входит в составной ключ, Access устанавливает связь «один ко многим» от главной таблицы к подчинённой.

При создании схемы данных пользователь включает в неё таблицы и устанавливает связи между таблицами. Причём для связей типа «один к одному» и «один ко многим» можно задать параметр обеспечения целостности данных, а также автоматическое каскадное обновление и удаление связанных записей.

Обеспечение целостности данных означает выполнение для взаимосвязанных таблиц следующих условий корректировки базы данных:

* в подчинённую таблицу не может быть добавлена запись, для которой не существует в главной таблице ключа связи;
* в главной таблице нельзя удалить запись, если не удалены связанные с ней записи в подчинённой таблице;
* значение ключа связи главной таблицы должно приводить к изменению соответствующих значений в записях подчинённой таблицы.

В случае если пользователь нарушил эти условия в операциях обновления или удаления данных в связанных таблицах, Access выводит соответствующее сообщение и не допускает выполнения операции. Access автоматически отслеживает целостность данных, если между таблицами в схеме данных установлена связь с параметрами обеспечения целостности.

Создание схемы данных начинается с выполнения команды **Схема данных** в группе **Отношения** на вкладке ленты **Работа с базами данных**. В результате выполнения этой команды открывается окно схемы данных и диалоговое окно **Добавление таблицы**, в котором осуществляется выбор таблиц, включаемых в схему. Диалоговое окно **Добавление таблицы** откроется автоматически, если в базе данных еще не определена ни одна связь.

В окне **Добавление таблицы** отображены все таблицы и запросы, содержащиеся в базе данных. Для добавления таблицы необходимо выделить её и с помощью кнопки **Добавить** разместить в окне Схема данных. Затем необходимо добавить и другие таблицы и нажать кнопку Закрыть. В результате в окне Схема данных таблицы базы будут представлены окнами со списками своих полей и выделенными жирным шрифтом ключам.

Для установления связей между парой таблиц в схеме данных необходимо выделить в главной таблице уникальное ключевое поле, по которому устанавливается связь. Далее при нажатой кнопке мыши протащить курсор в соответствующее поле подчинённой таблицы. При установлении связи по составному ключу необходимо выделить все поля, входящие в ключ главной таблицы, и перетащить их на одно из полей связи в подчинённой таблицы. При установлении связи откроется окно **Связи**. При этом в строке **Тип отношений** автоматически устанавливается тип.

В этом же окне можно задать параметр **Обеспечение целостности данных** для выбранной связи. После задания параметра целостности можно отметить **каскадное обновление связанных полей** или **каскадное удаление связанных записей**.

**II Практическая часть**

1. Создать базу данных «Приёмная комиссия».
2. Создать таблицу **Факультеты** со следующей структурой (ключевое поле – Код ФКТ):

|  |  |
| --- | --- |
| **Имя поля** | **Тип поля** |
| Код\_ФКТ | текстовый |
| Факультет | текстовый |
| Экзамен\_1 | текстовый |
| Экзамен\_2 | текстовый |
| Экзамен\_3 | текстовый |

4. Сохранить таблицу с именем Факультеты.

5. Создать таблицу **Специальности** следующей структуры (ключевое поле – Код Спец):

|  |  |
| --- | --- |
| **Имя поля** | **Тип поля** |
| Код Спец | текстовый |
| Специальность | текстовый |
| Код ФКТ | текстовый |
| План | текстовый |

6. Назначить главным ключом таблицы поле Код Спец.

7. Сохранить таблицу с именем Специальности.

8. Связать таблицы Факультеты и Специальности через общее поле Код ФКТ, активизировать флажки **Обеспечить целостность данных**, **Каскадное обновление связанных полей** и **Каскадное удаление связанных записей**.

9. Ввести данные в таблицу Факультеты.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Код ФКТ** | **Факультет** | **Экзамен 1** | **Экзамен 2** | **Экзамен 3** |
| 01 | экономический | математика | география | русский язык |
| 02 | исторический | история Отечества | иностранный язык | сочинение |
| 03 | юридический | русский язык | иностранный язык | обществознание |

10. Аналогично заполнить таблицу Специальности. Ввести шесть строк – данные о шести специальностях:

|  |  |  |  |
| --- | --- | --- | --- |
| **Код Спец** | **Специальность** | **Код ФКТ** | **План** |
| 101 | финансы и кредит | 01 | 25 |
| 102 | бухгалтерский учет | 01 | 40 |
| 201 | история | 02 | 50 |
| 203 | политология | 02 | 25 |
| 310 | юриспруденция | 03 | 60 |
| 311 | социальная работа | 03 | 25 |

11. Создать таблицу **Абитуриенты** следующей структуры:

|  |  |
| --- | --- |
| **Имя поля** | **Тип поля** |
| Рег номер | текстовый |
| Код Спец | текстовый |
| Медаль | логический |
| Стаж | числовой |

12. Организовать связь таблицы АБИТУРИЕНТЫ с таблицей СПЕЦИАЛЬНОСТИ через поле КОД\_СПЕЦ.

13. Создать таблицу АНКЕТЫ следующей структуры:

|  |  |
| --- | --- |
| **Имя поля** | **Тип поля** |
| Рег ном | текстовый |
| Фамилия | текстовый |
| Имя | текстовый |
| Отчество | текстовый |
| Дата рождения | дата |
| Город | текстовый |
| Уч заведение | текстовый |

14. Организовать связь таблиц Анкеты и Абитуриенты через поле Рег номер.

16. Ввести записи в таблицу Анкеты.

Содержание таблицы:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Рег номер** | Фамилия | **Имя** | **Отчество** | **Дата рождения** | **Город** | **Уч заведение** |
| 1012 | Васильева | Ольга | Николаевна | 12.10.81 | Пермь | ПТУ № 8 |
| 1023 | Быков | Алексей | Ильич | 24.04.82 | Кунгур | Школа № 7 |
| 1119 | Круг | Борис | Моисеевич | 18.09.82 | Пермь | Школа № 102 |
| 1120 | Листьев | Дмитрий | Владимирович | 01.12.81 | Березники | Школа № 5 |
| 2010 | Елькин | Виктор | Алексеевич | 20.07.82 | Елабуга | ПТУ № 1 |
| 2015 | Мухин | Олег | Иванович | 25.03.78 | Березники | Школа № 77 |
| 2054 | Григорьева | Наталья | Дмитриевна | 14.02.80 | Пермь | Школа № 3 |
| 2132 | Зубова | Ирина | Афанасьевна | 22.11.81 | Пермь | Школа № 96 |
| 3005 | Анохин | Сергей | Петрович | 30.03.82 | Пермь | Школа № 12 |
| 3034 | Жакин | Николай | Якимович | 19.10.81 | Пермь | Школа № 12 |
| 3067 | Дикий | Илья | Борисович | 28.12.77 | Березники | Школа № 3 |
| 3118 | Ильин | Петр | Викторович | 14.07.80 | Кунгур | ПТУ № 8 |

18. Ввести записи в таблицу АБИТУРИЕНТЫ. Содержание таблицы:

|  |  |  |  |
| --- | --- | --- | --- |
| **Рег номер** | **Код Спец** | **Медаль** | **Стаж** |
| 1012 | 101 | ИСТИНА | 1 |
| 1023 | 101 | ЛОЖЬ | 0 |
| 1119 | 102 | ИСТИНА | 0 |
| 1120 | 102 | ИСТИНА | 0 |
| 2010 | 201 | ЛОЖЬ | 0 |
| 2015 | 203 | ЛОЖЬ | 3 |
| 2054 | 203 | ИСТИНА | 2 |
| 2132 | 201 | ЛОЖЬ | 0 |
| 3005 | 310 | ЛОЖЬ | 0 |
| 3034 | 311 | ЛОЖЬ | 1 |
| 3067 | 310 | ЛОЖЬ | 3 |
| 3118 | 310 | ЛОЖЬ | 2 |

*Пояснение*: В Access логическое значение ИСТИНА обозначается галочкой, заключенной в квадратик, ЛОЖЬ – пустым квадратом.

19. Создать таблицу ИТОГИ следующей структуры:

|  |  |
| --- | --- |
| **Имя поля** | **Тип поля** |
| Рег номер | текстовый |
| Зачисление | логический |

20. Связать таблицы ИТОГИ и АБИТУРИЕНТЫ через поле РЕГ\_НОМ.

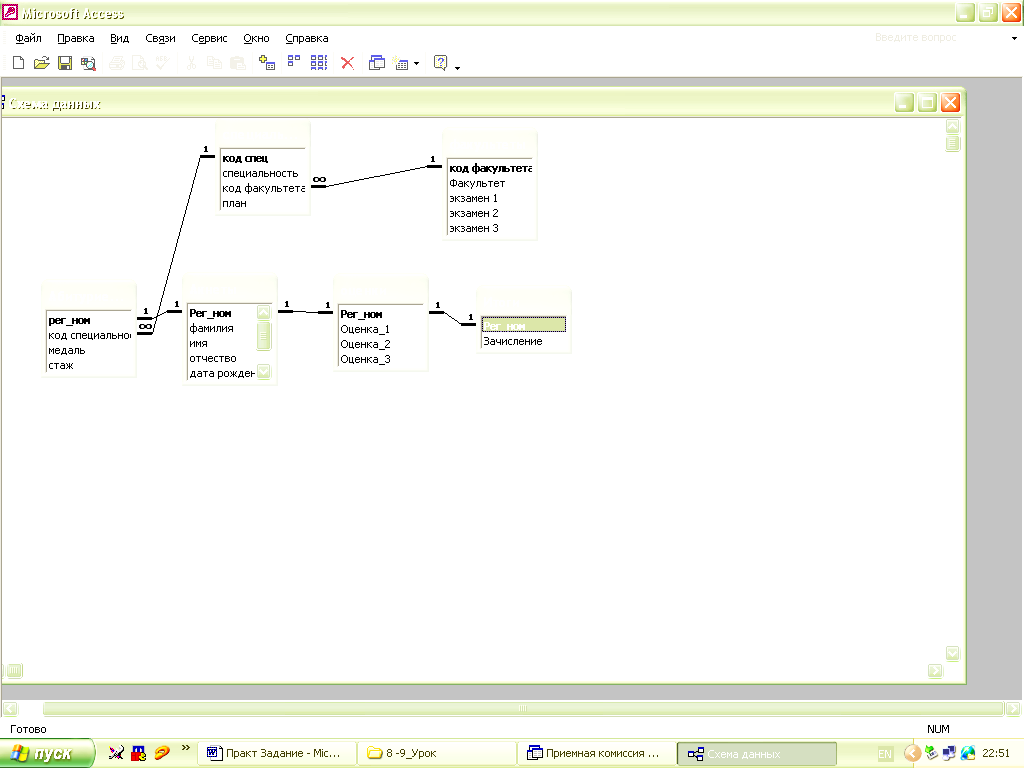
21. Ввести в таблицу данные в следующем виде:

|  |  |
| --- | --- |
| **РЕГ\_НОМ** | **ЗАЧИСЛЕНИЕ** |
| 1012 |  |
| 1023 |  |
| 1119 |  |
| 1120 |  |
| 2010 |  |
| 2015 |  |
| 2054 |  |
| 2132 |  |
| 3005 |  |
| 3034 |  |
| 3067 |  |
| 3118 |  |

*Пояснение*: В дальнейшем, после проведения приемной комиссией зачисления абитуриентов в университет по результатам приемных экзаменов, в графу ЗАЧИСЛЕНИЕ будет выставлено значение ИСТИНА (галочки в квадратах) для зачисленных абитуриентов.

**Проверь себя**

Если ты правильно установил все связи, то должно получится примерно следующее:



# Практическая работа 5. Создание проекта БД. Создание БД. Редактирование и модификация таблиц

**Цель:**Изучить механизмы приложения MSAccess для создания таблиц БД и ввода в них исходных данных

**Задание:**

1. Создать таблицы БД тремя различными способами в соответствии с индивидуальным заданием.
2. Заполнить созданные таблицы данными.
3. Описать ход работы.
4. Ответить на контрольные вопросы.
5. Оформить отчет.

**Теоретические сведения:**

**Основные режимы работы с таблицами**

В Microsoft Access существуют четыре режима работы с таблицами: режим Таблицы, режим Конструктора, режим Сводной таблицы и режим Сводной диаграммы.

В режиме Таблицы осуществляется работа с данными, находящимися в таблице: просмотр, редактирование, добавление, сортировка и т. п. В режиме Конструктора создается или модифицируется структура таблицы, т. е. задаются имена полей таблицы и их типы, поля описываются, задаются их свойства. В режимах Сводной таблицы и Сводной диаграммы удобно выполнять анализ данных, динамически изменяя способы их представления. Существует также дополнительный режим — режим Предварительного просмотра, который позволяет увидеть расположение данных на листе перед осуществлением печати таблицы.

Для быстрого перехода из одного режима в другой служит кнопка **Вид** на панелях инструментов **Таблица в режиме таблицы**, **Конструктор таблиц, Сводная таблица и Сводная диаграмма.** Чтобы перейти из режима в режим, достаточно нажать эту кнопку.

Открыть таблицу в режиме Таблицы можно несколькими способами:

* дважды щелкнуть мышью на имени таблицы в списке таблиц в окне базы данных;
* выделить таблицу в списке таблиц в окне базы данных и нажать кнопку **Открыть** в верхней части окна базы данных;
* щелкнуть правой кнопкой мыши на имени таблицы и из контекстного меню выбрать команду **Открыть**.

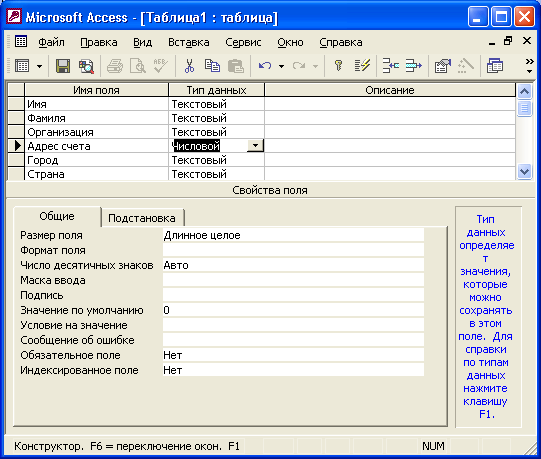
В любом случае на экране появится окно с содержимым таблицы. В верхней части таблицы располагаются имена *полей* (ячеек, находящихся в одном столбце таблицы), ниже следуют *записи* (строки таблицы), в которые вносятся данные. Одна запись всегда является *текущей*, и рядом с ней расположен *указатель текущей записи* (стрелка в поле выделения в левой части окна). В нижней части окна расположены *кнопки навигации*, позволяющие перемещать указатель текущей записи по таблице (на первую запись, на предыдущую запись, на следующую запись, на последнюю запись). Там же находятся поле номера текущей записи, кнопка создания новой записи и указатель общего количества записей в таблице. Для создания новой записи также служит последняя строка таблицы, отмеченная в поле выделения звездочкой.

*Горизонтальная полоса прокрутки* полей таблицы позволяет увидеть те поля таблицы, которые не поместились в окно таблицы. Аналогично вертикальная полоса прокрутки записей таблицы позволяет увидеть записи, находящиеся за пределами окна.

Открыть таблицу в режиме Конструктора можно одним из двух следующих способов:

* выделить таблицу в списке таблиц в окне базы данных и нажать кнопку **Конструктор** в верхней части окна базы данных;
* щелкнуть правой кнопкой мыши на имени таблицы и из контекстного меню выбрать команду **Конструктор**.

После выполнения указанных действий на экране появится окно таблицы в режиме Конструктора, а также панель инструментов **Конструктор таблиц** (рис. 1).

****рис. 1

Окно таблицы в режиме Конструктора состоит из двух панелей.

* Верхняя панель содержит таблицу из трех столбцов**: Имя поля**, **Тип данных** и **Описание**. В столбце **Имя поля** содержатся имена полей таблицы, в столбце **Тип данных** указывается тип данных для каждого поля. Заполнение двух первых столбцов является обязательным.
* На нижней панели располагаются две вкладки свойств полей: **Общие** и **Подстановка**, а также область контекстной справки.

*Ключевое поле* таблицы помечается специальным значком — ключик в поле выделения в левой части окна. Чтобы выделить поле, достаточно просто щелкнуть мышью по строке, в которой описывается данное поле. При этом в нижней части окна будут показаны параметры именно для этого поля. Перемещаться между столбцами в верхней панели окна можно с помощью клавиши <Таb>, а для перемещения по списку полей вверх и вниз пользуйтесь клавишами со стрелками. С помощью клавиши <F6> можно перемещаться между панелями.

В верхней части окна приложения появляется стандартная панель инструментов **Конструктор таблиц**. Рассмотрим основные кнопки этой панели инструментов (описываются только кнопки, уникальные для данного режима).

* Ключевое поле — добавляет в таблицу ключевое поле или удаляет его.
* Индексы — открывает окно редактирования индексов.
* Вставить строки — позволяет добавлять поля в таблицу.
* Удалить строки — позволяет удалять поля из таблицы.
* Свойства — открывает окно свойств таблицы.
* Построить — открывает окно Построителя полей.

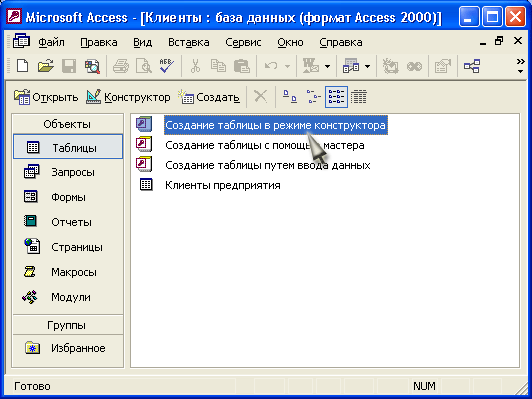
Остальные кнопки на этой панели те же самые, что и на описанной ранее панели **Таблица в режиме таблицы**.

Новый режим — Сводной таблицы — позволяет представлять табличные данные в более удобном и обозримом виде. Сводная таблица позволяет группировать, суммировать или каким-то другим образом обрабатывать данные из обычной таблицы Access. Этот режим является с одной стороны аналогом сводных таблиц в Excel, а с другой стороны — развитием уже давно используемых в Access перекрестных запросов.

Режим Сводной диаграммы тесно связан со сводной таблицей. Это просто графическое представление сводной таблицы. В предыдущих версиях Access диаграммы могли использоваться только в отчетах. Теперь они стали динамическими, и есть возможность создавать эти диаграммы и управлять ими в интерактивном режиме прямо на экране монитора.

**Создание таблиц**

В Access используются три способа создания таблиц: путем ввода данных, с помощью Конструктора таблиц и с помощью Мастера создания таблиц. Для каждого из этих способов существует специальный ярлык новых объектов в списке таблиц (рис. 2). Если вы создали новый файл базы данных, то, кроме этих ярлыков, в списке таблиц больше ничего нет.

****рис.2

Для создания новой таблицы можно воспользоваться любым из описанных способов. Обычно создание таблиц путем ввода данных используют тогда, когда структура таблицы очень проста, например справочник. Тогда Access сама догадывается о том, как установить свойства полей. Потом можно переключить таблицу в режим Конструктора и посмотреть, как эти свойства будут установлены. Использование Мастера таблиц оправдано, если нужно создать какую-то типовую таблицу, пользуясь шаблонами, которые он предложит.

**Ход работы:**

**1. Создать первую таблицу базы данных путем ввода данных**

* Открыть окно новой или существующей базы данных и в нем открыть список таблиц.
* Выделить в списке таблиц ярлык **Создание таблицы путем ввода данных** и дважды щелкнуть по нему левой кнопкой мыши. Появится пустая таблица со стандартными названиями столбцов: Поле1, Поле2 и т. д. По умолчанию для создания таблицы предоставляется десять полей.
* В поля этой таблицы нужно ввести требуемые данные. Тип данных в одном поле (столбце) во всех записях должен быть одинаковым.
* Можно ввести свои собственные заголовки столбцов, для чего щелкнуть правой кнопкой мыши по заголовку столбца и выберите из контекстного меню команду **Переименовать столбец**. После этого стандартное название столбца подсвечивается, и вы можете вводить свое название.
* По окончании ввода данных нажать кнопку **Закрыть.**
* В ответ на вопрос **Сохранить изменения макета или структуры таблицы<имя таблицы>?** нажатькнопку**Да**.
* В окне **Сохранение** в поле **Имя таблицы** ввести имя новой таблицы и нажать кнопку ОК.
* Microsoft Access выдаст сообщение **Ключевые поля не заданы** и вопрос **Создать ключевое поле сейчас?**. Нажать кнопку **Нет**, если одно или несколько полей в таблице могут однозначно идентифицировать записи в таблице и служить первичным ключом, или кнопку **Да**, и тогда Access создаст дополнительное поле, которое сделает ключевым.

**2. Создать вторую таблицу базы данных с помощью Мастера таблиц**

* Дважды щелкнуть левой кнопкой мыши на ярлыке **Создание таблицы с помощью мастера**.
* В диалоговом окне **Создание таблиц** выбрать назначение таблицы: **Деловые** (для иных целей можно выбрать и **Личные**).
* В поле **Образцы таблиц** выбрать подходящую таблицу — **Список рассылки**.
* Переместить нужные поля из списка **Образцы полей** в список **Поля новой таблицы**. Для этого следует воспользоваться кнопками со стрелками: > — переместить одно поле, » — переместить все поля.
* Повторять описанные в пп. 3—4 действия до тех пор, пока не будет создан подходящий перечень полей для новой таблицы.
* При необходимости переименовать какое-либо поле, нужно выделить его в списке **Поля новой таблицы** и нажать кнопку **Переименовать поле**, затем ввести в диалоговом окне **Переименование поля** новое имя поля и нажать кнопку **ОК**. Нажать кнопку **Далее**.
* В следующем диалоговом окне в поле **Задайте имя для новой таблицы** ввести имя новой таблицы.
* Выбрать способ определения ключа **Microsoft Access автоматически определяет ключ** Тогда в таблицу будет добавлено поле автоматической нумерации. Нажать кнопку **Далее**.
* В случае, если было выбрано самостоятельное определение ключа, в поле со списком **Выберите поле с уникальными для каждой записи данными** необходимо выбрать имя поля, которое станет ключевым. Затем нужно определить тип данных, которые будут содержаться в ключевом поле.
* В следующем окне Мастер таблиц предлагает установить связи между существующими таблицами и создаваемой таблицей. Он пытается сам создать связи между таблицами и предоставляет информацию о создаваемых связях пользователю в виде списка. При необходимости исправить связи нужно выбрать соответствующую строку списка и нажать кнопку **Связи**. В окне **Связи** нужно выбрать тип связи с данной таблицей.
* Открыв последнее диалоговое окно, можно решить, что вы будете делать с таблицей дальше. Выберите переключатель **Изменить структуру таблицы**, чтобы перейти в режим Конструктора таблиц, и нажмите на кнопку **Готово.**

**3. Создать третью таблицу базы данных с помощью Конструктора таблиц**

* Дважды щелкнуть левой кнопкой мыши на ярлыке **Создание таблицы с помощью конструктора** или нажать на кнопку **Создать** в верхней части окна базы данных, выбрать из списка в окне **Новая таблица** элемент **Конструктор** и нажать кнопку ОК. В том и в другом случае откроется пустое окно Конструктора таблиц.
* В окне Конструктора таблиц в столбец **Имя поля** ввести имена полей создаваемой таблицы.
* В столбце **Тип данных** для каждого поля таблицы выбрать из раскрывающегося списка тип данных, которые будут содержаться в этом поле.
* В столбце **Описание** можно ввести описание данного поля (не обязательно).

**4. Придумать всю информацию для базы данных по индивидуальному заданию и заполнить все созданные таблицы этой информацией.**

**Контрольные вопросы:**

1. Каковы основные режимы работы с таблицами в приложении MSAccess?

2. Как изменить режим работы с таблицами?

3. Какие функции выполняют основные кнопки панели инструментов Конструктор таблиц?

4. Как осуществляется создание таблиц в приложении MSAccess?

**Практическая работа 6. Редактирование, добавление и удаление записей в таблице. Применение логических условий к записям. Открытие, редактирование и пополнение табличного файла**

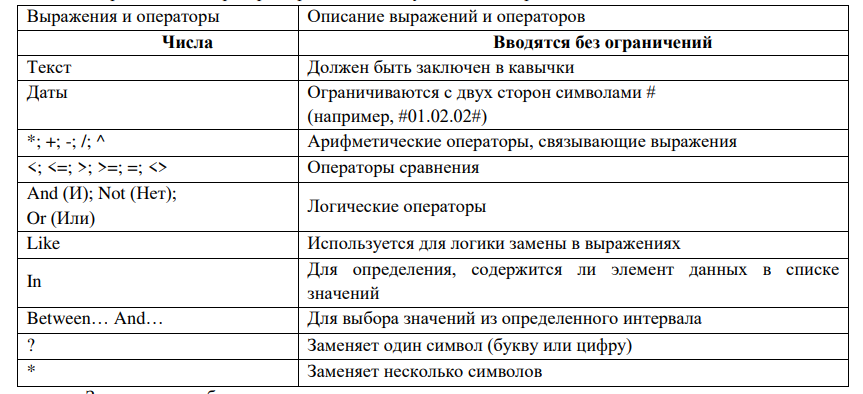
**Цели работы:** закрепить навыки по редактированию таблиц; познакомиться с основными видами запросов; научиться создавать запросы на выборку различными способами; научиться создавать сложные запросы; научиться создавать перекрестные запросы.

**Теоретические сведения**

Запрос - это средство, с помощью которого извлекается из базы данных информация, отвечающая определенным критериям. Результаты запроса представляют не все записи из таблицы, а только те, которые удовлетворяют запросу.

Запросы состоят из ряда условий, каждое условие состоит из трех элементов:

1. поле, которое используется для сравнения;
2. оператор, описывающий тип сравнения;
3. величина, с которой должно сравниваться значение поля.



Запросы могут быть простые, сложные перекрестные.

**Задание для практической работы**

Создайте запросы к вашей базе данных

1. Откройте свою учебную базу данных.
2. Создайте запрос на выборку студентов, у которых по всем предметам только хорошие оценки с помощью *Мастера запросов.*

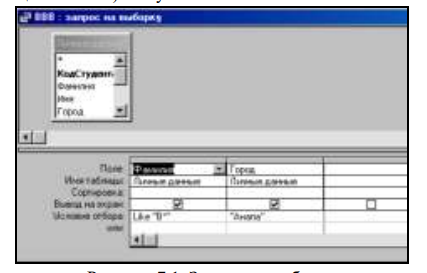
* На панели инструментов выберите команду <Мастер запросов>.
* В появившемся диалоговом окне выберите <Простой запрос> и щелкните по кнопке <OK>.
* В следующем окне выберите таблицу, по которой строится запрос (<Ведомость успеваемости>), и те поля, которые участвуют в запросе. Перенесите их в правую часть окна и нажмите <Далее>. В следующем окне тоже нажмите <Далее>.
* В другом окне дайте название запроса «Хорошисты» и нажмите <Готово>.
* Появится таблица <Хорошисты>, в которой отражены фамилии всех студентов и изучаемые предметы.
* Откройте таблицу «Хорошисты», перейдите в режим <Конструктор>. Здесь в поле <Условия отбора> под каждым предметом поставьте условие >=4 или 4OR5.

Примечание: Галочки в каждом поле означают, что по вашему выбору можно включить или убрать любое поле на выборку

* Перейдите в режим таблицы, ответив <Да> на вопрос о сохранении запроса. (В таблице должны остаться фамилии «хорошистов»).

1. С помощью <Конструктора запросов> создайте запрос на выборку по таблице <Личные данные>.

* Щелкните по таблице <Личные данные>, зайдите в меню <Создание>, выберите команду <Конструктор запросов >.
* Добавьте нужную таблицу в поле запроса. Выделите её в списке и щелкните по кнопке <Добавить>. Закройте окно <Добавление таблицы>.
* Выберите студентов, чьи фамилии начинаются на букву «В» и которые проживают в Анапе. Для этого:
* добавьте в строку <Поле> два поля <Фамилия> и <Г ород>;
* в строке <Условия отбора> в первом столбце укажите значение Like “B\* ”, а во втором столбце с названием <Город> - «Анапа»;
* закройте запрос, сохранив его под названием “ВВВ” (у вас должны остаться в списке студенты, проживающие в Анапе).



Индивидуальные задания к практической работе

а) Составьте запрос с названием <Запрос 1> на базе таблицы <Ведомость успеваемости>, в котором будут указаны студенты, имеющие по первым двум предметам оценки не менее «4». (Выполните запрос или через *Конструктор запросов,* или через *Мастер запросов)*

б) Составьте <Запрос 2> на базе таблицы <Ведомость успеваемости>, в котором будут указаны студенты, имеющие не более 30 часов пропусков по неуважительной причине. Добавьте в этот запрос поле пропуски по уважительной причине в интервале от 30 часов до 45 часов (используйте оператор *BetweenAnd*^)

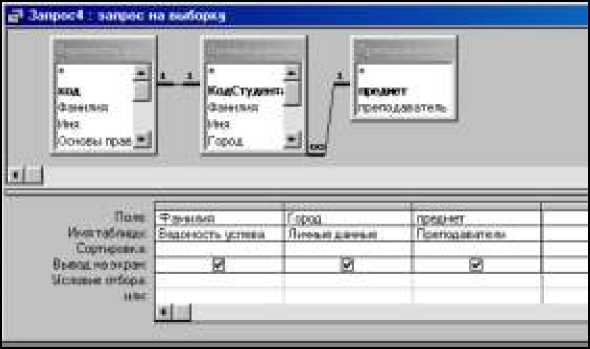
в) Составьте <Запрос> на базе таблицы <Личные данные>. Выведите список студентов, которым на данный момент, т.е. на сегодняшнее число, исполнилось уже 17 лет (используйте оператор *Between^ AndM)*

Примечание: Дата записывается с использованием символа #, например, #01.02.02.#

1. Составьте запрос на базе трех таблиц <Ведомость успеваемости>, <Личные данные> и <Преподаватель>. Выберите студентов, которые проживают в Новороссийске и у которых любимый предмет «Менеджмент». Озаглавьте <Запрос 4>. Используйте <Конструктор запросов>.

* В меню <Создание> выберите <Конструктор запросов>.
* Добавьте все три таблицы в поле запроса. Закройте окно <Добавление таблицы>.

В первый столбец в строку <Поле> перетащите из первой таблицы с помощью мышки <Фамилия>, из второй таблицы во второй столбец <Город> и из третей таблицы в третий столбец строки <Поле> - <Предмет> (Рисунок.2).



* В поле <Условия отбора> в столбце <Город> введите город «Новороссийск», в столбец <Предмет> введите «Менеджмент».
* Сохраните запрос под именем <Запрос 4>.
* Откройте запрос и проверьте результат проделанной работы.

1. Выполните запрос на создание новой таблицы, в которой должны быть поля <Фамилия>, <Имя>, <Пропуски по неуважительной причине>, <Город> и <Предмет>.

* В меню <Создание> выберите <Конструктор запросов>.
* Добавьте все три таблицы из списка окна <Добавление таблицы>. Закройте этоокно
* В первую строчку <Поле> из первой таблицы перенесите в первый столбец поля <Фамилия>, во второй <Имя> и в третий <Пропуски по уважительной причине>, в четвертый столбец перетащите поле <Город> из второй таблицы и в последнем столбце будет поле <Предмет> из третьей таблицы.
* Закройте запрос, сохранив его с именем <Запрос 5>.

1. Создайте *перекрестный запрос.*

Допустим, нужно посчитать для ведомости, сколько в группе человек получили по предмету “троек”, “четверок” и “пятерок”. Для этих целей используется *перекрестный запрос.*

* В меню <Создание> выберите <Мастер запросов>.
* В диалоговом окне выберите <Перекрестн^1й запрос>, щелкните по кнопке <OK>.
* В окне <Создание перекрестных запросов> выделите таблицу <Ведомость успеваемости> и щелкните <Далее>.
* Выберите поля, значения которого будут использоваться в качестве заголовок строк - это <Фамилия> и <Имя>. Щелкните по кнопке <Далее>.
* Выберите поле, значение которого будут использоваться в качестве заголовков столбцов, например <Менеджмент>. Щелкните по кнопке <Далее>.
* Выберите функцию, по которой будут вычисляться значения ячеек на пересечении столбцов и строк (в данном случае Count- количество). Щелкните по кнопке <Далее>.
* Задайте имя запроса <Итог по менеджменту> и щелкните по кнопке <Готово>.

Составьте аналогичные запросы для оценок по трем другим предметам.

1. Предъявите преподавателю все запросы своей базы данных на экране дисплея.

Завершите работу с Access

**Практическая работа 7. Создание ключевых полей. Задание индексов. Установление и удаление связей между таблицами**

**Цель:**Изучить механизмы приложения MSAccess для установлениявзаимосвязей между таблицами в базе данных.

**Задание:**

1. Установить взаимосвязи между таблицами БД в соответствии с индивидуальным заданием.
2. Описать ход работы.
3. Ответить на контрольные вопросы.
4. Оформить отчет

**Теоретические сведения:**

На практике часто используются связи, устанавливающие различные виды соответствия между объектами «связанных» ти­пов, — это один к одному (1:1), один ко многим (1:М), многие ко многим (М:М).

Связь ***один к одному***означает, что каждому экземпляру первого объекта *(А)* соответствует только один экземпляр второго объекта *(В)* и, наоборот, каждому экземпляру второго объекта *(В)* соот­ветствует только один экземпляр первого объекта *(А).*

Связь ***один ко многим***означает, что каждому экземпляру од­ного объекта *(А)* может соответствовать несколько экземпляров другого объекта *(В),* а каждому экземпляру второго объекта *(В)* может соответствовать только один экземпляр первого объекта *(А).*

Связь ***многие ко многим***означает, что каждому экземпляру од­ного объекта *(А)* могут соответствовать несколько экземпляров второго объекта *(В)* и, наоборот, каждому экземпляру второго объекта (В) могут соответствовать тоже несколько экземпляров первого объекта (А).

Для того чтобы было удобно просматривать, создавать, удалять и модифицировать связи между таблицами, в Microsoft Access используется **схема данных** (рис.1).

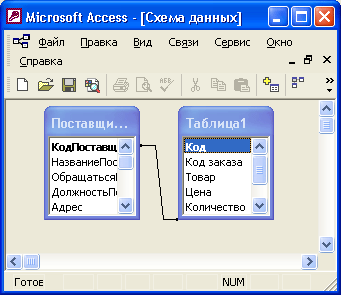


рис.1

**Создание и использование индексов**

С целью ускорения поиска и сортировки данных в любой СУБД используются индексы. *Индекс* является средством, которое обеспечивает быстрый доступ к данным в таблице на основе значений одного или нескольких столбцов. Индекс представляет собой упорядоченный список значений и ссылок на те записи, в которых хранятся эти значения. Чтобы найти нужные записи, СУБД сначала ищет требуемое значение в индексе, а затем по ссылкам быстро отбирает соответствующие записи. Индексы бывают двух типов: простые и составные. *Простые* индексы представляют собой индексы, созданные по одному столбцу. Индекс, построенный по нескольким столбцам, называется *составным.* Примером составного индекса может быть индекс, построенный по столбцам "Фамилия" и "Имя".

Однако применение индексов приносит не только преимущества, но и недостатки. Главным среди них является тот, что при добавлении и удалении записей или при обновлении значений в индексном столбце требуется обновлять индекс, что при большом количестве индексов в таблице может замедлять работу. Поэтому индексы обычно рекомендуется создавать только для тех столбцов таблицы, по которым наиболее часто выполняется поиск записей. Во многих СУБД (например, FoxPro) индексы хранятся в отдельных файлах и являются предметом заботы разработчиков, т. к. при нарушении индекса поиск данных выполняется некорректно. В Microsoft Access индексы хранятся в том же файле базы данных, что и таблицы и другие объекты Access. Индексировать можно любые поля, кроме МЕМО-полей и полей типа Гиперссылка.

Некоторые общие принципы, связанные с созданием индексов:

* индексы необходимо создавать для столбцов, по которым часто производится поиск и операции сортировки. При этом необходимо учесть, что индексы всегда автоматически создаются для столбцов, на которые накладывается ограничение первичного ключа;
* индекс обязательно в автоматическом режиме создается для столбцов, на которые наложено ограничение уникальности;
* лучше всего индексы создавать для тех полей, в которых - минимальное число повторяющихся значений и данные распределены равномерно.
* если поиск постоянно производится по определенному набору столбцов (одновременно), то в этом случае есть смысл создать композитный индекс - один индекс для группы столбцов;
* при внесении изменений в таблицы автоматически изменяются и индексы, наложенные на эту таблицу. В результате индекс может быть сильно фрагментирован, что сказывается на производительности. Периодически следует проверять степень фрагментации индексов и дефрагментировать их. При загрузке большого количества данных иногда есть смысл вначале удалить все индексы, а после завершения операции создать их заново.

В Microsoft Access автоматически создаются индексы для полей, на которые назначены ограничение первичного или вторичного ключа. Для остальных столбцов индексы можно создать явно в режиме конструктора для таблицы: свойства поля, строка "Индексированное поле". Для этого поля предусмотрено три варианта: Нет (по умолчанию), Да (допускаются совпадения) и Нет (Совпадения не допускаются).

**Типы индексов**

*Уникальный индекс*

Уникальным является индекс, в котором не может быть двух строк с одним и тем же значением индекса. В большинстве баз данных не допускается сохранение таблицы с созданным в ней уникальным индексом, если в существующих данных имеются повторяющиеся значения ключа. В базе данных может быть также запрещено добавление новых данных, приводящее к появлению дубликатов значений ключа в таблице. Например, если создать уникальный индекс по фамилии сотрудника, в ней не должно быть двух сотрудников с одинаковыми фамилиями.

*Индекс первичного ключа*

В таблице базы данных обычно имеется столбец или комбинация столбцов, значения которых однозначно идентифицируют каждую строку таблицы. Такой столбец называется первичным ключом таблицы.

Определение первичного ключа для таблицы в схеме базы данных автоматически приводит к созданию индекса первичного ключа, который является разновидностью уникального индекса. Этот индекс требует уникальности каждого значения первичного ключа. Он также обеспечивает быстрый доступ к данным при использовании индекса первичного ключа в запросах.

*Кластерный индекс*

В кластерном индексе физический порядок строк в таблице совпадает с логическим (индексированным) порядком значений ключа. У таблицы может быть только один кластерный индекс.

Если индекс не кластеризован, физический порядок строк таблицы отличается от логического порядка значений ключа. Кластерный индекс обычно обеспечивает более быстрый доступ к данным, чем прочие индексы.

*Полнотекстовый индекс*

Полнотекстовый индекс создается, если нужно выполнять полнотекстовый поиск по текстовым столбцам таблиц базы данных. Полнотекстовый индекс зависит от обычного индекса, поэтому сначала надо будет создать его. Обычный индекс должен быть создан для одного столбца, не допускающего значения NULL, и лучше всего выбирать столбцы с маленькими значениями, а не большими.

Диалоговое окно **Индексы** используется также для просмотра, изменения и удаления существующих индексов. Изменить можно:

* название индекса в поле **Индекс**;
* поле таблицы, соответствующее данному индексу, выбрав новое поле из списка в поле **Имя поля**;
* порядок сортировки в поле **Порядок сортировки**;

Свойства индекса в нижней части окна:

* **Ключевое поле** определяет, является ли индексированное поле ключевым;
* **Уникальный индекс** определяет, должно ли быть каждое значение в этом поле уникальным;
* **Пропуск пустых полей** определяет, включаются или не включаются в индекс записи с пустым значением данного поля

**Ход работы;**

**1. Установить взаимосвязи между таблицами**

* Чтобы открыть схему данных, необходимо выполнить команду **Сервис, Схема данных**. По умолчанию схема будет содержать все таблицы со связями.
* Чтобы убрать какую-либо таблицу из схемы данных, необходимо щелкнуть правой кнопкой мыши на любом месте этой таблицы и из контекстного меню выбрать команду **Скрыть таблицу**.
* Чтобы вновь добавить в эту схему свернутую таблицу или таблицу, у которой связи еще не установлены, необходимо:

1. Щелкнуть правой кнопкой мыши на свободном пространстве схемы данных и из контекстного меню выбрать команду **Добавить таблицу**.
2. В диалоговом окне **Добавление таблицы** раскрыть вкладку **Таблицы**, выбрать из списка нужную таблицу и нажать кнопку **Добавить**.
3. Нажать кнопку **Закрыть**, чтобы закрыть диалоговое окно **Добавление таблицы**.

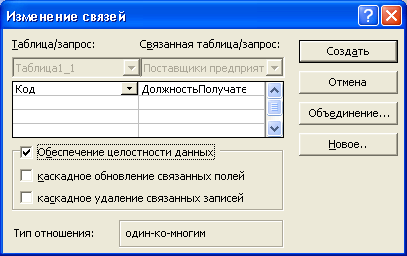


рис.2

* Если нужная связь автоматически не создана, необходимо выбрать команду **Связи, Изменить связь** из главного меню Access. Эта команда появляется в меню, когда открыто окно **Схема данных**.
* В диалоговом окне **Изменение** связей можно выбрать из списков названия связанных таблиц и полей для связывания. 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 устанавливали связь графически, то все поля в списках уже выбраны, и нужно только определить правила ссылочной целостности.
* Для этого устанавливают флажок **Обеспечение целостности данных** и один или оба флажка: **каскадное обновление связанных полей** и **каскадное удаление связанных записей**.
* При необходимости можно изменить параметры объединения, для чего требуется нажать кнопку **Объединение** и выбрать один из переключателей:

- Объединение только тех записей, в которых связанные поля обеих таблиц совпадают;

- Объединение ВСЕХ записей из <имя первой таблицы> и только тех записей из <имя второй таблицы>, в которых связанные поля совпадают;

- Объединение ВСЕХ записей из <имя второй таблицы> и только тех записей из <имя первой таблицы >, в которых связанные поля совпадают.

* По умолчанию устанавливается первый переключатель. Нажать кнопку **ОК**.
* 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.
* После установки всех параметров связи необходимо нажать кнопку ОК в окне **Параметры объединения** и закрыть окно **Схема данных**, нажав на кнопку  **Закрыть** в правом верхнем углу окна.

**2. Проверить, что все взаимосвязи установлены верно**

* Включить просмотр любой таблицы. Слева перед каждой строчкой должен появиться знак «**+**».
* Щелкнуть на «+» левой кнопкой мыши и убедиться, что при этом появляется подтаблица с данными.

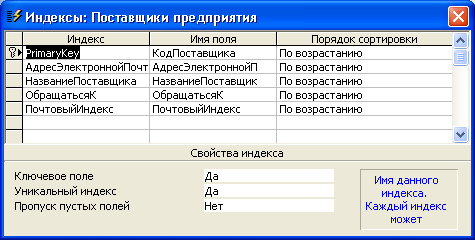
**3. Создать простые индексы базы данных:**

* Открыть таблицу в режиме Конструктора.
* Выбрать поле, для которого требуется создать индекс.
* Открыть вкладку **Общие** и выбрать для свойства **Индексированное поле** значение **Да (Допускаются совпадения)** или **Да (Совпадения не допускаются)**

Ключевое поле таблицы автоматически индексируется и свойству **Индексированное поле** присваивается значение Да **(Совпадения не допускаются)**.

**4. Создать составные индексы базы данных:**

* Открыть таблицу в режиме Конструктора.
* На панели инструментов **Конструктор таблиц** нажать кнопку **Индексы**
* В первой пустой строке поля **Индекс** ввести имя индекса.



* В поле **Имя поля** нажать на стрелку и выбрать первое поле, для которого необходимо создать индекс.
* В следующей строке поля **Имя поля** указать второе индексируемое поле. (Для данной строки поле **Индекс** должно оставаться пустым.) Повторите эту операцию для всех полей, которые необходимо включить в индекс. В индексе может быть использовано до 10 полей.

**Контрольные вопросы:**

1. Как добавить в Схему данных новую таблицу?
2. Как изменить связи между таблицами?
3. Какие три типа связей можно установить между таблицами?
4. Что такое индекс, простой и составной индекс?
5. Как осуществляется алгоритм создания простого индекса?
6. Как осуществляется алгоритм создания составного индекса?

# Практическая работа 8. Проведение сортировки и фильтрации данных. Поиск данных по одному и нескольким полям. Поиск данных в таблице

**Цель:**Изучить механизмы приложения MSAccess для проведения сортировки, фильтрации, поиска данных.

**Задание:**

1. Отсортировать данные таблиц по ключевым полям:
2. Отсортировать данные таблиц по значению нескольких полей
3. Выполнить поиск данных по одному из полей таблицы, поиск и замену данных
4. Выполнить фильтрацию данных по выделенному фрагменту
5. Выполнить экспорт отфильтрованных данных
6. Выполнить фильтрацию данных с помощью расширенного фильтра
7. Описать ход работы.
8. Ответить на контрольные вопросы.
9. Оформить отчет

**Теоретические сведения**

**Сортировка данных в таблице**

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). Именно так и позволяет делать Access.

Чтобы правильно применять сортировку, нужно знать несколько простых правил:

* При сортировке в возрастающем порядке записи, содержащие пустые поля (с пустыми значениями), указываются в списке первыми.
* Числа, находящиеся в текстовых полях, сортируются как строки символов, а не как числовые значения. Если нужно отсортировать их в числовом порядке, все текстовые строки должны содержать одинаковое количество символов. Если строка содержит меньшее количество символов, то сначала нужно вставить незначащие нули.
* При сохранении таблицы сохраняется и порядок ее сортировки.

**Фильтрация данных**

Фильтрация данных - процесс, связанный с автоматическим определением принадлежности данных к некоторому множеству значений.

В Microsoft Access предусмотрено четыре способа отбора записей с помощью фильтров: фильтр по выделенному фрагменту, обычный фильтр, поле Фильтр для и расширенный фильтр.

Фильтр по выделенному фрагменту, обычный фильтр и поле Фильтр для являются очень простыми способами отбора записей, причем самым простым является фильтр по выделенному фрагменту — он позволяет найти все записи, содержащие определенное значение в выбранном поле. Обычный фильтр используется для отбора записей по значениям нескольких полей. Поле Фильтр для используется, если фокус ввода находится в поле таблицы и нужно ввести конкретное искомое значение или выражение, результат которого будет применяться в качестве условия отбора. Для создания сложных фильтров следует использовать окно расширенного фильтра.

**Ход работы:**

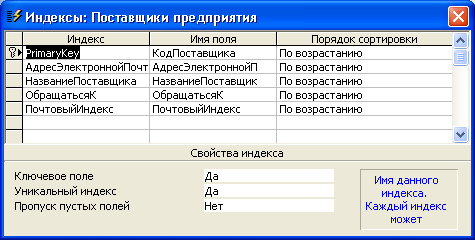
**1. Создать простые индексы базы данных:**

* Открыть таблицу в режиме Конструктора.
* Выбрать поле, для которого требуется создать индекс.
* Открыть вкладку **Общие** и выбрать для свойства **Индексированное поле** значение **Да (Допускаются совпадения)** или **Да (Совпадения не допускаются)**

Ключевое поле таблицы автоматически индексируется и свойству **Индексированное поле** присваивается значение Да **(Совпадения не допускаются)**.

**2. Создать составные индексы базы данных:**

* Открыть таблицу в режиме Конструктора.
* На панели инструментов **Конструктор таблиц** нажать кнопку **Индексы**
* В первой пустой строке поля **Индекс** ввести имя индекса.



* В поле **Имя поля** нажать на стрелку и выбрать первое поле, для которого необходимо создать индекс.
* В следующей строке поля **Имя поля** указать второе индексируемое поле. (Для данной строки поле **Индекс** должно оставаться пустым.) Повторите эту операцию для всех полей, которые необходимо включить в индекс. В индексе может быть использовано до 10 полей.

**3. Отсортировать данные таблиц по ключевым полям:**

* 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**.

**4. Отсортировать данные таблиц по значению нескольких полей:**

* переместите сначала сортируемые столбцы таким образом, чтобы они оказались, во-первых, рядом, а во-вторых, с учетом приоритетов, — приоритеты устанавливаются слева направо, т. к. первыми будут сортироваться значения в крайнем левом столбце.
* выделите все столбцы и нажмите, соответственно, кнопку **Сортировка по возрастанию** или **Сортировка по убыванию** на панели инструментов.

**5. Выполнить поиск данных по одному из полей таблицы**

* Открыть таблицу в режиме Таблицы.
* Если известно, в каком столбце нужно вести поиск, выделить этот столбец (достаточно поместить курсор в любое поле этого столбца).
* Нажать кнопку *Найти* на панели инструментов *Режим таблицы* и открыть вкладку *Поиск* или выполнить команду меню *Правка, Найти.* Появится диалоговое окно *Поиск и замена*.
* Ввести значение, которое требуется найти, в поле *Образец*. Если точное значение неизвестно, можно использовать подстановочные знаки.
* Остальные параметры в окне можно изменить или оставить так, как они установлены по умолчанию:

- значение в поле со списком *Поиск в*, определяющее место поиска, по умолчанию содержит название выбранного столбца;

- значение в поле со списком *Совпадение* определяет один из трех вариантов совпадения образца со значением поля — *С любой частью поля*, *Поля целиком*, *С начала поля*;

- поле со списком *Просмотр*, в котором можно задать направление поиска: *Все*, *Вверх*, *Вниз*;

- флажок *С учетом регистра* позволяет при поиске учитывать начертание букв — прописные или строчные;

- флажок *С учетом формата полей* позволяет выполнять поиск данных в указанном формате отображения.

* Чтобы найти первое вхождение указанного образца, нажмите кнопку *Найти следующий*.
* Для поиска следующих вхождений этого значения нажимайте кнопку *Найти далее* до тех пор, пока не будет найдено нужное вхождение.

**6**. **Выполнить поиск и замену данных по одному из полей таблицы**

* Чтобы выполнить не только поиск, но и изменение данных в найденном поле, необходимо в том же диалоговом окне *Поиск и замена* раскрыть вкладку *Замена* и в поле *Заменить на* указать значение, на которое нужно заменить искомое значение. После этого нажмите кнопку *Заменить* или *Заменить все*.
* Все остальные поля имеют тот же смысл, что и на вкладке *Поиск*. Для замены одного значения необходимо нажать кнопку *Заменить.* Чтобы заменить все вхождения, нажмите кнопку *Заменить все*.
* Установить параметры поиска/замены по умолчанию. В меню *Сервис* выбрать команду *Параметры*.
* В окне *Параметры* открыть вкладку *Правка и поиск*.
* В группе *Поиск и замена по умолчанию* выбрать нужный тип поиска: *поле - целиком*, *везде - любая часть*, *поле - с начала*.
* Нажать кнопку ОК.

**7. Выполнить фильтрацию данных по выделенному фрагменту**

* Открыть таблицу в режиме Таблицы.
* Нажать кнопку *Изменить фильтр* на панели инструментов *Режим таблицы*. Появится форма *фильтр* — специальное окно для изменения фильтра. Форма содержит линейку полей таблицы. В любое из этих полей можно ввести или выбрать из списка значение, которое и будет являться условием отбора. Если условия ввести в несколько полей, они будут объединяться с помощью логического оператора И. Для того чтобы объединить условия по ИЛИ, нужно раскрыть другую вкладку формы, щелкнув по ярлычку Или в нижней части формы.
* Выберите значение из списка, по которому будут отфильтрованы данные.
* Щелкните мышью по ярлычку Или и раскройте вторую вкладку.
* Выберите из списка еще одно значение. Теперь будут записи по двум значениям.
* Нажмите кнопку *Применение фильтра* на панели инструментов.

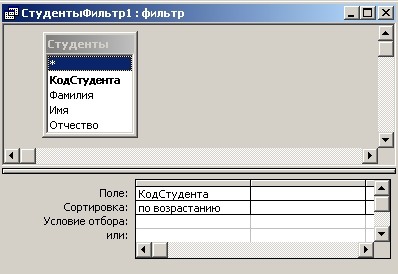
**8. Выполнить экспорт отфильтрованных данных**

Отобранные с помощью фильтра данные можно копировать, экспортировать и рассылать. Копирование данных из выборки осуществляется так же, как копирование данных в таблице.

* Выбрать в меню *Файл* команду *Экспорт.*
* В окне *Экспорт объекта: Таблица <имя таблицы> в* выбрать папку для экспорта файла.
* В поле *Имя файла* ввести имя файла, в который осуществляется экспорт.
* В поле *Тип файла* выбрать тип файла, в который осуществляется экспорт.
* Нажать кнопку *Сохранить все*.

**9. Выполнить фильтрацию данных с помощью расширенного фильтра**

Если возможностей обычной фильтрации не хватает, например, необходимо не только отображать определенные записи, удовлетворяющие тем или иным условиям, но и выполнять сортировку данных по нескольким столбцам, то в этом случае можно использовать **расширенный фильтр**.



* Выполнить команду *Записи | Фильтр | Расширенный фильтр*, после чего отобразится окно расширенного фильтра.
* Выбрать поля, на которые будут накладываться ограничения (Поле), а также направление их сортировки (Сортировка).
* Указать ограничение на содержимое выбранного поля (Условие отбора).

**Пример:**

Необходимо отображать только тех студентов, имена которых начинаются на букву "М", а отчества - на букву "В", причем нужно их отсортировать в алфавитном порядке фамилий, имен и отчеств.

* В первой ячейке параметра Поле выбрать из списка поле Фамилия (или перетащить его в эту ячейку из структуры таблицы в верхней части окна расширенного фильтра).
* Во второй ячейке выбрать поле Имя и установить условие отбора Like "М\*".
* В третьей ячейке параметра Поле выбрать Отчество и указать условие отбора Like "В\*".
* Указать для всех полей режим сортировки по возрастанию. После этого в таблице будут отображены только те записи, которые удовлетворяют указанным условиям, отсортированные в указанном направлении (сначала - сортировка по фамилии, затем - по имени, и после этого - по отчеству).

**Контрольные вопросы:**

1. Как осуществляется сортировка данных в таблице?
2. Каковы основные правила применения сортировки?
3. Как осуществляется алгоритм обычного поиска данных?
4. Как осуществляется алгоритм поиска и замены данных?
5. Что такое фильтрация данных?
6. Какие существуют алгоритмы использования фильтров?
7. Как осуществляется алгоритм экспорта данных из выборки?

**Практическая работа 9. Работа с переменными. Написание программного файла и работа с табличными файлами. Заполнение массива из табличного файла. Заполнение табличного файла из массива**

**Цель:**Изучить механизмы приложения MSAccess для работы с переменными и массивами

**Задание:**

1. Изучить способы создания и работу с переменными и массивами в БД.
2. Ответить на контрольные вопросы.
3. Оформить отчет

**Теоретические сведения:**

**Программирование на языке VBA**

Язык VBA стал общим инструментов для всех приложений MSOffice, позволяющим решать любые задачи программирования, начиная от автоматизации действий конкретного пользователя и заканчивая разработкой полномасштабных приложений, использующих MSOffice как среду разработки.

**Переменные, константы и типы данных**

Как и в других языках программирования, в VBA для хранения временных значений, передачи параметров и проведения вычислений используются переменные. Кратко остановимся на основных особенностях описания и использования переменных в VBA.

Обычно перед тем, как использовать переменную, производится ее объявление, т. е. вы заранее сообщаете Visual Basic, какие именно имена переменных вы будете использовать в своей программе, при этом объявляется также тип данных, для хранения которых предназначена эта переменная. В VBA, как и в обычном языке Basic, для этого используется оператор Dim. Вот синтаксис этого оператора:

Dim <имяПеременной> [Аs<типДанных>]

В VBA действуют следующие правила именования переменных. Имя не может быть длиннее 255 символов, оно должно начинаться с буквы, за которой могут следовать буквы, цифры или символ подчеркивания. Оно не должно содержать пробелов, знаков препинания или специальных символов, за исключением самого последнего знака. В конце к имени переменной может быть добавлен еще один из следующих шести специальных символов — описателей типа данных: ! # $ % & @

Эти символы не являются частью имени переменной: если в программе используются одновременно имена stringl$ и stringl, то они ссылаются на одну и ту же строковую переменную. Нельзя использовать одно и то же имя переменной с разными символами определения типа данных или одновременно явное описание типа данных и не соответствующий этому типу данных специальный символ.

В VBA объявление переменных не является обязательным. Как и в его предшественнике, обычном языке Basic, допускается использование неописанных переменных. Выделение памяти переменным может выполняться динамически, а тип данных, хранящихся в переменной, может определяться по последнему символу имени переменной.

Приведем теперь краткую сводку используемых типов данных VBA:

|  |  |
| --- | --- |
| **Тип данных** | **Описание** |
| Array | Массив переменных, для ссылки на конкретный элемент массива используется индекс. |
| Boolean | Требуемая память: зависит от размеров массива Принимает одно из двух логических значений: True или False. Требуемая память: 2 байта |
| Byte | Число без знака от 0 до 255 Требуемая память: как нетрудно догадаться, 1 байт |
| Currency | Используется для произведения денежных вычислений с фиксированным количеством знаков после десятичной запятой, в тех случаях, когда важно избежать возможных ошибок округления. Диапазон возможных значений: от -922 337 203 685 477,5808 до 922 337 203 685 477,5807. Требуемая память: 8 байтов. Символ определения типа по умолчанию: @ |
| Date | Используется для хранения дат. Диапазон возможных значений: от 1 января 0100 г. до 31 декабря 9999 г. Требуемая память: 8 байтов |
| Double | Числовые значения с плавающей точкой двойной точности. Диапазон возможных значений для отрицательных чисел: от -1 ,797693 13486232Е308 до -4,94065645841 247Е-324. Диапазон возможных значений для положительных чисел: от 4,94065645841 247Е-324 до 1, 7976931 3486232Е308. Требуемая память: 8 байтов. Символ определения типа по умолчанию: # |
| Integer | Короткие целые числовые значения. Диапазон возможных значений: от -32 768 до 32 767. Требуемая память: 2 байта. Символ определения типа по умолчанию: % |
| Long | Длинные целые числовые значения. Диапазон возможных значений: от -2 147 483 648 до 2 147 483 647. Требуемая память: 4 байта. Символ определения типа по умолчанию: & |
| Object | Используется только для хранения ссылок на объекты. Требуемая память: 4 байта |
| Single | Числовые значения с плавающей точкой обычной точности. Диапазон возможных значений для отрицательных чисел: от -3.402823Е38 до -1 ,401 298Е-45. Диапазон возможных значений для положительных чисел: от 1 ,401 298Е-45 до 3.402823Е38. Требуемая память: 4 байта. Символ определения типа по умолчанию: ! |
| String | Используется для хранения строковых значений. Длина строки: от 0 до 64 Кбайтов. Требуемая память: 1 байт на символ. Символ определения типа по умолчанию: $ |
| Variant | Может использоваться для хранения различных типов данных: даты/времени, чисел с плавающей точкой, целых чисел, строк, объектов. Требуемая память: 16 байтов, плюс 1 байт на каждый символ строковых значений. Символ определения типа по умолчанию: отсутствует |

При описании переменной указание типа данных может быть опущено. Тип переменной в таком случае определяется последним символом имени переменной: @, #, %, &, ! или $ (Currency, Double, Integer, Long, Single или String соответственно). Например, поскольку символ "$" является символом определения типа для строковых данных, то переменная под именем text$ автоматически становится переменной типа "строка символов". В дальнейшем этот специальный символ указания типа данных может быть опущен, однако постоянное присутствие в имени переменной символа определения типа будет напоминать о том, к какому типу данных относится эта переменная, что поможет избежать ошибок использования несовместных типов данных.

Если же последний символ не является ни одним из вышеперечисленных и явное указание типа тоже не используется, в этом случае переменной будет назначен по умолчанию тип данных Variant, который позволяет хранить в ней данные любого типа.

Учтите также, что нельзя использовать в одной и той же процедуре имена переменных, отличающиеся друг от друга только специальным символом определения типа в конце переменной. Например, не допускается одновременное использование переменных var$ и var%. He допускается и явное объявление переменной, уже содержащей символ определения типа в конце имени, с помощью описателя As <типПеременной> (даже если такое определение не противоречит обычному применению символа определения типа). Так, например, вы получите сообщение об ошибке, попытавшись ввести любое из следующих определений:

Dim var1% *As* String Dim var2% As Integer

Для определения типа данных аргументов процедуры или функции используется описание типа данных непосредственно в заглавной строке процедуры или функции. Например, следующая заглавная строка процедуры описывает ее параметры как переменные строкового типа: SubSplitStr(str1 AsString, str2 AsString, str3 AsString)

Определение типа данных возвращаемого функцией значения завершает заглавную строку функции, например: FunctionFindSplitSpace (strlAsString) AsInteger

описывает возвращаемое функцией значение как переменную короткого целого типа.

Для описания именованных констант применяется оператор Const, схожий с оператором описания переменных Dim. Вот синтаксис этого оператора:

Const <имяКонстанты> [As <типДанных>] = <выражение>

где <выражение> — эго любое значение или формула, возвращающая значение, которое должно использоваться в качестве константы. Например, следующий оператор определяет целую константу maxLen: Const maxLen% = 30

Как и переменные, константы могут содержать значения различных типов данных, но при этом они не меняют своих значений во время выполнения программы.

**Массивы**

***Массив*** *—* это переменная, в которой хранится одновременно несколько значений одинакового типа. Формальное определение массива таково: он представляет собой совокупность однотипных индексированных переменных.

Количество используемых индексов массива также может быть различным. Чаще всего применяются массивы с одним или двумя индексами, реже — с тремя, еще большее количество индексов встречается крайне редко. В VBA допускается использовать до 60 индексов. О количестве индексов массива обычно говорят как о размерности массива. Массивы с одним индексом называют одномерным, с двумя — двумерными и т. д. Массивы с большим количеством измерений могут занимать очень большие объемы памяти, так что следует быть осторожным в их применении.

Прежде чем использовать массив, нужно обязательно объявить его с помощью оператора Dim и указать при этом тип хранящихся в массиве значений. Все значения в массиве обязаны принадлежать к одному типу данных. Это ограничение на практике можно обойти, использовав при объявлении массива тип Variant — в этом случае элементы массива смогут принимать значения разных типов. Вот синтаксис оператора объявления массива:

Dim <имяМассива> (<размер1>, <размер2>, ...) As <типДанных>

где указанные в скобках величины <размер1>, <размер2> и т.д. задают размеры массива — количество индексов и максимально допустимое значение для каждого. конкретного индекса. При этом индексирование элементов массива по умолчанию начинается с нуля. Так, объявление: Dim Array1 (9) As Integer

определяет одномерный массив из 10 элементов, являющихся переменными целого типа, а объявление: Dim Array2 (4, 9) As Variant

определяет двумерный массив из пятидесяти (5x10) элементов, являющихся переменными универсального типа variant.

При объявлении массива можно указать не только верхнюю границу индекса, но и его нижнюю границу, т. е. явно задать диапазон изменения конкретного индекса массива, причем нижняя граница может быть любым целым числом, необязательно неотрицательным. Вот синтаксис такого определения:

Dim <имяМассива> (<мин1> То <макс1>, ...) As <типДанных>

Например, если вы собираетесь работать с массивом метеорологических данных, представляющих собой средние дневные температуры за последние две недели, то может оказаться весьма удобным дать следующее определение массива:

Dim Temperature (-14 То 0) As Single

При этом, например, Temperature (-2) будет соответствовать позавчерашней температуре, а для определения нужного индекса для интересующего вас дня будет достаточно использовать разность дат.

В приведенных выше примерах речь шла о массивах фиксированного размера, количество элементов в которых явно указано во время описания массива в операторе Dim. Такие массивы называются *статическими.* В VBA допускается использование и *динамических* массивов, размеры которых при описании не фиксируются. Определение размера динамического массива может быть сделано непосредственно во время выполнения программы.

При определении динамического массива в операторе Dim после имени массива стоят лишь пустые скобки и описание типа переменных. Количество индексов и диапазон их изменения не задаются. Однако перед тем, как использовать массив, нужно выполнить оператор ReDim, который задаст размерность и диапазоны изменения индексов динамического массива.

Синтаксис объявления и определения размеров динамического массива таков:

Dim <имяМассива> ( ) As <типДанных>

ReDim <имяМассива> (<размер1>, <размер2>, . . . )

Вот как может выглядеть объявление, определение размеров и использование динамического массива, а затем последующее изменение размерности и размеров этого же массива:

Dim dArray ( ) As Variant

ReDim dArray ( 1 , 2 )

dArray (0, 0) = 2

dArray (0, 1) = 3

k = dArray (0, 0) + dArray (0, 1)

ReDim dArray (k)

dArray (0) = "Строка1"

В этом примере массив dArray сначала определяется как двумерный массив из шести элементов, а затем переопределяется как одномерный массив, причем верхняя граница индекса задается значением переменной k.

**Контрольные вопросы:**

1. Что такое VBA?
2. Какие типы данных существуют в VBA?
3. Что такое массив и с помощью каких операторов он создается?

**Практическая работа 10. Создание меню различных видов. Модификация и управление меню**

**Цель:**Изучить механизмы приложения MSAccess для создания меню различных типов.

**Задание:**

1. Создать меню в приложении Access, задать все свойства для этого меню, назначить клавиши доступа.
2. Описать ход работы.
3. Ответить на контрольные вопросы.
4. Оформить отчет

**Теоретические сведения:**

В Access, так же как и в других приложениях Microsoft Office, используются два типа меню: раскрывающиеся и контекстные. Контекстное меню появляется на экране при щелчке правой кнопкой мыши по объекту, с которым вы хотите работать. Оно содержит перечень действий, которые можно выполнять с данным объектом в данном контексте.

Примером раскрывающегося меню, может служить системное меню окна приложения или документа. Системное меню окна приложения появляется при щелчке левой кнопкой мыши по значку приложения, который находится в левой части заголовка окна приложения, или при нажатии комбинации клавиш <Аlt>+<Пробел>. Системное меню документа открывается после щелчка левой кнопкой мыши по значку документа, который находится в левой части заголовка окна документа, или при нажатии комбинации клавиш <Alt>+<->. Системное меню предназначено для изменения параметров окна.

Строка под заголовком окна приложения содержит главное меню приложения, с помощью которого можно получить доступ к командам меню. Вид главного меню Access 2003 зависит от режима, в котором находится приложение. Например, когда активным окном является окно таблицы, в главном меню появляется команда **Записи**, которая отсутствует в других режимах работы.

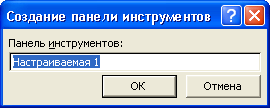
Меню и панели инструментов по умолчанию отображают лишь наиболее часто используемые команды и кнопки. Чтобы получить полный список команд меню, необходимо щелкнуть мышью по двойной стрелке у нижнего края раскрывающегося меню. Если не все кнопки поместились на панели инструментов, то у ее правого края тоже появляется значок двойной стрелки. Если щелкнуть мышью по нему, будут показаны оставшиеся кнопки. Если вы использовали какую-то кнопку или команду, которой не было по умолчанию в соответствующей панели или меню, то в дальнейшем она будет там видна.

Меню и панели инструментов Access являются настраиваемыми, т. е. можно изменять состав кнопок, удаляя ненужные или добавляя новые кнопки

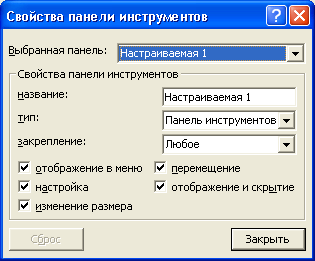
**Ход работы:**

**1. Создать меню в приложении Access**

* Откройте окно ***Настройка***, например, щелкнув правой кнопкой мыши по панели инструментов и выбрав команду ***Настройка*.** Таким способом осуществляется переход в режим настройки панелей команд.
* По умолчанию открывается вкладка ***Панели инструментов***. Если у вас открылась другая вкладка, раскройте именно эту вкладку. Пока в списке на этой вкладке отображаются только стандартные панели команд. К этому списку нужно добавить новую панель команд типа ***Строка меню***. Нажмите кнопку ***Создать****.*
* В диалоговом окне, запрашивающем имя создаваемой панели команд, введите название: ***Главное меню*** (рис. 1) и нажмите кнопку ОК.

 рис.1

* Появится маленькая пустая панель, которая по умолчанию является панелью инструментов. Чтобы превратить ее в строку меню, необходимо изменить ее тип. Для этого выделите элемент Главное меню в списке на вкладке ***Панели инструментов***(он добавляется в конец списка) и нажмите кнопку ***Свойства*.** Откроется диалоговое окно ***Свойствапанели инструментов*** (рис. 2).
* В раскрывающемся списке ***тип*** выберите значение ***Строка меню***.

 рис.2

Кроме типа, для данной панели команд можно задать еще ряд свойств:

*- закрепление*. Можно задать расположение панели команд на экране: в любом месте, только по горизонтали, только по вертикали или вообще запретить его изменять.

*- отображение в меню*. Установите этот флажок, если вы хотите, чтобы название этой панели команд отображалось в списке, появляющемся при выполнении команды меню *Вид, Панели инструментов*.

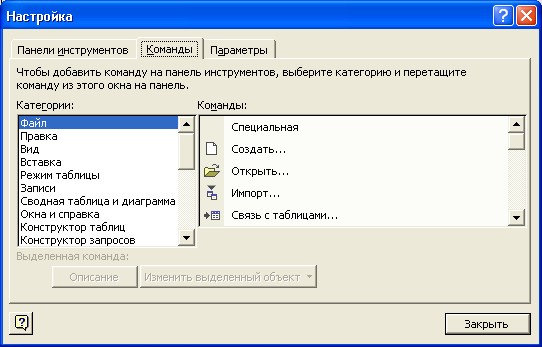
*- настройка*. Сброс этого флажка запретит изменение данной панели команд.

*- изменение размера*. Если этот флажок не установлен, изменение размеров данной панели команд будет не возможно.

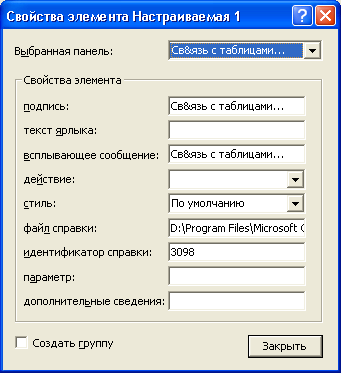
*- перемещение*. Сбросьте этот флажок, если нужно запретить перемещение панели команд по экрану.

*- отображение и скрытие*. Когда этот флажок сброшен, пользователь не будет иметь возможности скрывать и снова отображать панель команд на экране.

* Добавьте меню — списки команд, раскрывающиеся при выборе меню в строке меню или на панели инструментов.
* Раскройте вкладку ***Команды***в диалоговом окне ***Настройка****.*
* Выделите в списке ***Категории***значение ***Новое меню***.
* Перетащите элемент ***Новое меню*** из списка ***Команды***на новую панель.
* Теперь вы увидите на ней элемент ***Новое меню*** (рис. 3).

 рис.3

* Изменить название созданного меню можно в окне свойств. Для этого необходимо щелкнуть правой кнопкой мыши по меню и выбрать в контекстном меню команду ***Свойства.*** Появится диалоговое окно ***Свойства***элемента ***Главное меню*** (рис. 4).

 рис. 4

В этом окне можно задать следующие значения:

- В поле ***подпись***введите название меню.

- В поле ***всплывающее сообщение*** можно ввести текст всплывающей подсказки.

- Поле ***действие*** следует оставить пустым. Обычно оно определяет действие, выполняемое при выборе данного элемента панели команд.

- Поле ***стиль*** позволяет определить вид элемента панели команд, в данном случае — меню.

- Раскрывающийся список имеет следующие значения:

- ***По умолчанию*** — на элементе меню выводятся одновременно значок и надпись;

- ***Только текст (всегда)*** — и на элементе меню и на кнопке выводится только надпись;

- ***Только текст (в меню****)* — выводится надпись на элементе меню и значок на кнопке;

- ***Рисунок и текст*** — и на элементе меню и на кнопке выводится как значок, так и надпись.

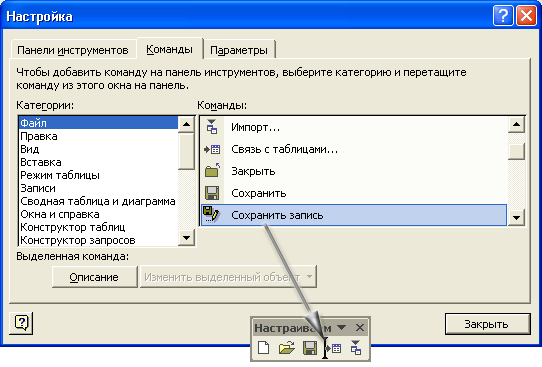
- Два поля — ***файл справки*** и ***идентификатор справки*** — позволяют связать с данным элементом панели команд раздел из справочной системы приложения, который будет отображаться при нажатии клавиши <F1>, если данный элемент выделен.- Поле ***параметр***связано с полем *действие* и определяет значение аргумента функции, если в поле *действие*  определен вызов функции. В данном случае оно тоже остается пустым.

- В поле ***дополнительные сведения*** можно ввести информацию, которая может быть доступна из процедуры VBA.

- Установленный флажок ***Создать группу*** означает, что данный элемент панели команд является началом группы и слева от него (в этом случае в строке меню) будет отображаться разделительная линия.

Закройте окно ***Свойства элемента меню*** и будем добавлять в меню команды. Для этого можно использовать ту же вкладку ***Команды***диалогового окна ***Настройка*.** Выбрав категорию, а затем команду из выбранной категории, можно перетащить ее в данное меню. Например:

* 1. Выберите категорию ***Все формы****.*
  2. Перетащите в меню формы (рис. 5).

 рис.5

* Чтобы добавить в строку меню следующее меню со списком команд, нужно снова выбрать на вкладке ***Команды*** диалогового окна ***Настройка*** категорию ***Новое меню***, затем элемент ***Новое меню*** и перетащить его в данное меню, поместив его справа от первого элемента. Чтобы изменить вид меню или подменю, щелкните по соответствующему элементу правой кнопкой мыши. Можно использовать команды появившегося при этом контекстного меню или открыть окно ***Свойства элемента*.**
* Чтобы окончательно придать меню профессиональный вид (как в стандартном меню Access), необходимо каждому меню и команде меню назначить клавиши доступа. Буквы, соответствующие этим клавишам, выделяются в названии элемента подчеркиванием. Клавиши доступа используются следующим образом. Для того чтобы раскрыть меню, нужно при нажатой клавише <Alt> нажать клавишу, соответствующую подчеркнутой букве в названии этого меню. Например, комбинация клавиш <Alt>+«Ф» (<Alt>+<F>) раскрывает меню ***Файл.*** Далее, чтобы выбрать команду меню, достаточно просто нажать клавишу, соответствующую подчеркнутой букве в названии команды.

**Контрольные вопросы:**

1. Как осуществляется создание меню в приложении MSAccess?
2. Какие свойства можно задать для панели инструментов ***Строка меню***?
3. Какие значения можно задать в диалоговом окне ***Свойства*** элемента ***Главное меню****?*
4. Как придать меню профессиональный вид?

**Практическая работа 11. Создание рабочих и системных окон. Добавление элементов управления рабочим окном**

**Цель:**Изучить механизмы приложения MSAccess для создания рабочих окон и элементов управления рабочим окном

**Задание:**

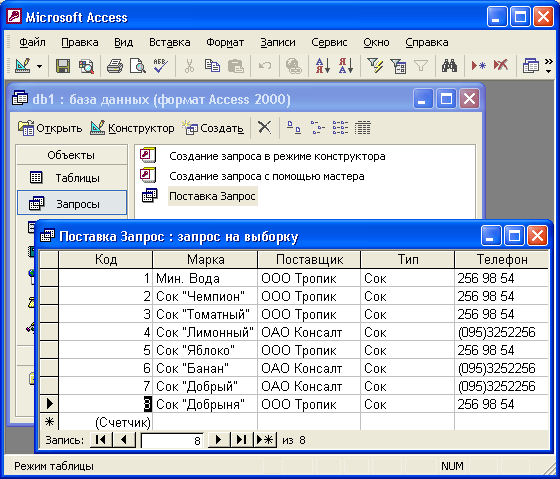
1. Упорядочить объекты в окне базы данных автоматически, по типу, по алфавиту, по дате создания
2. Добавить новую папку на панель объектов.
3. Просмотреть свойства любой таблицы.
4. Описать ход работы.
5. Ответить на контрольные вопросы.
6. Оформить отчет

**Теоретические сведения**

**Рабочая среда Access**

Открыв базу данных Access, пользователь видит на экране главное окно Access и окно базы данных. Главное окно Access имеет вид, аналогичный окнам других приложений Microsoft Office. В верхней части окна размещается его заголовок, меню и панель инструментов, а в нижней части — строка состояния. Наиболее важным элементом интерфейса в среде Access является окно базы данных. Это контейнер, содержащий все объекты базы данных. Доступ к данным объектам возможен только через окно базы данных.

На рис. 1 показано главное окно Access, в котором открыто несколько таблиц в режиме Таблицы. Ниже мы прокомментируем этот рисунок, а заодно определим используемые термины.

 рис.1

**Окна рабочей среды Access**

Каждое запущенное приложение Windows имеет свое окно, в котором выводятся результаты его работы — *окно приложения.* В данном случае окном приложения является главное окно Access. Кроме окна приложения, в Access широко используются еще два типа окон: *диалоговые окна* и *окна документов.* Окна приложения и окна документов могут перемещаться по экрану монитора, имеют изменяемые размеры и могут быть свернуты, развернуты или развернуты до максимальных размеров. Диалоговые окна также можно перемещать по экрану, но они имеют постоянные размеры и их нельзя свернуть или развернуть.

Строка в верхней части окна приложения, содержащая название приложения, является *заголовком* окна приложения. Заголовок окна приложения используется для перемещения окна по экрану (только если окно приложения не свернуто и не развернуто до максимальных размеров). Окно документа, так же как и окно приложения, имеет заголовок, который содержит название этого документа.

*Строка состояния* в нижней части окна приложения или документа служит для отображения подсказок о командах и кнопках и сведений о выполняемых операциях.

Большинство объектов базы данных Access: таблицы, запросы, формы, отчеты, страницы доступа к данным и макросы — выводятся в окнах, которые в терминологии Windows являются окнами документов. Исключение составляют программные модули — для их представления используется специальная среда программирования.

Access поддерживает *интерфейс множественных документов*, который позволяет открыть в рабочем поле окна приложения несколько окон документов одновременно.

На рис. 1 показаны в рабочем поле окна приложения окно базы данных и три окна документов, из которых одно окно развернуто и два — свернуты.

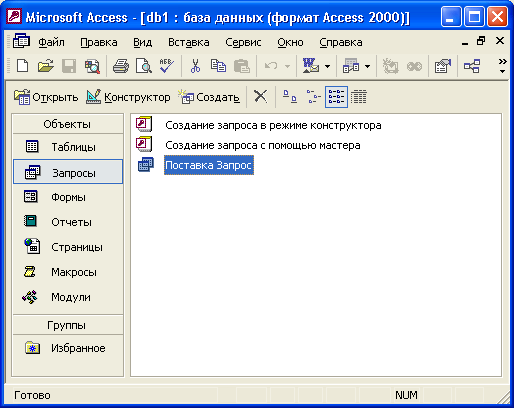
Окна приложения и все окна документов Access по умолчанию отображаются на Панели задач вашей операционной системы. Вы можете активизировать любое из этих окон, щелкнув мышью по соответствующему значку.

*Активным* окном называется окно, в которое направляются все сообщения (действия) от мыши или клавиатуры. Заголовок окна активного приложения или документа подсвечивается цветом, выбранным при установке параметров оформления рабочего стола (обычно это синий цвет). Если окно приложения и окно документа в нем являются активными одновременно, то все сообщения от клавиатуры и мыши получает окно документа.

Одновременно могут быть активными только одно окно приложения и одно окно документа в этом приложении. Все остальные окна считаются *неактивными* (их заголовок выводится на сером фоне). Щелчок левой кнопкой мыши по неактивному окну делает его активным и переводит на передний план. Если нужное окно целиком закрыто другими окнами, то сделать его активным можно, выбрав название документа в меню **Окно**.

**Окно базы данных**

Особым окном в Access является *окно базы данных,* которое позволяет получить доступ ко всем объектам базы данных и выбрать режим работы с объектом. В левой части окна находится панель объектов, которая содержит ярлыки для каждого из *объектов* Access: **Таблицы**, **Запросы**, **Формы**, **Отчеты**, **Страницы**, **Макросы**, **Модули**) (рис. 2).

 рис.2

Щелкнув на ярлыке мышью, в правой части окна вы откроете список соответствующих объектов. Список объектов может быть представлен четырьмя разными способами (обычными для папок операционной системы Windows):

* в виде мелких значков;
* в виде крупных значков (см. рис. 2);
* в виде списка;
* в виде таблицы.

Переключение этих режимов отображения выполняется с помощью четырех правых кнопок на панели инструментов, расположенной вдоль верхнего края окна.

При представлении объектов в виде значков эти значки можно перетаскивать с помощью мыши и располагать внутри окна базы данных любым удобным образом. Если же вы хотите расположить значки так, чтобы они находились рядом друг с другом, нужно:

1. Щелкнуть правой кнопкой мыши на любом свободном участке окна базы данных.
2. Из контекстного меню выбрать команду **Выстроить значки**.

Представление перечня объектов в виде списка не позволяет располагать значки произвольным образом в окне базы данных, но их можно "перетаскивать" за пределы окна базы данных (это один из способов активизировать объект, например открыть таблицу). Представление в виде таблицы позволяет для каждого объекта посмотреть не только его имя, но и описание (столбец **Описание)**, дату и время последнего изменения (столбец **Дата изменения**, дату и время создания (столбец **Дата создания)**, а также тип объекта.

С каждым объектом базы данных можно работать в двух режимах. Первый режим назовем *режимом выполнения* (условно, т. к. он несколько различается для разных типов объектов):

* для таблиц, запросов, форм и страниц доступа к данным этот режим означает открытие соответствующего объекта и называется, соответственно, режим Таблицы (для таблиц и запросов), режим Формы, режим Страницы;
* для отчета — это режим предварительного просмотра;
* для макроса — это действительно режим выполнения;
* для модуля этот режим отключен.

Второй режим — это *режим Конструктора.* Данный режим применяется ко всем типам объектов и предназначен для создания и изменения объектов.

Выбрать нужный режим можно с помощью кнопок, которые находятся в левой части панели инструментов окна базы данных, с помощью команд меню **Вид** или с помощью кнопок на панели инструментов Access.

На панели инструментов окна **База данных** слева имеются три основных кнопки:

* первая кнопка имеет изменяющееся название, в зависимости от того, какой объект выбран. Если выбраны таблица, запрос, форма, страница или группа Избранное, кнопка приобретает вид **Открыть**. Если выбран отчет — **Просмотреть,** если макрос или модуль — **Запустить**. При этом значок рядом с названием соответственно изменяется. Название этой кнопки в каждом варианте наглядно отражает назначение;
* вторая кнопка **Конструктор** постоянна для всех объектов и предназначена для редактирования ранее созданного объекта;
* третья кнопка **Создать** имеет постоянное название, но при этом изменяются значки на ней, в соответствии с типом объекта.

**Ход работы:**

**1. Упорядочить объекты в окне базы данных по типу.**

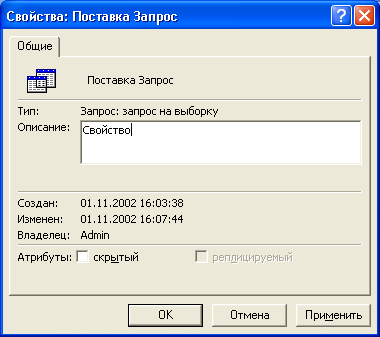
* Щелкнуть правой кнопкой мыши на любом свободном участке окна базы данных.
* Из контекстного меню выбрать команду **Упорядочить значки**.
* Из раскрывающегося меню выбрать способ упорядочения: **по типу**.

**2**. **Упорядочить объекты в окне базы данных автоматически.**

* Из контекстного меню выбрать команду **Упорядочить значки**.
* В раскрывающемся меню выбрать команду **автоматически**. Перед командой появится метка, свидетельствующая о том, что режим автоматического упорядочения включен.
* Чтобы отменить автоматическое упорядочение, необходимо снова выбрать команду **автоматически** в меню **Упорядочить значки**. Метка команды будет удалена, а режим автоматического упорядочения выключен.

**3. Просмотреть свойства любой таблицы.**

* щелкнуть правой кнопкой мыши на имени объекта и из контекстного меню выбрать команду **Свойства**;
* выделить объект из списка в окне базы данных и выбрать команду **Вид, Свойства** из главного меню Access.

Рис. 3

На рис. 3 показано окно свойств таблицы. В нем отражается следующая информация:

* **Тип** — тип объекта (в данном случае Таблица);
* **Описание**  — описание таблицы, определяемое пользователем;
* **Создан**  — дата создания таблицы;
* **Изменен**  — дата последнего изменения таблицы;
* **Владелец**  — владелец (создатель) таблицы;
* **Атрибуты**: **Скрытый**  — позволяет скрыть таблицу из окна базы данных, **Реплицируемый**  — позволяет управлять реплицируемостью объекта

**4. Добавить новую папку на панель объектов.**

На панели объектов можно размещать также папки, которые содержат ярлыки к различным объектам баз данных. Таким образом, можно объединять разные типы объектов в группы. По умолчанию в этой части панели объектов размещается одна папка — **Избранное**. Щелкнув мышью по папке, можно увидеть список объектов, входящих в данную группу.

* Щелкнуть правой кнопкой мыши на панели объектов и выбрать из контекстного меню команду **Новая группа**.
* В диалоговом окне **Новая группа** ввести имя создаваемой папки и нажать кнопку **ОК.**

**Контрольные вопросы:**

1. Что такое *окно базы данных*?
2. Как упорядочить объекты в окне базы данных?
3. Какие кнопки расположены на панели инструментов окна *База данных?*
4. Что показывает окно свойств таблицы?
5. Как добавить новую папку на панель объектов?

**Практическая работа 12. Создание файла проекта базы данных. Создание интерфейса входной формы. Использование исполняемого файла проекта БД, приемы создания и управления**

**Цель:**Изучить механизмы приложения MSAccess для создания файла проекта базы данных, а так же ее интерфейса.

**Задание:**

1. Создать формы для базы данных двумя способами: с помощью средства автоматического создания форм и с помощью Мастера форм.
2. Добавить в одну из таблиц 10 записей с помощью формы
3. Запретить добавление записей в таблицу с помощью формы
4. Описать ход работы.
5. Ответить на контрольные вопросы.
6. Оформить отчет

**Теоретические сведения:**

**Создание форм для ввода данных**

Access позволяет организовать удобный и интуитивно понятный интерфейс пользователя для работы с данными с помощью *форм.* Формами называются настраиваемые диалоговые окна, сохраняемые в базе данных в виде объектов специального типа. Формы Access являются объектами базы данных, так же как таблицы и запросы. Формы используются в приложении для ввода и отображения данных. Формами можно управлять программно с помощью процедур на VBA.

Формы предоставляют более удобный способ просмотра и правки данных в таблицах, чем режим/Таблицы. Формы содержат так называемые *элементы управления,* с помощью которых осуществляется доступ к данным в таблицах. Элементами управления являются текстовые поля для ввода и правки данных, кнопки, флажки, переключатели, списки, надписи. Создание форм, содержащих необходимые элементы управления, существенно упрощает процесс ввода данных и позволяет предотвратить ошибки.

Формы Access предоставляют функциональные возможности для выполнения многих задач, которые нельзя выполнить другими средствами. Формы позволяют выполнять проверку корректности данных при вводе, проводить вычисления, и обеспечивают доступ к данным в связанных таблицах с помощью подчиненных форм.

**Автоматическое создание формы на основе таблицы или запроса**

Access предлагает несколько способов создания форм. Самым простым из них является использование средств автоматического создания форм на основе таблицы или запроса. Автоматически создаваемые формы *(автоформы)* бывают нескольких видов, каждый из которых отличается способом отображения данных.

* Автоформа, организованная *"в столбец"*. В такой форме поля каждой записи отображаются в виде набора элементов управления, расположенных в один или несколько столбцов. Это компактное и, пожалуй, самое удачное представление для быстрого создания формы.
* *Табличная*. Форма будет выглядеть так же, как обычная таблица Access.
* *Ленточная.* В такой форме поля каждой записи располагаются в отдельной строке. Это очень удобно для работы с большими массивами данных, поскольку данные располагаются в таком же порядке, как в простой таблице. Преимуществом именно этого представления формы по сравнению с табличным является то, что каждое поле представлено в виде отдельного элемента управления, которое можно оформить в любом стиле по вашему вкусу и для которого можно определить функции обработки событий, т. е. "оживить" поле с помощью программирования.

**Создание формы с помощью мастера**

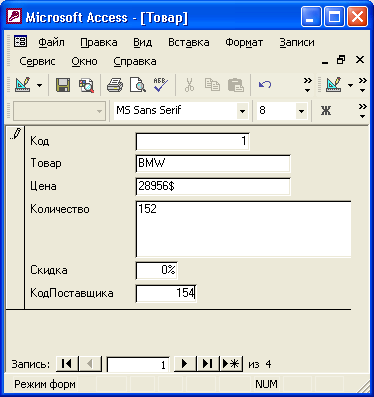
Другим простым и быстрым способом создания формы является использование Мастера форм. С помощью мастера можно создавать формы на основе одной таблицы и более сложные формы на основе нескольких таблиц и запросов, имеющие подчиненные формы. Намного проще и быстрее создавать формы с помощью мастера, а затем усовершенствовать их в режиме Конструктора. Поэтому Мастер форм полезен не только начинающим пользователям, но и профессиональным разработчикам.

**Режимы работы с формами**

Работа с формами Access может происходить в пяти режимах: в режиме Формы, в режиме Таблицы, в режиме Конструктора, в режиме Сводной таблицы и в режиме Сводной диаграммы. Выбрать режим работы можно либо с помощью кнопки **Вид** на панели инструментов текущего режима работы с формой, либо с помощью соответствующей команды меню **Вид**.

*Режим Формы* является "рабочим" для пользователя базы данных. В этом режиме осуществляются просмотр и редактирование записей, удаление записей или добавление новых. В этом же режиме по умолчанию открывается форма из окна базы данных. Если форма была открыта в другом режиме, то для перехода в режим Формы выберите команду **Вид, Режим формы** или нажмите на стрелку, расположенную справа от кнопки **Вид** на панели инструментов и в открывшемся списке выберите элемент

На рис. 1 показана форма, открытая в режиме Формы.

 рис.1

**Режим Конструктора**

Разработку структуры формы, того, как она будет выглядеть в режиме Таблицы и режиме Формы, можно выполнить только в *режиме Конструктора.* В этом режиме можно изменить источник данных для формы, количество отображаемых полей, внешний вид формы и элементов управления, добавить или удалить элементы управления, настроить их свойства.

Панель инструментов **Панель элементов**, отображаемая в режиме Конструктора, позволяет создавать элементы управления, при этом некоторые элементы управления можно создавать с помощью Мастера элементов, осуществляющего пошаговое руководство этим процессом.

Чтобы создать пустую форму, не пользуясь автоматическим созданием форм и Мастером форм, в окне базы данных щелкните по ярлыку **Формы**, нажмите кнопку **Создать** на панели в окне базы данных, в появившемся диалоговом окне **Новая форма** выделите элемент **Конструктор** и нажмите кнопку **ОК.** То же самое можно сделать, просто дважды щелкнув по ярлыку **Создание формы в режиме конструктора**, находящемуся перед списком существующих форм в базе данных.

**Режим Формы**

Добавление, удаление и редактирование записей с помощью формы происходит при работе с ней в *режиме Формы* или в *режиме Таблицы.* Все же режим Формы более функционален по сравнению с режимом Таблицы. Работа с формой в режиме Таблицы ничем не отличается от работы с обычными таблицами Access*.* А вот в режиме Формы представление и автоматическая обработка данных отличаются более широкими возможностями.

**Добавление записи**

В формах, так же как и в таблицах, предусмотрена *пустая запись,* которая предназначена для добавления новых записей в таблицу. Эта пустая запись, как и в таблице, отображается в форме после всех заполненных записей. К этой записи можно перейти, пролистав в форме все записи с помощью кнопок перехода по записям или с помощью специальной кнопки перехода на новую запись. После ввода данных в эту запись и ее сохранения данные автоматически попадают в таблицу.

При добавлении новой записи в таблицу с помощью формы необходимо, переместившись на пустую запись, заполнить поля этой формы в соответствии с правилами, определенными разработчиком формы. Перемещение между полями формы, используемыми для ввода и редактирования данных, происходит аналогично тому, как это делается в запросах и таблицах, за исключением того, что клавиши перемещения курсора вверх и вниз перемещают курсор между полями, а не между записями. Для сохранения введенных значений нажмите комбинацию клавиш <Shift>+<Enter>.

**Ход работы:**

**1. Создать форму с помощью средства автоматического создания форм:**

* Щелкните по ярлыку **Формы в** окне **База данных** и нажмите кнопку **Создать**. Появится диалоговое окно **Новая форма**.
* В списке диалогового окна **Новая форма** выделите один из вариантов автоформы, например: **Автоформа: в столбец**.
* В поле со списком, находящимся в нижней части диалогового окна **Новая форма,** содержатся имена всех таблиц и запросов базы данных, которые могут быть использованы в качестве источника данных для формы. Щелкните левой кнопкой мыши по кнопке со стрелкой, чтобы раскрыть список, и выберите в нем нужный элемент.
* Нажмите кнопку **ОК.** В результате будет автоматически создана и открыта форма выбранного вида.
* Чтобы созданную форму можно было использовать в дальнейшем, ее необходимо сохранить. Для сохранения формы выберите команду **Файл, Сохранить** или нажмите на кнопку **Сохранить н**а панели инструментов **Режим формы**.
* В поле **Имя формы** появившегося диалогового окна **Сохранение** введите нужное название и нажмите кнопку **ОК.**

**2. Создать форму с помощью Мастера форм:**

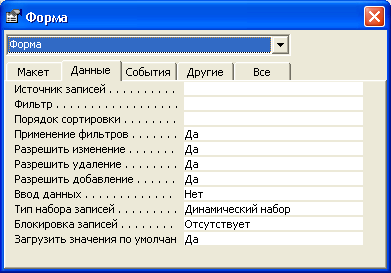
* Нажмите кнопку **Создать** на панели инструментов окна базы данных. В списке вариантов в появившемся диалоговом окне **Новая форма** выделите элемент **Мастер форм** и нажмите кнопку ОК. То же самое можно сделать, дважды щелкнув по ярлыку **Создание формы с помощью мастера**, находящемуся перед списком существующих форм в базе данных.
* Появится первое диалоговое окно Мастера форм. В поле со списком **Таблицы и запросы**, как и в раскрывающемся списке в окне **Новая форма,** будут отображены имена всех таблиц и запросов базы данных, которые могут использоваться в качестве источника данных для формы. Раскройте этот список и выберите имя таблицы или запроса.
* В списке **Доступные поля** этого диалогового окна отображаются все поля выбранной таблицы или запроса. Добавление полей в форму позволит просматривать и редактировать данные выбранной таблицы. Чтобы добавить в создаваемую форму только некоторые поля, выделите каждое из этих полей и нажмите кнопку ">". Выделенное поле будет перемещено из списка **Доступные поля** в список **Выбранные поля**. Чтобы добавить в создаваемую форму сразу все поля из выбранной таблицы или запроса, нажмите кнопку ">>". Нажмите кнопку **Далее** для отображения второго диалогового окна Мастера форм.
* Во втором диалоговом окне мастера можно определить вид формы.
* Третье диалоговое окно Мастера форм предназначено для выбора стиля оформления новой формы. Мастер предлагает несколько стандартных стилей оформления. Можно определить собственные стили оформления форм с помощью диалогового окна **Автоформат**. Тогда эти стили будут отображаться вместе со стандартными в этом диалоговом окне Мастера форм. Выберите один из предлагаемых стилей и нажмите кнопку **Далее***.*
* В последнем диалоговом окне Мастера форм требуется указать название формы. В поле ввода этого диалогового окна введите название формы. Чтобы отобразить созданную мастером форму в режиме Формы, выберите переключатель **Открыть форму для просмотра и ввода данных**. А если после автоматического создания формы с помощью мастера требуется внести собственные изменения, выберите переключатель **Изменить макет формы**, тогда созданная форма будет открыта в режиме Конструктора.

**3. Добавить в одну из таблиц 10 записей с помощью формы**

* Откройте форму. Перейдите в режим Формы, если форма была открыта в другом режиме. Для этого нажмите кнопку **Вид** панели инструментов текущего режима и выберите из списка режимов элемент **Режим формы**. В текстовых полях формы появятся значения полей первой записи таблицы или запроса, являющегося источником данных формы. Access помещает точку вставки в первое поле формы.
* Перейдите к пустой записи, предназначенной для ввода новых данных. Для этого нажмите кнопку **Новая запись** на панели инструментов **Режим формы,** либо маленькую кнопку **Новая запись**, расположенную в нижней части окна формы.
* При переходе на новую запись все текстовые поля и другие элементы управления в форме станут пустыми, кроме тех, значение которых определено по умолчанию. Access помещает точку вставки в первое поле формы. Для перемещения точки вставки к следующему полю нажмите клавишу <Enter> или клавишу <Таb>. Чтобы вернуться к предыдущему полю, нажмите комбинацию клавиш <-Shift>+<Tab> или используйте клавиши управления курсором для перемещения между полями. Можно, конечно, переходить от поля к полю с помощью одинарного щелчка левой кнопкой по нужному элементу управления.
* В поля формы введите требуемые значения, используя стандартные методы редактирования. В текстовые поля введите нужные значения, а в раскрывающихся списках или полях со списками с помощью мыши выберите необходимые элементы, так же как это делается в стандартных диалоговых окнах Windows.
* После завершения ввода данных Access сохранит их в памяти, но не добавит новую запись в таблицу. Для добавления записи в таблицу нажмите комбинацию клавиш <Shift>+<Enter> или выберите команду **Записи, Сохранить запись**.
* Для добавления нескольких записей повторите шаги с 2 по 5.

**4. Запретить добавление записей в таблицу с помощью формы**

* Откройте форму в режиме Конструктора.
* Выделите всю форму.
* На панели инструментов **Конструктор форм** нажмите кнопку **Свойства**.
* В появившемся окне свойств формы (рис. 2) раскройте вкладку Данные.

 рис.2

* На этой вкладке в раскрывающемся списке свойства **Разрешить добавление** выберите элемент **Нет**.
* Закройте окно свойств и сохраните форму. Теперь кнопка **Новая запись**, как на панели инструментов **Режим формы,** так и внизу окна формы, будет заблокирована, и добавление новых записей в таблицу с помощью формы будет невозможно.

**Контрольные вопросы:**

1. Что такое форма?
2. Какие существуют виды автоформ?
3. Как создать форму с помощью средства автоматического создания форм?
4. Как осуществляется алгоритм создания форм с помощью Мастера?
5. Какие существуют режимы работы с формами?
6. Как добавить новую запись в таблицу с помощью формы?

**Практическая работа 13. Создание формы. Управление внешним видом формы**

**Цель занятия.** Изучение информационной технологии создания пользовательских форм для ввода данных с использованием Мастера форм и Конструктора в СУБД MSAccess.

***Задание 1.*** Создать автоформу в столбец по таблице «Мои Расходы»

**Порядок работы**

1. Запустите программу СУБД Microsoft Access и откройте свою созданную базу данных.

2. Выберите объект базы — Формы. Нажмите кнопку Создать, в открывшемся окне Новая форма выберите способ создания формы: «Автоформа: в столбец»; в качестве источника данных укажите таб. «Мои расходы». Сохраните созданную форму с именем — «Мои расходы»

3. Введите две новых записи с истюльзованием формы «Мои расходы»

4. Сохраните созданную форму.

***Задание 2.*** Создать форму с помощью Мастера форм на основе таблицы «Культурная программа»

**Порядок работы**

1. Для создания формы *Мастером форм* выберите объект базы - *Формы*. Нажмите кнопку Создать, в открывшемся окне *Новая Форма* выберите способ создания формы — «*Мастер форм*»; в качестве источника данных укажите таблицу «*Культурная прогиамма*»

2. Выберите:

поля — дата мероприятия, Приглашенные, домашний телефон (для выбора полей используйте кнопки *Выбор одного/ всех полей* между окнами выбора);

внешний вид формы — в один столбец;

сталь — официальный;

имя формы — «Культурная программа»

3. Перейдите в режим Формы (*Вид/Режим формы*) и добавьте несколько записей. Для перехода по записям и создания новой записи используйте кнопки в нижней части окна.

4. *Мастером форм* на основе всех полей таблицы «Культурная программа» создайте форму «Культурная программа2». Сравните внешний вид созданной формы с формой «Культурная программа». Введите пять записей, пользуясь формой «Культурная программа 2»

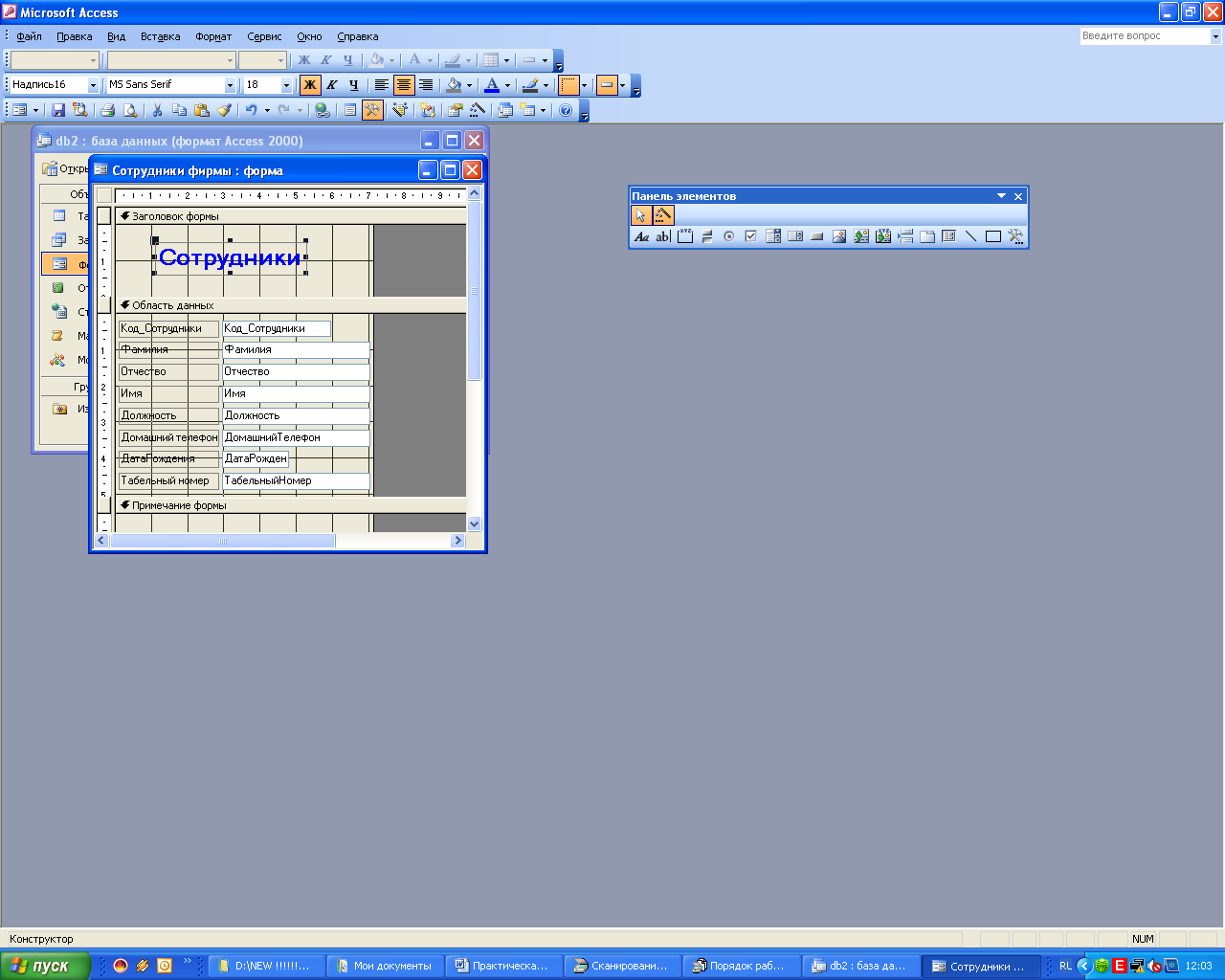
***Задание 3.*** Мастером форм создайте новую форму «Сотрудники фирмы» со всеми полями таблицы «Сотрудники фирмы». Отредактируйте форму в режиме Конструктор.

**Порядок работы**

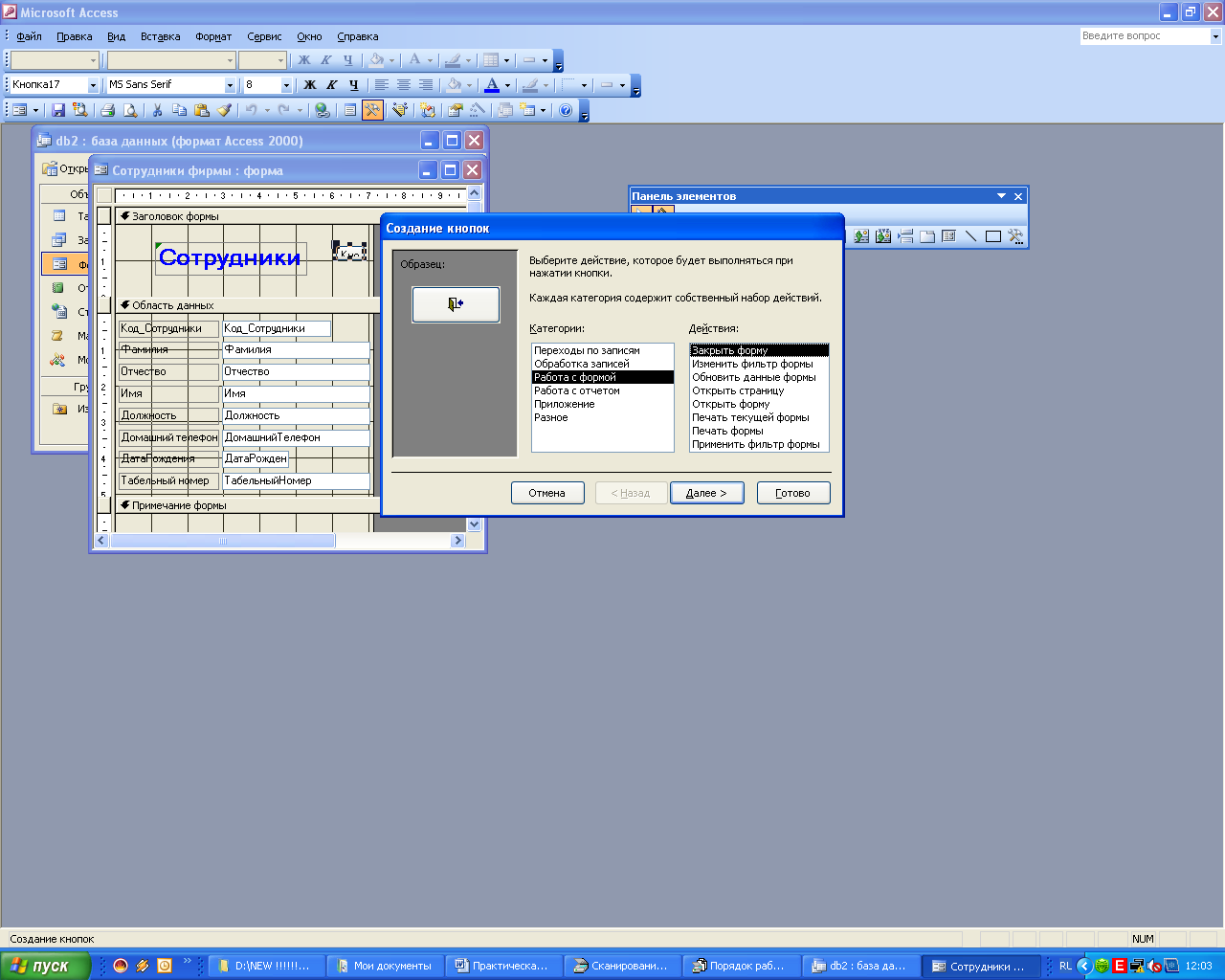
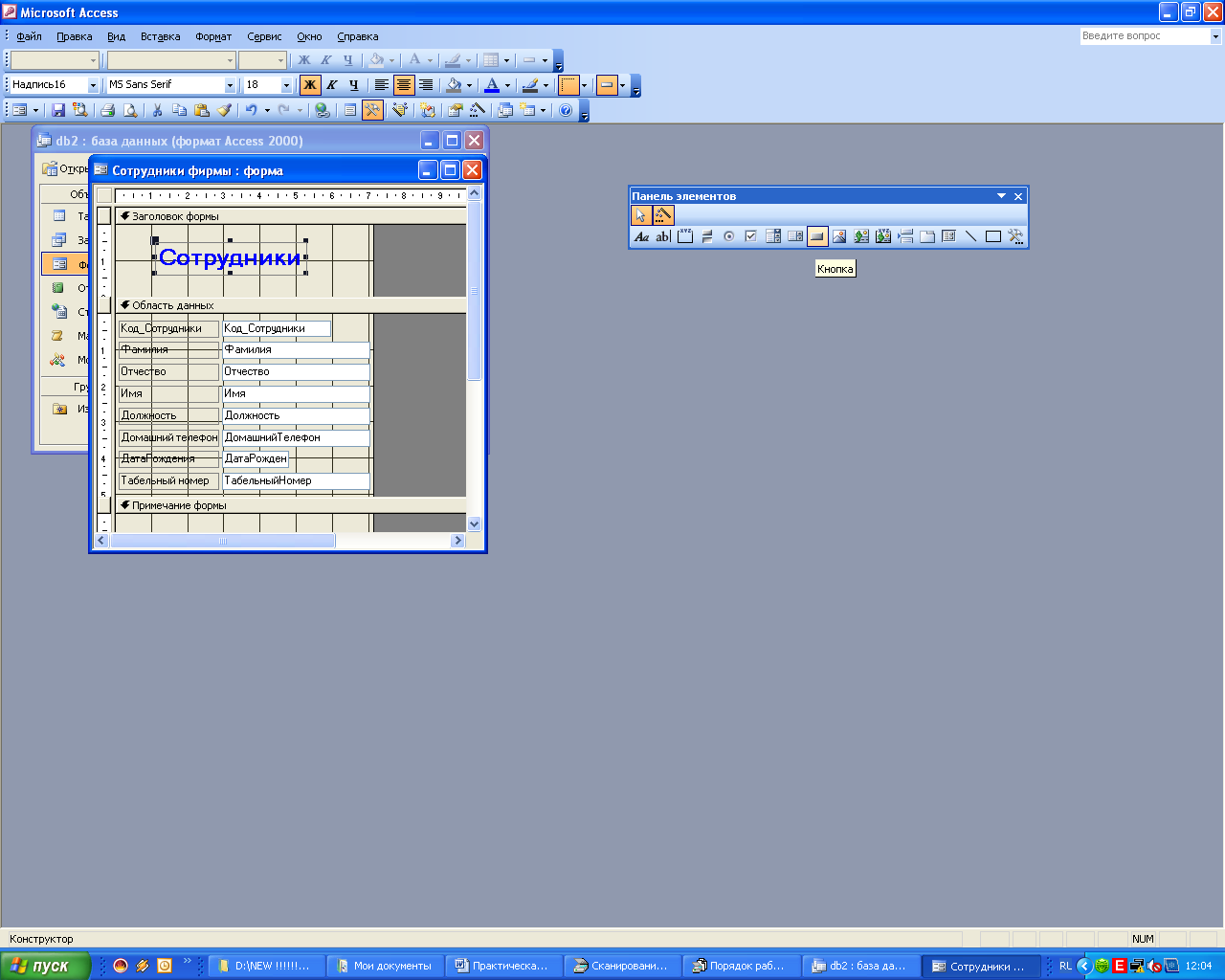
1. Мастером форм создайте новую форму «Сотрудники фирмы» со всеми полями таблицы «Сотрудники фирмы»

2. Откройте форму «Сотрудники фирмы», перейдите в режим Конструктор (*Вид/Конструктор*). Добавьте к форме Заголовок и Примечание (*Вид/Заголовок/Примечание формы*). Раздвиньте область заголовка примерно на два сантиметра и, пользуясь кнопкой Надпись (*Аа*) панели элементов создайте в области заголовка название Форм — «Сотрудники». Параметры заголовка — полужир шрифт, размер — 14, цвет — синий.

3. Рядом с надписью «*Сотрудники*» создайте кнопку для закрытия. Для этого активизируйте на панели элементов кнопку мастер элементов, а затем используйте инструмент *«Кнопка*». После переноса кнопки курсором мыши в нужное место формы и вычеркивание ее рамки запуститься мастер *Создание кнопок*. В окне мастера нужно выбрать действие, которое будет выполняться при нажатии кнопки. В группе «категория» выберите «Работа с формой», в группе «Действия» выберите категорию «Закрыть форму»

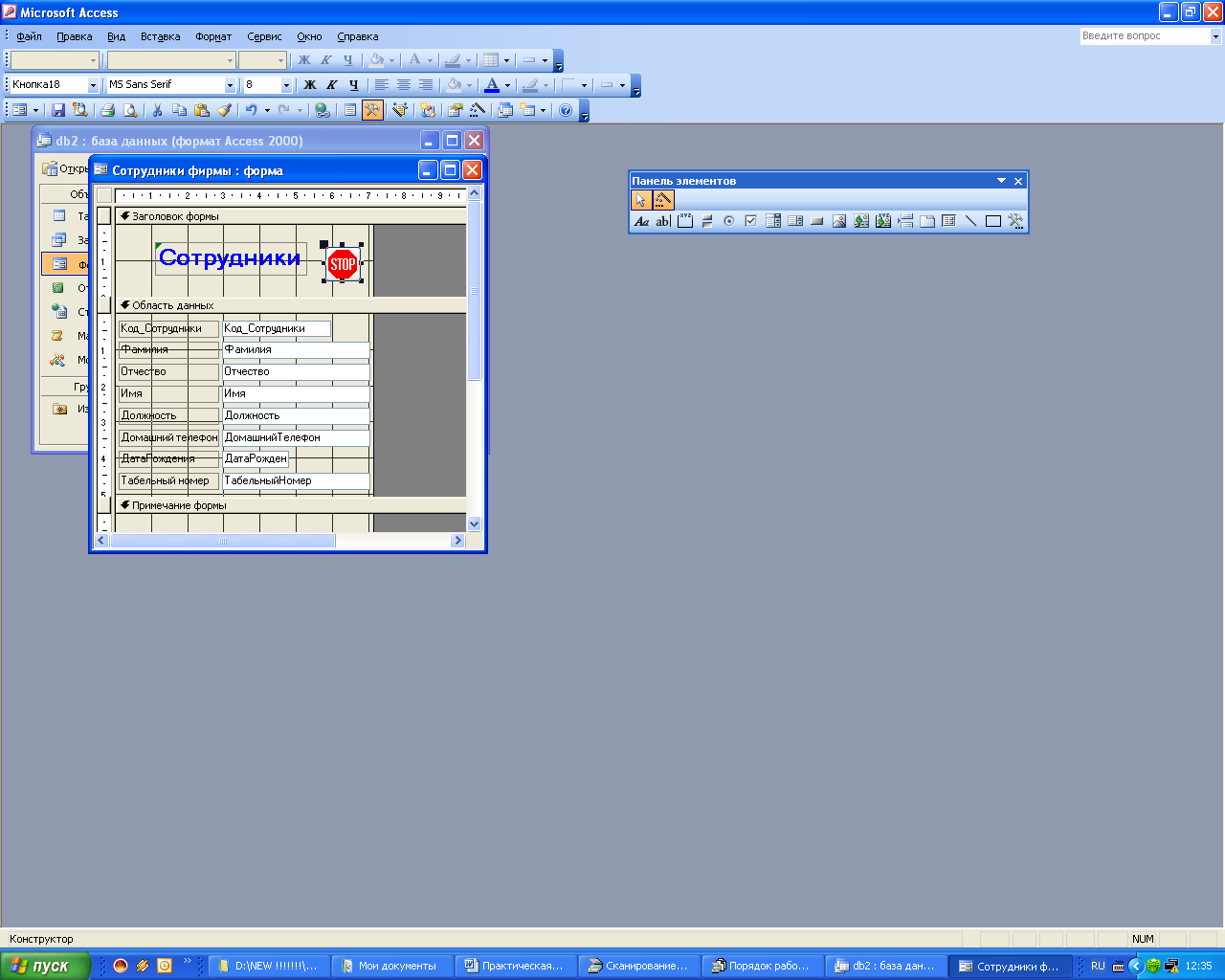
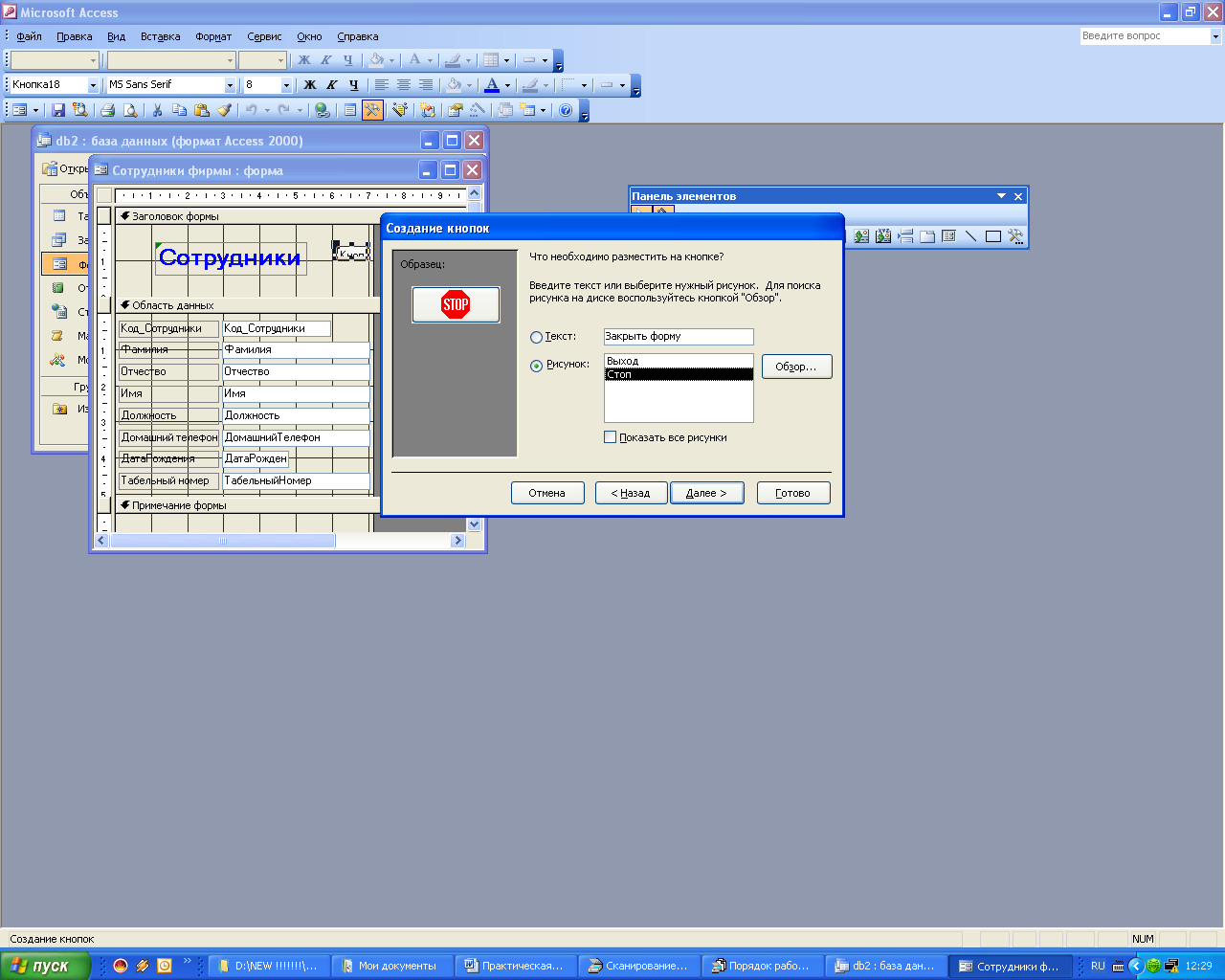


Создание в Конструкторе заголовка формы «Сотрудники»



. Создание кнопки Создание кнопки в заголовке формы «Сотрудники»

4.Вследуюцюм сеанссе диалога с мастером опрсделяется вид кнопки — «Текст» или «Рисунок» (выбираем «Рисунок») и выбирается подходящий рисунок из списка. После нажатия кнопки *Готово* мастер встраивает кнопку в нужное место на форме. Примерный конечный вид формы приведен на рис. Аналогичные действия выполняются при встраивании кнопок формы.



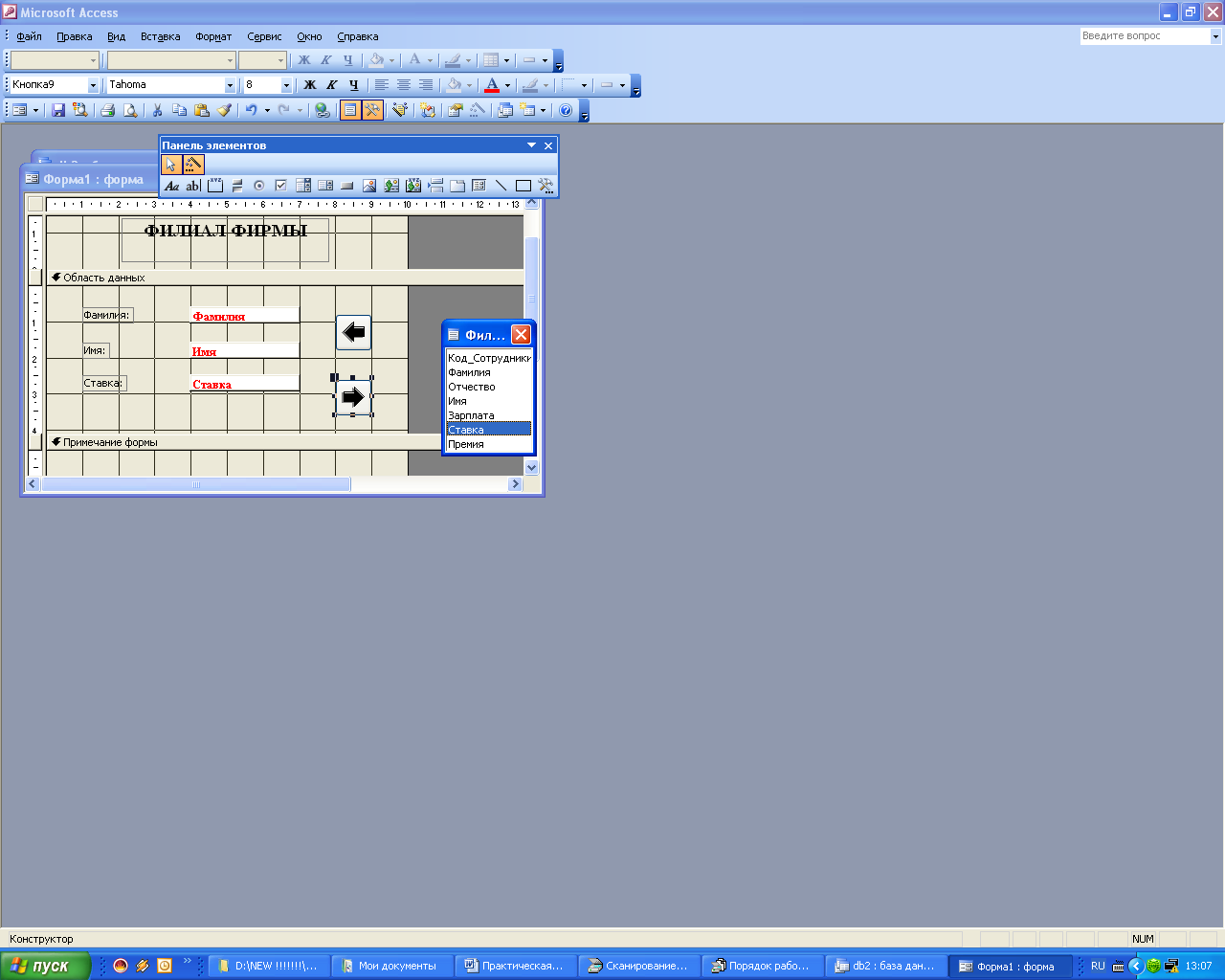
Определение вида кнопки при создании кнопки Конечный вид формы «Сотрудники фирмы»

***Задание 4.*** Создать форму с помощью Конструктора наоснове таблицы «филиал фирмы»

**Порядок работы**

1. Для создания формы выберите объект базы— *Формы*. Нажмите кнопку *Создать*, в открывшемся окне Новая форма выберите способ создания формы — «*Конструктор*»; в качестве источника данных укажите таблицу «*Филиал фирмы*»

2. В «Область данных» включите поля *Фамилия, Имя, Ставка* перетаскиванием каждого поля из «Списка полей» (располагайте поля между 4 и 5 см по горизонтальной линейке). для изменения размеров и перемещения полей по листу используйте маркеры.



Создание в конструкторе формы «Филиал фирмы»

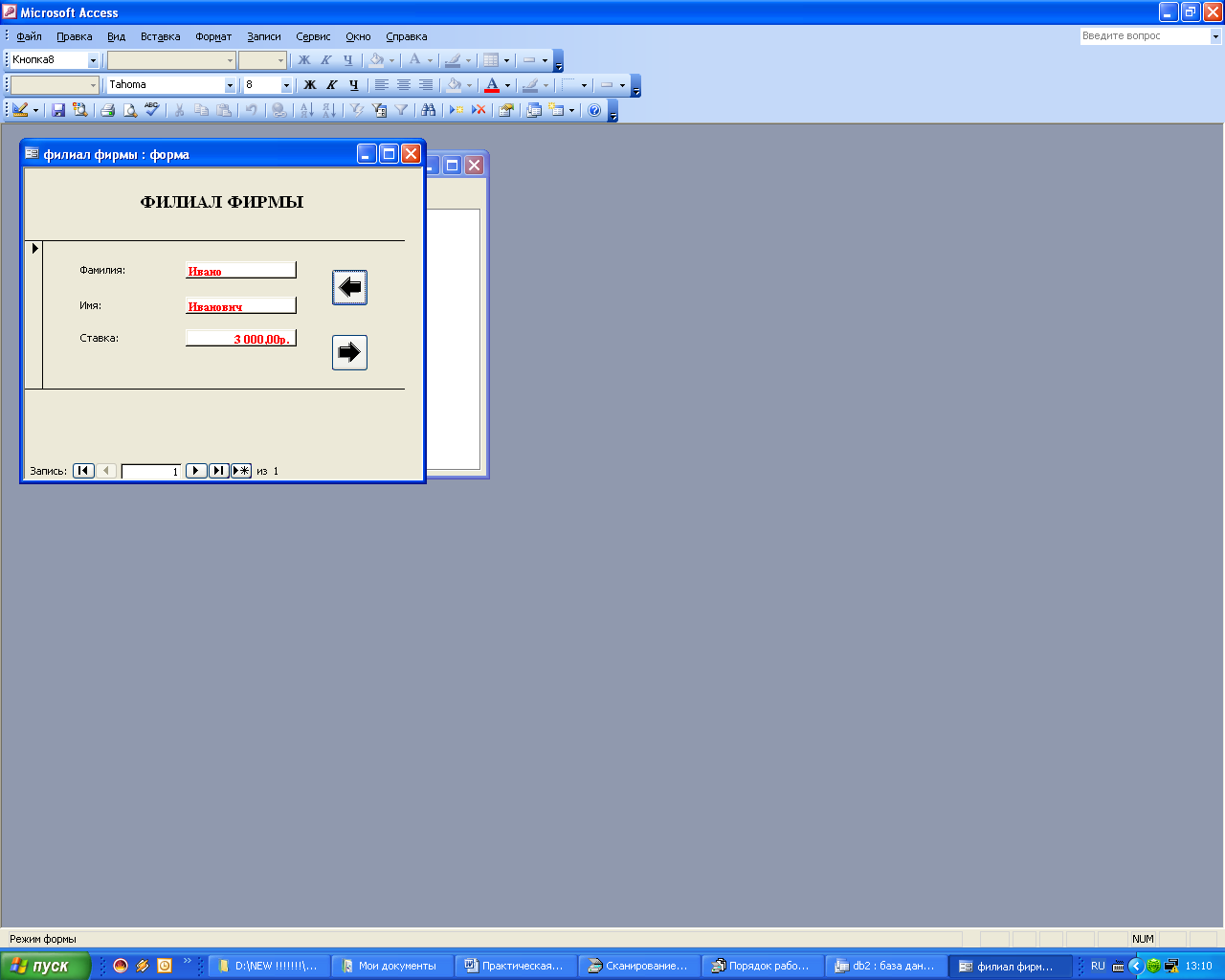
3. Выполните форматирование формы, используя соответвующие кнопки панели форматирования или команды контекстного меню, вызываемого правой кнопкой мыши:

* произведите выравнивание полей и надписей;
* измените шрифт наименования полей на Times New Roman, размер 10, начертание — полужирный курсив;
* задайте следующее оформление формы: цвет фона формы — светло-зеленый; цвет текста — темно-зеленый; выравнивание текста — по центру; цвет фона поля — желтый; цвет границы — черный; толщина границы линии — 2; оформление — с тенью.

4. В область «*Заголовок формы*» введите надпись «*Филиал фирмы*», используя кнопку Надпись (*Аа*) панели элементов.

5. В «*Область данных*» введите две кнопки категории «*Переходы по записям*» Предыдущая запись (верхняя стрелка) и Следующая Запись (нижняя стрелка).

6. Сохраните созданную форму.



Примерные конечный вид формы «Филиал фирмы»

**Практическая работа 14. Задание значений и ограничений поля. Проверка введенного в поле значения. Отображение данных числового типа и типа дата**

**Цель работы:** овладение практическими навыками обработки табличных данных

**Вставка, удаление и обновление данных**

После создания БД и таблиц перед разработчиком встает задача заполнения таблиц данными. В реляционных БД традиционно применяют три подхода:

* однострочный оператор INSERT- добавляет в таблицу новую запись;
* многострочный оператор INSERT- добавляет в таблицу несколько записей;
* пакетная загрузка LOADDATAINFILE- добавление данных из файла.

**Вставка данных с помощью оператора**INSERT.Однострочный оператор INSERTможет использоваться в нескольких формах. Упрощенный синтаксис первой формы:

INSERT [IGNORE] [INTO] имя\_таблицы [(имя\_столбца, ... )]

VALUES(выражение, ... );

Оператор вставляет новую запись в таблицу имя таблицы. Значения полей записи перечисляются в списке (выражение, ... ). Порядок следования столбцов задается списком (имя столбца, ... ). Список столбцов (имя столбца, ... ) позволяет менять порядок следования столбцов при добавлении.

Первичный ключ таблицы является уникальным, и попытка добавить уже существующее значение приведет к ошибке. Чтобы новые записи с дублирующим ключом отбрасывались без генерации ошибки, следует добавить после оператора INSERTключевое слово IGNORE.

Другая форма оператора INSERTпредполагает использование слова SET:

INSERT [IGNORE] [INTO] имя таблицы

SETимя\_столбца1 = выражение!, имя\_столбца2 = выражение2, ... ;

Оператор заносит в таблицу имя\_таблицы новую запись, столбец имя\_столбца в которой получает значение выражение.

Многострочный оператор INSERTсовпадает по форме с однострочным оператором, но после ключевого слова VALUESдобавляется через запятую несколько списков (выражение, ... ).

Практические примеры использования оператора INSERTдля заполнения учебной БД bookсм. ниже, в пункте «Пример выполнения работы».

**Удаление данных.** Для удаления записей из таблиц предусмотрены:

* оператор DELETE;
* оператор TRUNCATE TABLE.

Оператор DELETEимеет следующий синтаксис:

DELETE FROM имя\_таблицы[WHERE условие]

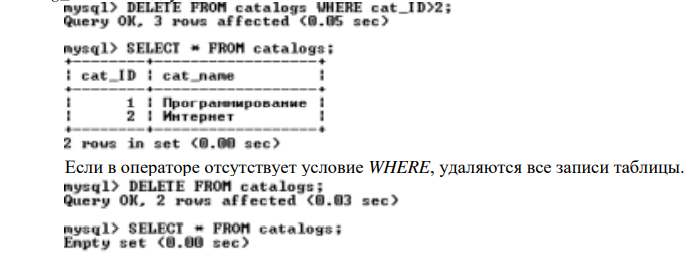
[ORDERBYимя поля]

[LIMITчисло строк];

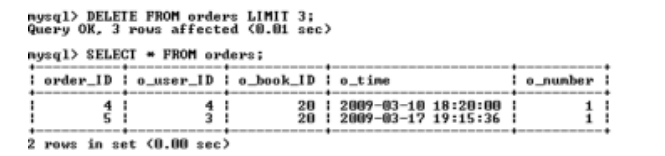
Оператор удаляет из таблицы имя\_таблицы записи, удовлетворяющие условию. В

следующем примере из таблицы catalogsудаляются записи, имеющие значение первичного ключа

catalogedбольше двух.



Ограничение LIMIT позволяет задать максимальное число записей, которые могут быть удалены. Следующий запрос удаляет все записи таблицы orders, но не более 3 записей.



Конструкция *ORDERBY*обычно применяется вместе с ключевым словом *LIMIT.* Например, если необходимо удалить 20 первых записей таблицы, то производится сортировка по полю типа *DATETIME*- тогда в первую очередь будут удалены самые старые записи.

Оператор *TRUNCATETABLE*полностью очищает таблицу и не допускает условного удаления. Он аналогичен оператору *DELETE*без условия *WHERE*и ограничения *LIMIT.*Удаление происходит гораздо быстрее, т. к. осуществляется не перебор записей, а полное очищение таблицы

**Обновление данных.** Обновление данных (изменение значений полей в существующих записях) обеспечивают:

* оператор UPDATE;
* оператор REPLACE.

Оператор UPDATEпозволяет обновлять отдельные поля в существующих записях. Имеет следующий синтаксис

UPDATE [IGNORE] имя таблицы

SETимя\_столбца1= выражение! [, имя\_столбца2 = выражение2 .^ ]

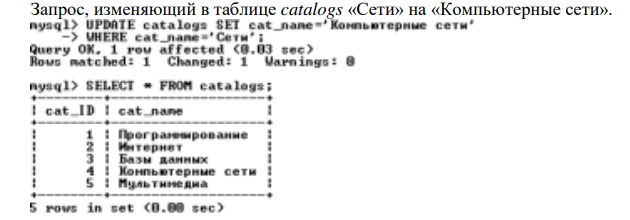
[WHEREусловие ]

[ORDERBYимяполя ]

[LIMITчисло строк] ;

После ключевого слова UPDATEуказывается таблица, которая изменяется. В предложении SETуказывается, какие столбцы обновляются и устанавливаются их новые значения. Необязательное условие WHEREпозволяет задать критерий отбора строк (обновляться будут только строки, удовлетворяющие условию).

Если указывается необязательное ключевое слово IGNORE,то команда обновления не будет прервана, даже если при обновлении возникнет ошибка дублирования ключей. Строки, породившие конфликтные ситуации, обновлены не будут.



Инструкции LIMIT и ORDER BY позволяют ограничить число изменяемых записей. При этом за один запрос можно обновить несколько столбцов таблицы. Например, необходимо в таблице books для десяти самых дешевых товарных позиций уменьшить количество книг на складе на единицу, а цену – на 5 %.

**Задание для практической работы**

При выполнении практической работы необходимо для заданной предметной области средствами MySQL:

* заполнить согласованными данными таблицы БД;
* при необходимости исправить введенную информацию;
* составить отчет по практической работе.

Операторы заполнения БД bookимеют следующий вид.

USE book;

SET CHARACTER SET cp1251;

DELETE FROM catalogs;

INSERT INTO catalogs VALUES (1,'Программирование');

INSERT INTO catalogs VALUES (2,’Интернет’);

INSERT INTO catalogs VALUES (3,'Базыданных');

INSERT INTO catalogs VALUES (4,'Сети');

INSERT INTO catalogs VALUES (5,'Мультимедиа');

DELETE FROM books;

INSERT INTO books VALUES (1,'JavaScript вкармане ','РеваО.Н.', 2008, 42.00, 10, 1); INSERT INTO books VALUES (2,'Visual FoxPro 9.0','КлепининВ.Б.', 2007, 660.00, 2, 1); INSERT INTO books VALUES (3,'C++ Каконесть','ТимофеевВ.В.',2009, 218.00, 4, 1); INSERT INTO books VALUES (4, 'СозданиеприложенийспомощьюC#', 'Фаронов В.В.2008,

1. 1, 1);

INSERT INTO books VALUES (5,'Delphi. Народныесоветы', 'ШкрыльА.А. ',2007,243.00,6,1); INSERT INTO books VALUES (6,'Delphi. Полное руководство', 'Сухарев М.',2008,500.00,6,1); INSERTINTObooksVALUES(7,'Профессиональное программирование на PHP', 'Шлосснейгл Дж.', 2006, 309.00, 5, 1);

INSERTINTObooksVALUES (8,'Совершенный код','Макконнелл С.', 2007, 771.00, 1, 1); INSERTINTObooksVALUES(9,'Практика программирования','Керниган Б.', 2004, 214.00,

12, 1);

INSERT INTO books VALUES (10,'ПринципымаршрутизациивInternet','Хелеби С.', 2001,

1. 4, 2);

INSERT INTO books VALUES (11,'ПоисквInternet','Гусев В.С. ',2004,107.00,2,2);

INSERT INTO books VALUES (12,'Web-конструирование','Дуванов А.А.', 2003, 177.00, 6, 2); INSERT INTO books VALUES (13,'СамоучительИнтернет','КонстантиновЮ.П.', 2009,

1. 4, 2);

INSERT INTO books VALUES (14,'Популярныеинтернет-браузеры','МарининС.А.', 2007,

1. 6, 2);

INSERT INTO books VALUES (15,'ОбщениевИнтернете', 'ЭкслерА.', 2006, 85.00, 5, 2); INSERT INTO books VALUES (16,'Базыданных', 'МалыхинаМ.П.', 2006, 326.00, 2, 3);

INSERT INTO books VALUES (17,'Базыданных. Разработка приложений','Рудикова Л.В.', 2006, 189.00, 6, 3);

INSERTINTObooksVALUES(18,'Раскрытие тайн SQL','Оппель Э.', 2007, 200.00, 3, 3); INSERTINTObooksVALUES(19,'Практикум по Access', 'Золотова С.И.', 2007, 87.00, 6, 3); INSERTINTObooksVALUES (20,'Компьютерные сети','Танненбаум Э.', 2007, 630.00, 6, 4); INSERTINTObooksVALUES(21,'Сети. Поиск неисправностей','Бигелоу С.', 2005, 434.00,

4, 4);

INSERTINTObooksVALUES(22,'Безопасностьсетей','БреггР.', 2006, 462.00, 5, 4);

INSERTINTObooksVALUES(23,'Анализ и диагностика компьютерных сетей', 'Хогдал Дж.', 2001, 344.00, 3, 4);

INSERTINTObooksVALUES(24,'Локальные вычислительные сети', 'Епанешников А.' , 2005, 82.00, 8, 4);

INSERT INTO books VALUES (25,'Цифроваяфотография','НадеждинН.', 2004, 149.00,

20,5);

INSERTINTObooksVALUES(26,'Музыкальный компьютер для гитариста', 'Петелин Р.Ю.', 2004, 217.00, 15, 5);

INSERT INTO books VALUES (27,'ВидеонаПК','ФедороваА.',2003,231.00,Ю,5);

INSERT INTO books VALUES (28,’МультипликациявоFlash','КиркпатрикГ.', 2006, 211.00,

20, 5);

INSERT INTO books VALUES (29,'ЗаписьCD иDVD',Tультяев А.К.', 2003, 167.00, 12, 5); INSERT INTO books VALUES (30, 'Записьиобработказвуканакомпьютере', 'ЛояничА.А. 2008, 51.00, 8, 5);

DELETE FROM users;

INSERT INTO users VALUES (1,'Александр','Валерьевич','Иванов','58-98-78',

'Ivanov®email.ru', 'active');

INSERT INTO users VALUES (2,'Сергей','Иванович','Лосев','90-57-77', 'losev®email.ru', 'passive');

INSERT INTO users VALUES(3,'Игорь', 'Николаевич', 'Симонов','95-66-61',

'simonov®email.ru', 'active');

INSERT INTO users VALUES (4,'Максим','Петрович','Кузнецов',NULL, 'kuznetsov®email.ru', 'active');

INSERT INTO users VALUES (5,'Анатолий','Юрьевич','Петров', NULL, NULL, 'lock');

INSERT INTO users VALUES(6,'Александр','Александрович','Корнеев','89-78-36',

'korneev®email.ru', 'gold');

DELETE FROM orders;

INSERT INTO orders VALUES (I,3,8,'2009-0I-04 10:39:38',!);

INSERT INTO orders VALUES (2,6,I0,'2009-02-I0 09:40:29',2);

INSERT INTO orders VALUES (3,1,20,'2009-02-18 13:41:05',4);

INSERT INTO orders VALUES (4,4,20,'2009-03-10 18:20:00',1);

INSERT INTO orders VALUES (5,3,20,'2009-03-1719:15:36',1);

**Практическая работа 15. Создание и модификация таблиц БД. Выборка данных из БД. Модификация содержимого БД**

**Цель:**Изучить основные операторы языка программирования SQL.

**Задание:**

1. Создать базу данных с помощью языка SQL.
2. Описать ход работы.
3. Ответить на контрольные вопросы.
4. Оформить отчет

**Теоретические сведения:**

Язык SQL предназначен для организации доступа к базам данных. При этом предполагается, что доступ к БД может быть осуществлен в двух режимах: в интерактивном режиме и в режиме выполнения прикладных программ (приложений).

SQL, хотя и является языком программирования, в силу своей специфической направленности не обладает многими возможно­стями универсальных языков программирования. В нем отсутству­ют традиционные операторы, организующие циклы, позволяющие объявить и использовать внутренние переменные, организовать анализ некоторых условий и возможность изменения хода про­граммы в зависимости от выполненного условия. В общем случае SQL можно назвать подъязыком, который служит исключительно для управления базами данных. Для создания приложений, насто­ящих программ необходимо использовать другие, базовые языки программирования, в которые операторы языка SQL будут встра­иваться.

Основные достоинства языка SQL заключаются в следующем:

* стандартность – как уже было сказано, использование языка SQL в программах стандартизировано международными организациями;
* независимость от конкретных СУБД – все распространенные СУБД используют SQL, т.к. реляционную базу данных можно перенести с одной СУБД на другую с минимальными доработками;
* возможность переноса с одной вычислительной системы на другую – СУБД может быть ориентирована на различные вычислительные системы, однако приложения, созданные с помощью SQL, допускают использование как для локальных БД, так и для крупных многопользовательских систем;
* реляционная основа языка – SQL является языком реляционных БД, поэтому он стал популярным тогда, когда получила широкое распространение реляционная модель представления данных. Табличная структура реляционной БД хорошо понятна, а потому язык SQL прост для изучения;
* возможность создания интерактивных запросов – SQL обеспечивает пользователям немедленный доступ к данным, при этом в интерактивном режиме можно получить результат запроса за очень короткое время без написания сложной программы;
* возможность программного доступа к БД – язык SQL легко использовать в приложениях, которым необходимо обращаться к базам данных. Одни и те же операторы SQL употребляются как для интерактивного, так и программного доступа, поэтому части программ, содержащие обращение к БД, можно вначале проверить в интерактивном режиме, а затем встраивать в программу;
* обеспечение различного представления данных – с помощью SQL можно представить такую структуру данных, что тот или иной пользователь будет видеть различные их представления. Кроме того, данные из разных частей БД могут быть скомбинированы и представлены в виде одной простой таблицы, а значит, представления пригодны для усиления защиты БД и ее настройки под конкретные требования отдельных пользователей;
* возможность динамического изменения и расширения структуры БД – язык SQL позволяет манипулировать структурой БД, тем самым обеспечивая гибкость с точки зрения приспособленности БД к изменяющимся требованиям предметной области;
* поддержка архитектуры клиент-сервер – SQL – одно из лучших средств для реализации приложений на платформе клиент-сервер. SQL служит связующим звеном между взаимодействующей с пользователем клиентской системой и серверной системой, управляющей БД, позволяя каждой из них сосредоточиться на выполнении своих функций.

Структурированный язык запросов SQL является обычным языком программирования, состоящим из операторов и правил грамматики. Запрос к таблице базы данных на языке SQL представляет собой инструкцию SELECT, которую можно описать следующим образом.

SELECT [ALL|] (список полей таблицы или запроса)

FROM (список таблиц или запросов, на основе которых формируется запрос)

[WHERE (условия отбора данных)]

[GROUPBY (список полей, выводимых в результат выполнения запроса)]

[HAVING (условия для группировки данных в запросе)]

[ORDERBY (список полей, по которым упорядочивается вывод данных в запросе)]

В рассмотренной структуре инструкции SELECTALL — клю­чевое слово, которое означает, что в результирующий набор за­писей включаются все записи таблицы или запроса, которые удов­летворяют условиям запроса. Ключевые слова могут отсутствовать в запросе.

В зависимости от характера выполняемых действий операторы SQL можно разделить на следующие группы:

• операторы определения данных;

• операторы манипулирования данными;

• операторы (язык) запросов;

• операторы управления действиями (транзакциями);

• операторы администрирования данными;

• операторы управления (управления курсором).

**Операторы определения данных**

|  |  |
| --- | --- |
| Оператор | Действие |
| CREATE TABLE | Создает новую таблицу БД |
| DROP TABLE | Удаляет таблицу из БД |
| ALTER TABLE | Изменяет структуру существующей таблицы или ограничения целостности, задаваемые для данной таблицы |
| CREATE VIEW | Создает виртуальную таблицу, соответствующую некоторому SQL-запросу |
| ALTER VIEW | Изменяет ранее созданное представление |
| DROP VIEW | Удаляет ранее созданное представление |
| CREATE INDEX | Создает индекс для некоторой таблицы для обеспечения быстрого доступа по атрибутам, входящим в индекс |
| DROP INDEX | Удаляет ранее созданный индекс |

**Операторы манипулирования данными**

|  |  |
| --- | --- |
| Оператор | Действие |
| DELETE | Удаляет одну или несколько строк, соответству­ющих условиям фильтрации, из базовой таблицы. Применение оператора согласуется с принципами поддержки целостности, поэтому этот оператор не всегда может быть выполнен корректно, даже если синтаксически он записан правильно |
| INSERT | Вставляет одну строку в базовую таблицу. Допусти­мы модификации оператора, при которых сразу несколько строк могут быть перенесены из одной таблицы или запроса в базовую таблицу |
| UPDATE | Обновляет значения одного или нескольких столб­цов в одной или нескольких строках, соответству­ющих условиям фильтрации |

**Оператор запросов**

|  |  |
| --- | --- |
| Оператор | Действие |
| SELECT | Оператор, заменяющий все операторы реляцион­ной алгебры и позволяющий сформировать резуль­тирующее отношение, соответствующее запросу |

**Операторы управления действиями (транзакциями)**

|  |  |
| --- | --- |
| Оператор | Действие |
| COMMIT | Завершает комплексную, взаимосвязанную обра­ботку информации, объединенную в транзакцию |
| ROLLBACK | Отменяет изменения, проведенные в ходе выполнения транзакции |
| SAVEPOINT | Сохраняет промежуточное состояние БД, помечает его для того, чтобы можно было в дальнейшем к нему вернуться |

**Операторы администрирования данными**

|  |  |
| --- | --- |
| Оператор | Действие |
| ALTER DATABASE | Изменяет набор основных объектов в базе данных, ограничений, касающихся всей базы данных |
| ALTER DBAREA | Изменяет ранее созданную область хранения |
| ALTER PASSWORD | Изменяет пароль для всей базы данных |
| CREATE DATABASE | Создает новую базу данных |
| CREATE DBAREA | Создает новую область хранения базы данных |
| DROP DATABASE | Удаляет базу данных |
| DROP DBAREA | Удаляет область хранения базы данных |
| GRANT | Предоставляет права доступа к базе данных или отдельным ее элементам |
| REVOKE | Лишает права доступа к базе данных или отдельным ее элементам |

**Операторы управления курсором**

|  |  |
| --- | --- |
| Оператор | Действие |
| DECLARE | Определяет курсор для запроса. Задает имя и определяет связанный с ним запрос к БД |
| OPEN | Открывает курсор. Открывает объект базы данных |
| FETH | Устанавливает курсор на определенную запись и считывает ее |
| CLOSE | Закрывает курсор. Закрывает объект базы данных |
| PREPARE | Генерирует план выполнения запроса в соответствии с инструкцией SELECT |
| EXECUTE | Выполняет сгенерированный ранее запрос |

**Типы данных языка SQL**

Данные – это совокупная информация, хранимая в базе данных в виде одного из нескольких различных типов. С помощью типов данных устанавливаются основные правила для данных, содержащихся в конкретном столбце таблицы, в том числе размер выделяемой для них памяти.

В языке SQL имеется шесть скалярных типов данных, определенных стандартом.

**Символьные данные**

Символьные данные состоят из последовательности символов, входящих в определенный создателями СУБД набор символов. Поскольку наборы символов являются специфическими для различных диалектов языка SQL, перечень символов, которые могут входить в состав значений данных символьного типа, также зависит от конкретной реализации. Для определения данных символьного типа используется следующий формат:

<символьный\_тип>::=

{ CHARACTER [ VARYING][длина] | [CHAR |

VARCHAR][длина]}

При определении столбца с символьным типом данных параметр длина применяется для указания максимального количества символов, которые могут быть помещены в данный столбец (по умолчанию принимается значение 1). Символьная строка может быть определена как имеющая фиксированную или переменную (VARYING) длину. Если строка определена с фиксированной длиной значений, то при вводе в нее меньшего количества символов значение дополняется до указанной длины пробелами, добавляемыми справа. Если строка определена с переменной длиной значений, то при вводе в нее меньшего количества символов в базе данных будут сохранены только введенные символы, что позволит достичь определенной экономии внешней памяти.

**Битовые данные**

Битовый тип данных используется для определения битовых строк, т.е. последовательности двоичных цифр (битов), каждая из которых может иметь значение либо 0, либо 1. Данные битового типа определяются при помощи следующего формата:

<битовый\_тип>::=

BIT [VARYING][длина]

**Точные числа**

Тип точных числовых данных применяется для определения чисел, которые имеют точное представление, т.е. числа состоят из цифр, необязательной десятичной точки и необязательного символа знака. Данные точного числового типа определяются точностью и длиной дробной части. Точность задает общее количество значащих десятичных цифр числа, в которое входит длина как целой части, так и дробной, но без учета самой десятичной точки. Масштаб указывает количество дробных десятичных разрядов числа.

<фиксированный\_тип>::=

{NUMERIC[точность[,масштаб]|{DECIMAL|DEC}

[точность[, масштаб]

| {INTEGER |INT}| SMALLINT}

По умолчанию длина дробной части равна нулю, а принимаемая по умолчанию точность зависит от реализации. Тип INTEGER (INT) используется для хранения больших положительных или отрицательных целых чисел. Тип SMALLINT – для хранения небольших положительных или отрицательных целых чисел; в этом случае расход внешней памяти существенно сокращается.

**Округленные числа**

Тип округленных чисел применяется для описания данных, которые нельзя точно представить в компьютере, в частности действительных чисел. Округленные числа или числа с плавающей точкой представляются в научной нотации, при которой число записывается с помощью мантиссы, умноженной на определенную степень десяти (порядок), например: 10Е3, +5.2Е6, -0.2Е-4. Для определения данных вещественного типа используется формат:

<вещественный\_тип>::=

{ FLOAT [точность]| REAL |

DOUBLEPRECISION}

Параметр точность задает количество значащих цифр мантиссы. Точность типов REAL и DOUBLEPRECISION зависит от конкретной реализации.

**Дата и время**

Тип данных "дата/время" используется для определения моментов времени с некоторой установленной точностью. Стандарт SQL поддерживает следующий формат:

<тип\_даты/времени>::=

{DATE | TIME[точность][WITHTIMEZONE]|

TIMESTAMP[точность][WITH TIME ZONE]}

Тип данных DATE используется для хранения календарных дат, включающих поля YEAR (год), MONTH (месяц) и DAY (день). Тип данных TIME – для хранения отметок времени, включающих поля HOUR (часы), MINUTE (минуты) и SECOND (секунды). Тип данных TIMESTAMP – для совместного хранения даты и времени. Параметр точность задает количество дробных десятичных знаков, определяющих точность сохранения значения в поле SECOND. Если этот параметр опускается, по умолчанию его значение для столбцов типа TIME принимается равным нулю (т.е. сохраняются целые секунды), тогда как для полей типа TIMESTAMP он принимается равным 6 (т.е. отметки времени сохраняются с точностью до миллисекунд). Наличие ключевого слова WITHTIMEZONE определяет использование полей TIMEZONEHOUR и TIMEZONEMINUTE, тем самым задаются час и минуты сдвига зонального времени по отношению к универсальному координатному времени (Гринвичскому времени).

Данные типа INTERVAL используются для представления периодов времени.

**Ход работы:**

1. Создать базу данных с помощью языка SQL.
2. Создать таблицы и индексы базы данных.
3. Заполнить таблицы данными
4. Создать запросы с помощью оператора SELECT.

**Пример:**

1. Создание базы данных my\_base

CREATEDATABASE `my\_base`

2. Создание таблицы и индекса базы данных.

CREATE TABLE

`users` (

`id` INT(11),

`name` CHAR(30) NOT NULL,

`age` SMALLINT(6) NOT NULL,

PRIMARY KEY(`id`),

INDEX (`name`(12))

)

3. Заполнениетаблицыданными

INSERT INTO `users` (`name`, `age`)

VALUES

('Катя', 12),

('Лена', 18),

('Миша', 16),

('Саша', 20)

4. Создание запроса с помощью оператора SELECT.

SELECT \* FROM `users` where age>18

**Контрольные вопросы:**

1. Какие основные достоинства имеет язык SQL?
2. На какие группы можно разделить операторы SQL?
3. Из каких основных операторов состоит каждая группа?
4. Какие типы данных существуют в языке SQL?