П.В. Сенченко

БАЗЫ ДАННЫХ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ КОНТРОЛЬНЫХ, ЛАБОРАТОРНЫХ И САМОСТОЯТЕЛЬНЫХ РАБОТ

для студентов направления подготовки бакалавров
09.03.04 (231000.62) – Программная инженерия
38.03.05 (080500.62) – Бизнес-информатика
38.03.04 (081100.62) – Государственное и муниципальное управление»

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

П.В. Сенченко

БАЗЫ ДАННЫХ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ КОНТРОЛЬНЫХ, ЛАБОРАТОРНЫХ И САМОСТОЯТЕЛЬНЫХ РАБОТ

для студентов направления подготовки бакалавров
09.03.04 (231000.62) – Программная инженерия
38.03.05 (080500.62) – Бизнес-информатика
38.03.04 (081100.62) – Государственное и муниципальное управление»

Томск 2015

П.В. Сенченко.

Базы данных: методические указания к выполнению контрольных, лабораторных и самостоятельных работ. — Томск: Томск. гос. ун-т систем управления и радиоэлектроники, 2015. — 54 с.

Цель учебно-методического пособия к курсу Базы данных – методическое сопровождение самостоятельной деятельности студентов, обучающихся по дистанционной технологии.

Учебно-методическое пособие предназначено для студентов направления подготовки бакалавров для студентов направления подготовки бакалавров 09.03.04 (231000.62) – Программная инженерия 38.03.05 (080500.62) – Бизнес-информатика 38.03.04 (081100.62) – Государственное и муниципальное управление», а также студентов родственных специальностей и направлений, сотрудников и специалистов, занимающихся разработкой баз данных.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ К ДИСЦИПЛИНЕ

Цель дисциплины – дать студентам теоретические знания и практические навыки в области проектирования, разработки и управления базами данных (БД), их использование при разработке автоматизированных информационных систем.

Основные задачи – изучение теоретических основ, определяющих организацию баз данных; приобретение практических навыков проектирования баз данных.

В ходе изучения дисциплины рассматриваются проблемы организации баз данных, теория структуризации данных, принципы построения баз данных и методы доступа к ним, современные системы управления базами, современные методики проектирования баз данных.

Данная дисциплина изучается студентами направления подготовки 231000.62 – Программная инженерия, 080500.62 – Бизнес-информатика, 081100.62 – Государственное и муниципальное управление, а также студентов родственных специальностей и направлений.

Изучение данной дисциплины в зависимости от направления подготовки базируется на дисциплинах «Информационные технологии обработки данных», «Исследование систем управления», «Дискретная математика».

Для эффективного освоения дисциплины студент должен знать: основы проектирования алгоритмов и структур данных, основные приемы проектирования человеко-машинного интерфейса, общие представления о разработке информационных технологий, методики анализа данных, основы теории множеств и применения теорикомножественных операций.

Для организации работы студентов требуется следующее программно-операционное обеспечение: ОС Windows, MS Office, СУБД MS Access.

Процесс изучения дисциплины по направлению Бизнес-информатика направлен на формирование следующих компетенций:

- по направлению «Бизнес-информатика»:
  - управлять контентом предприятия и Интернет-ресурсов, управлять процессами создания и использо-

вания информационных сервисов (контент-сервисов) (ПК-7);
  - разрабатывать контент и ИТ-сервисы предприятия и Интернет-ресурсов (ПК-18).
- по направлению «Государственное и муниципальное управление»:
  - владение основными способами и средствами информационного взаимодействия, получения, хранения, переработки, интерпретации информации, наличием навыков работы с информационно-коммуникационными технологиями; способностью к восприятию и методическому обобщению информации, постановке цели и выбору путей ее достижения (ОК-8);
  - умение обобщать и систематизировать информацию для создания баз данных, владением средствами программного обеспечения анализа и моделирования систем управления (ПК-17);
  - владение технологиями защиты информации (ПК-27);
  - способность осуществлять технологическое обеспечение служебной деятельности специалистов (по категориям и группам должностей государственной гражданской службы и муниципальной службы) (ПК-46).
- по направлению «Программная инженерия»:
  - способность к формализации в своей предметной области с учетом ограничений используемых методов исследования (ПК-2);
  - навыки использования операционных систем, сетевых технологий, средств разработки программного интерфейса, применения языков и методов формальных спецификаций, систем управления базами данных (ПК-15).
  - умение применять основные методы и инструменты разработки программного обеспечения (ПК-17).

Проверка формирования заявленных компетенций, знаний, умений и навыков осуществляется путем выполнения контрольных и проверки преподавателем контрольных работ, защиты лабораторных работ, обоснования выбранных технических решений и способов достижения результата.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ КОНТРОЛЬНЫХ РАБОТ

Контрольная работа заключается в проверке знаний по темам «Нормализация отношений» и язык «SQL».

Варианты контрольной работы

Вариант 1.

Задание 1

Заполните значения атрибутов отношения «Банки», учитывая, что атрибут № лицензии банка уникален для каждого филиала. Выявите первичный ключ и все возможные зависимости. Нормализуйте отношение по 3-й нормальной форме (3НФ).

Банки (Код филиала банка; Наименование филиала; Адрес филиала; ФИО заведующего филиалом; Наименование головного отделения банка; ФИО управляющего головным отделением; № лицензии банка)

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Банки».

Задание 3

Реализуйте SQL-запросы на добавление нескольких записей в созданные таблицы.

Вариант 2.

Задание 1

Заполните значения атрибутов отношения «Счета клиентов», учитывая, что один клиент может иметь несколько счетов как в одном, так и в нескольких банках, при этом номера счетов в разных банках могут совпадать. Выявите первичный ключ и все возможные зависимости, нормализуйте отношение по 3НФ.

Счета клиентов (Код клиента; ФИО клиента; Код банка; Наименование банка; № счета; ФИО управляющего банком).

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Счета клиентов».

Задание 3

Реализуйте SQL-запросы на добавление записей в созданные таблицы.

Вариант 3.

Задание 1

Заполните значения атрибутов отношения «Операции», учитывая, что одному пациенту может быть сделано несколько операция (в том числе одинаковых), но не более 1-й операции в день, а номер истории болезни уникален для каждого пациента. Выявите первичный ключ и все возможные зависимости, нормализуйте отношение по 3НФ.

Операции (№ оперируемого; ФИО пациента; № истории болезни; Адрес пациента; ФИО хирурга; Дата операции; Наименование операции; Вид операции).

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Операции».

Задание 3

Реализуйте SQL-запросы на добавление записей (по одной записи) в созданные таблицы.

Вариант 4.

Задание 1

Заполните значения атрибутов отношения «Клиенты банков», учитывая, что один клиент может иметь несколько счетов как в одном, так и в нескольких банках, при этом номера счетов в разных банках могут совпадать. Выявите первичный ключ и все возможные зависимости, нормализуйте отношение по 3НФ.

Клиенты банков (Серия паспорта клиента; № паспорта клиента; ФИО клиента; Код банка; Наименование банка; Адрес банка; № счета; ФИО оператора).

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Клиенты банков».

Задание 3

Реализуйте SQL-запросы на добавление записей в созданные таблицы.

Вариант 5.

Задание 1

Заполните значения атрибутов отношения «Операции», учитывая, что одному пациенту может быть сделано несколько операция (в том числе одинаковых), но не более 1-й операции в день. Выявите первичный ключ и все возможные зависимости, нормализуйте отношение по 3НФ.

Операции (№ пациента; Фамилия пациента; Дата операции; Адрес пациента; ФИО хирурга; Наименование операции).

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Операции».

Задание 3

Реализуйте SQL-запросы на добавление записей в созданные таблицы.

Вариант 6.

Задание 1

Заполните значения атрибутов отношения «Банки», учитывая, что атрибут № лицензии банка уникален для каждого головного отделения. Выявите первичный ключ и все возможные зависимости. Нормализуйте отношение по 3НФ.

Банки (Код филиала банка; Наименование филиала; Адрес филиала; ФИО заведующего филиалом; Наименование головного отделения банка; ФИО управляющего головным отделением; № лицензии банка)

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Банки».

Задание 3

Реализуйте SQL-запросы на добавление нескольких записей в созданные таблицы.

Вариант 7.

Задание 1

Заполните значения атрибутов отношения «Отделы». Выявите первичный ключ и все возможные зависимости, учитывая возможность совмещения сотрудником нескольких должностей в разных отделах. Нормализуйте отношение по 3НФ.

Отделы (Код отдела; Название отдела; ФИО сотрудника отдела; Дата рождения сотрудника отдела; Телефон сотрудника отдела; Должность сотрудника; Дата приема на работу)

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Отделы».

Задание 3

Реализуйте SQL-запросы на добавление нескольких записей в созданные таблицы.

Вариант 8.

Задание 1

Заполните значения атрибутов отношения «Успеваемость». Выявите первичный ключ и все возможные зависимости. Нормализуйте отношение по 3НФ.

Успеваемость (Серия паспорта студента; № паспорта студента; ФИО студента; Дата рождения студента; № группы, Название предмета; Семестр; ФИО преподавателя; Оценка)

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Успеваемость».

Задание 3

Реализуйте SQL-запросы на добавление нескольких записей в созданные таблицы.

Вариант 9.

Задание 1

Заполните значения атрибутов отношения «Автосалон», учитывая, что один автомобиль может продаваться в автосалоне несколько раз. Выявите первичный ключ и все возможные зависимости. Нормализуйте отношение по 3НФ.

Автосалон (VIN автомобиля; Марка автомобиля; Модель автомобиля; Дата выпуска; Цвет; Дата продажи; Серия паспорта покупателя; № паспорта покупателя; ФИО покупателя; Дата рождения покупателя; Адрес покупателя)

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Автосалон».

Задание 3

Реализуйте SQL-запросы на добавление нескольких записей в созданные таблицы.

Вариант 10.

Задание 1

Заполните значения атрибутов отношения «Входящие документы», учитывая, что один документ может быть передан на исполнения разным сотрудникам, а также возможность совмещения сотрудником нескольких должностей. Выявите первичный ключ и все возможные зависимости. Нормализуйте отношение по 3НФ.

Входящие документы (Входящий номер; Входящая дата; Краткое содержание; Организация отправитель; Сотрудник-исполнитель; Дата передачи документа; Срок исполнения; Должность сотрудника; Ставка на занимаемой должности).

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Входящие документы».

Задание 3

Реализуйте SQL-запросы на добавление нескольких записей в созданные таблицы.

Вариант 11.

Задание 1

Заполните значения атрибутов отношения «Исходящие документы», учитывая, что один документ может быть отправлен в разные организации, а также возможность совмещения сотрудником нескольких должностей. Выявите первичный ключ и все возможные зависимости. Нормализуйте отношение по 3НФ.

Исходящие документы (Исходящий номер; Исходящая дата; Краткое содержание; Организация-получатель; ФИО руководителя организации-получателя; Сотрудник-автор документа; Дата подписания документа; Должность сотрудника; Ставка на занимаемой должности).

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Исходящие документы».

Задание 3

Реализуйте SQL-запросы на добавление нескольких записей в созданные таблицы.

Вариант 12.

Задание 1

Заполните значения атрибутов отношения «Приказы», учитывая, что один приказ может быть передан на исполнение нескольким сотрудникам, а также возможность совмещения сотрудником нескольких должностей. Выявите первичный ключ и все возможные зависимости. Нормализуйте отношение по 3НФ.

Внутренние документы (Номер приказа; Дата утверждения; Краткое содержание; Сотрудник-исполнитель; Дата передачи на исполнение; Должность сотрудника; Ставка на занимаемой должности).

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Приказы».

Задание 3

Реализуйте SQL-запросы на добавление нескольких записей в созданные таблицы.

Вариант 13.

Задание 1

Заполните значения атрибутов отношения «Внутренние документы», учитывая, что один документ может быть передан на исполнение нескольким сотрудникам, а также возможность совмещения сотрудником нескольких должностей. Также необходимо учесть, что документы разных типов (Поручения, Служебные записки, Распоряжения и т.д.) могут иметь одинаковые номера. Выявите первичный ключ и все возможные зависимости. Нормализуйте отношение по 3НФ.

Внутренние документы (Номер документа; Дата утверждения; Тип документа; Краткое содержание; Сотрудник-Исполнитель; Дата передачи на исполнение; Должность сотрудника; Ставка на занимаемой должности).

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Внутренние документы».

Задание 3

Реализуйте SQL-запросы на добавление нескольких записей в созданные таблицы.

Вариант 14.

Задание 1

Заполните значения атрибутов отношения «Организации», учитывая, что отношение не нормализовано по 1НФ (атрибут «Адрес организации» является составным). Выявите первичный ключ и все возможные зависимости, нормализуйте отношение по 3НФ.

Организации (ИНН организации; Наименование организации; ФИО директора; Адрес организации; Телефон; Отдел в организации; ФИО начальника отдела).

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Организации».

Задание 3

Реализуйте SQL-запросы на добавление нескольких записей в созданные таблицы.

Вариант 15.

Задание 1

Заполните значения атрибутов отношения «Книговыдача», учитывая, что один автор может написать несколько книг, а одна книга может быть написана несколькими авторами. Выявите первичный ключ и все возможные зависимости, нормализуйте отношение по 3НФ.

Книговыдача (Код книги; Название книги; ФИО автора; год выпуск; Номер читательского билета; ФИО читателя; Телефон читателя; Дата выдачи книги; Дата возврата).

Задание 2

Реализуйте SQL-запросы на создание таблиц, созданных в ходе нормализации отношения «Организации».

Задание 3

Реализуйте SQL-запросы на добавление нескольких записей в созданные таблицы.

Порядок выполнения контрольной работы

Выполнение задания 1.

Нормализация предложенного отношения должна быть проведена с учетом правил нормализации отношений, изложенных в разделе 4.1. учебного пособия.

В отношениях должны быть определены первичные ключи и внешние ключи (для результирующих отношений), все возможные зависимости между атрибутами отношений. ***Суррогатные первичные ключи допускается создавать в нормализованных отношениях только в том случае, если отсутствует возможность явного определения первичного ключа.***

При нормализации отношений необходимо руководствоваться следующими основными определениями.

Отношение R находится во второй нормальной форме (2НФ) тогда и только тогда, когда отношение находится в первой нормальной форме, и каждый его не ключевой атрибут полностью зависит от первичного ключа. Или, что тоже справедливо, отношение, находящееся во второй нормальной форме не содержит атрибутов, зависящих от части ключа.

Отношение R находится в третьей нормальной форме (3НФ) в том и только в том случае, если находится в 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа, т.е. среди атрибутов отношения нет атрибутов, транзитивно зависящих от ключа (среди его неключевых атрибутов нет зависящих от другого неключевого атрибута).

Нормализации подвергаются отношения, каждое из которых содержит характеристики объектов предметной области, при этом на каждом следующем шаге проектирования (нормализации) с помощью декомпозиции отношений, достигается такой набор схем отношений, что каждая следующая нормальная форма обладает лучшими свойствами, чем предыдущая.

Для выявления аномалий и коллизий необходимо заполнить значения атрибутов исходного и результирующих отношений. Обратите внимание, что каждому нормализованному отношению должно быть присвоено имя.

Выполнение задания 2 и 3.

Формирование запросов на создание таблиц.

Перед созданием SQL-запросов, рекомендуется повторить раздел 5 учебного пособия.

При создании запросов необходимо изучить синтаксис команды CREATE TABLE, с помощью которой создается новая таблица и которая используется для описания полей и индексов в таблице.

Для каждого поля необходимо определить размер и тип данных.

В следующем примере представлено создание двумя запросами новых таблиц «Студент» и «Задолженность_за_обучение» с внешним ключом «Код_студента», связанным с полем «Код_студента», в таблице «Студент».

```
CREATE TABLE Студент
(Код_студента AUTOINCREMENT PRIMARY KEY,
Номер_зачетной_книжки INTEGER , ФИО_студента TEXT(50),
Место_рождения TEXT (50));

CREATE TABLE Задолженность_за_обучение
(Код_задолженности AUTOINCREMENT PRIMARY KEY,
Код_студента INTEGER, Сумма_задолженности MONEY,
CONSTRAINT f1_i FOREIGN KEY (Код_студента)
REFERENCES Студент (Код_студента)
```

Формирование запросов на добавление данных.

Добавление записей в таблицы необходимо осуществить с помощью команды INSERT INTO. Для этого необходимо изучить синтаксис данной команды.

В следующем примере с помощью запроса добавляется новая запись в таблицу «Студент»:

```
INSERT INTO Студент (Номер_зачетной_книжки,
ФИО_студента, Место_рождения, Дата_рождения )
VALUES (201454321, 'Иванов Иван Петрович', 'г. Томск',
'12.02.1996');
```

Все запросы необходимо выполнить в какой-либо СУБД (например, в СУБД MS Access). Результат выполнения запросов следует представить в виде скриншотов, либо в виде mdb-файла (для СУБД MS Access).

3 МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Лабораторная работа № 1 «Создание базы данных в СУБД MS Access»

Тема: Организация хранения данных в СУБД MS Access. Создание таблиц, Построение схемы БД.

Раздел дисциплины: Обоснование концепции баз данных.

Цель работы: разработать структуру базы данных (БД) для выбранной предметной области, содержащую не менее пяти взаимосвязанных таблиц.

Продолжительность: 6 часов.

Организация базы данных в среде MS Access

Microsoft Access – это функционально полная реляционная СУБД. ***База данных*** в MS Access представляет собой совокупность объектов, хранящихся в одном файле с расширением mdb (рис.1).

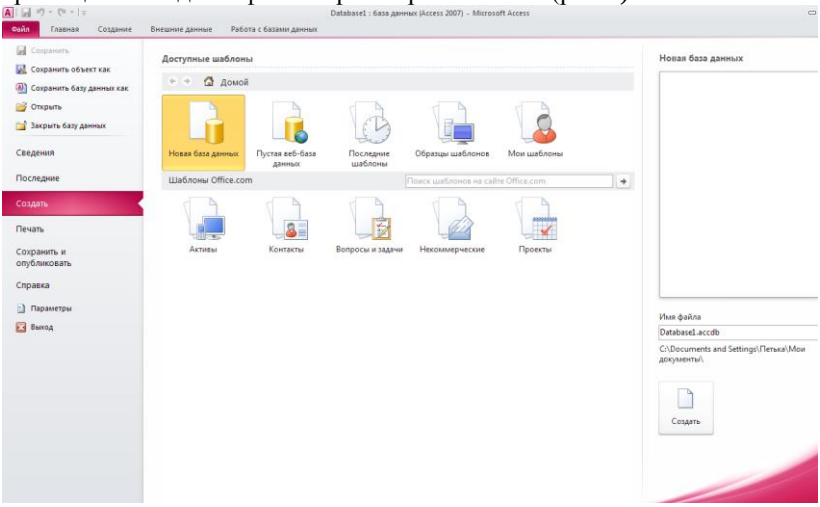


Рис. 1. Окно БД MS Access

Поддерживаются следующие типы объектов: таблицы, формы, запросы, отчеты, макросы, программные модули.

Ниже представлены характеристики БД в СУБД MS Access XP:

- размер файла базы данных Microsoft Access (.mdb) — 2 Гбайт за вычетом места, необходимого системным объектам;
- число объектов в базе данных — 768;
- модули (включая формы и отчеты, свойство Наличие модуля (HasModule) который имеет значение True) – 1 000;
- число знаков в имени объекта — 64;
- число знаков в пароле — 14;
- число знаков в имени пользователя или имени группы — 20;
- число одновременно работающих пользователей — 255;

Основным объектом в БД является ***таблица***, хранящая данные о том или ином предмете реального мира. Остальные типы объектов – это различные способы представления информации из таблиц (формы, отчеты, динамические наборы) или действия над таблицами (запросы, макросы, модули).

Запрос – это объект, позволяющий как извлекать данные из таблиц с использованием различных критериев, задаваемых пользователем, так и производить различные изменения в таблицах БД. С помощью запроса можно выбрать, изменить или сгруппировать какие-либо данные, содержащиеся в одной или нескольких таблицах. Ответ на запрос также выглядит в виде таблицы и называется ***динамическим набором записей.***

Форма – это объект, предназначенный для ввода, изменения и просмотра записей в удобном виде на экране. Форма может содержать данные из одной или нескольких взаимосвязанных таблиц, а также не связанные с таблицами данные. Для создания и изменения формы используется методика визуального программирования.

Отчет – это объект, предназначенный для печати данных в определенном пользователем виде. Отчет позволяет сгруппировать записи, производить расчеты и выводить как промежуточные, так и полные итоговые значения.

Макрос – это набор из одной или нескольких макрокоманд, позволяющих производить различные операции с объектами БД. Например, с помощью макроса при загрузке БД можно автоматически открыть нужные формы или при нажатии кнопки в форме выполнить различные действия (печать формы, открытие другой формы и т.п.) Макрокоманды выбираются из списка стандартных макрокоманд, например.

Модуль - это набор процедур и функций на языке Visual Basic. Модули обычно используют для создания достаточно сложных информационных систем. Каждый модуль может быть привязан к объектам форм и отчетам.

Каждый объект имеет структуру, характерную для его типа. Например, таблицы состоят из полей и записей. Формы и отчеты состоят из элементов управления, заголовка и др. Модули состоят из процедур и функций; макросы из макрокоманд. Многие из структурных элементов объектов также считаются объектами.

Все объекты имеют уникальные имена. Имя объекта может состоять из 64 символов, включая пробелы и другие знаки, кроме символов точка (.), восклицательный знак (!), апостроф (‘), квадратные скобки [ ]. Рекомендуется не включать в имена объектов пробелы и избегать слишком длинных имен, что затрудняет программирование приложений.

Свойство представляет собой характеристику объекта, например, имя, размер, цвет, тип данных поля и т.п. Свойства текущего объекта сведены в таблицу и доступны для изменения в окне свойств, которое открывается при нажатии кнопки  на панели инструментов. Набор свойств различен для каждого типа объектов.

Над любым объектом можно выполнить три стандартных действия (им соответствуют три кнопки в окне БД): открыть текущий объект для работы; создать новый объект текущего типа; изменить текущий объект (конструктор).

Порядок выполнения лабораторной работы

Выберите один из предложенных ниже вариантов предметных областей:

1. Библиотека;
2. Магазин продовольственных товаров;
3. ВУЗ;
4. Супермаркет;
5. Документооборот предприятия;
6. Агентство недвижимости;
7. Компьютерная фирма;
8. Поликлиника;
9. Турфирма;
10. Гостиница;
11. Автосалон;
12. Банк;
13. Деканат;
14. Отдел кадров;
15. Аэропорт.

Проанализируйте объекты выбранной предметной области и создайте не менее 8-ми взаимосвязанных таблиц в соответствии с поряд-

ком выполнения работы, учитывая, что все таблицы должны быть нормализованы по 3НФ.

После создания схемы базы данных заполните созданные таблицы.

Для запуска MS Access выберите иконку  в меню программ MS Windows. Чтобы начать разработку новой базы данных, следует в меню *Файл* выбрать команду *Создать* после чего выбрать пункт *Новая база данных* и присвоить имя новой БД. Затем возможно создание объектов БД “вручную” либо с помощью Мастера, который автоматически генерирует объект в диалоге с пользователем. Независимо от способа создания объекта режим конструктора позволяет в любой момент изменить его структуру и свойства.

Создание структуры таблиц

В СУБД MS Access отношение БД называют таблицей, кортежи отношения – записями, атрибуты – полями.

Для создания структуры таблицы в окне База данных необходимо выбрать пункт Таблица и нажать кнопку Создать. В результате откроется диалоговое окно Создание таблицы, в котором следует выбрать режим Новая таблица. Создание структуры таблицы необходимо производить в режиме конструктора таблиц.

В результате выполнения указанных действий Access выводит на экран окно пустой таблицы в режиме конструктора (рис. 2).

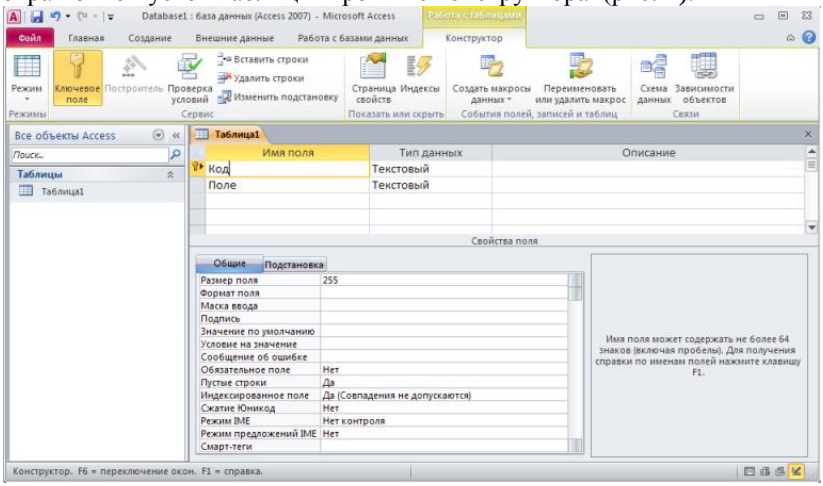


Рис. 2. Новая таблица в режиме конструктора

После того как окно таблицы откроется, активизируется панель инструментов Конструктор таблицы. При определении полей таблицы

для каждого поля необходимо ввести имя, тип данных и краткое описание.

Обязательными свойствами каждого поля являются имя, тип и размер. Имя поля задается в столбце *Поле* по тем же правилам, что имена других объектов. Во втором в столбце *Тип данных* открывается список возможных типов данных. Требуемое значение типа данных можно либо выбрать из списка, либо ввести непосредственно с клавиатуры, не прибегая к помощи списка. *Тип данных* определяет, какого вида данные будут храниться в поле – текст, числа, даты и т.д. Важно правильно определить тип поля до того, как начнется ввод данных, в противном случае при изменении типа данные могут быть искажены или утеряны. Ниже приведены типы данных, используемых в СУБД MS Access:

- текстовый (до 255 символов);
- числовой с разной степенью точности;
- дата / время;
- примечания (МЕМО) - до 64000 символов;
- счетчик (для служебных полей, типа КодТовара и т.п.);
- денежный;
- логический (да / нет);
- гиперссылка
- OLE (для хранения данных, сформированных другими прикладными программами - рисунков, схем, звукозаписей, форматированных текстов и т.п.).

Для текстовых и числовых полей пользователь может задать необходимый *размер* поля, при этом следует учитывать специфику хранимых в конкретном поле данных.

Для каждого поля можно задать *дополнительные свойства* – способ отображения (формат), подпись, используемая в запросах, формах и отчетах, значение поля по умолчанию, правила контроля для ввода данных. Определение этих свойств в ряде случаев позволяет ускорить разработку прикладной программы, описать часть ограничений целостности БД, которые будут проверяться автоматически. Для типов данных *текстовый* и *мемо* может быть задан пользовательский формат ввода значений данных, описание которого приведено в разделе справочной системы Access Форматирование текста.

Описание форматов для различных типов данных представлено в таблице 1.

Описание форматов данных Таблица 1

Наименование формата	**Описание**
Для типов данных: **Числовой, Денежный**	
стандартный формат	устанавливается по умолчанию (разделители и знаки валют отсутствуют)
Денежный	символы валют и два знака после запятой
Евро	Используется денежный формат с символом евро (€) вне зависимости от символа денежной единицы
Фиксированный	выводится, по крайней мере, один разряд
с разделителями разрядов	два знака после запятой и разделители тысяч
Процентный	процент
Экспоненциальный	экспоненциальный формат (например 3.46 * 10^3)
Для типа данных **Дата/Время** существует следующий набор форматов поля:	
длинный формат	Среда, 29 января 2003 г.
средний формат	29 – янв – 03
Наименование формата	**Описание**
краткий формат	29.01.03
длинный формат времени	10:30:10 PM
средний формат времени	10:30 PM
краткий формат времени	15:30

Для *логического* типа данных используется следующий набор форматов: Да/Нет, Истина/Ложь, Вкл/Выкл.

Число десятичных знаков – для числового и денежного типов данных задает число знаков, выводимых после запятой. По умолчании устанавливается значение Авто, при котором для форматов *денежный*, *фиксированный*, *с разделителем разрядов* и *процентный* выводятся два десятичных знака после запятой. Для формата *стандартный*, число выводимых знаков определяется текущей точностью числовых значений. Можно задать фиксированное число десятичных знаков от 0 до 15.

Маска ввода – для *текстового, числового, денежного* типов данных, а так же для типов Дата/Время задается маска ввода, которую пользователь увидит при вводе данных в это поле (например, разделители (_._.__) для поля типа Дата).

Для обеспечения уникальности записей в каждой таблице необходимо наличие первичного ключа – **ключевого поля** таблицы. Общепринятые правила при определении первичного ключа:

- в качестве ключа чаще всего выбирают числовой или символьный код, который используется только для внутренних целей БД и не доступен для изменения пользователем;
- тип ключевого поля – «счетчик» или «числовой».

При необходимости первичный ключ в таблице может состоять из нескольких полей – составной первичный ключ.

Для определения первичного ключа необходимо убедиться, что курсор установлен в поле, которое будет определено как ключ (или выделить несколько полей для составного ключа), затем в меню Правка выбрать команду Ключевое поле либо нажать кнопку  на панели инструментов, в результате должен появиться значок ключа слева от имени поля.

Создание связи между таблицами

Связи между таблицами являются необходимым элементом структуры БД. После определения нескольких таблиц необходимо определить, как данные таблицы связаны между собой, для обеспечения полноценной работы с БД – эти связи Access будет использовать в запросах, формах и отчетах.

Для определения связей в БД необходимо в меню *Сервис* выбрать пункт *Схема данных*, либо нажать кнопку  на панели управления В результате откроется диалоговое окно Схема данных, а затем – диалоговое окно Добавление таблицы, в котором следует выбрать соединяемые таблицы.

При установлении связи необходимо помнить, что для второй (подчиненной) таблицы должен быть определен внешний ключ – поле, предназначенное для связи с главной таблицей тип данных и размер которого совпадают с полем первичного ключа главной таблицы. Например, для сопоставления сведений о товарах и оплате за проданный товар следует определить связь по полю «Код_товара» в двух таблицах: «Список товаров» (Код_товара, Наименование, Единица измерения) и «Оплата» (Код_товара, Дата_продажи, Сумма). В первой таблице общее поле является первичным ключом, а во второй – внешним ключом.

Для установления непосредственной связи между двумя выбранными таблицами следует перенести с помощью мыши ключевое поле одной таблицы в другую. В результате откроется диалоговое окно Связи (рис. 3)

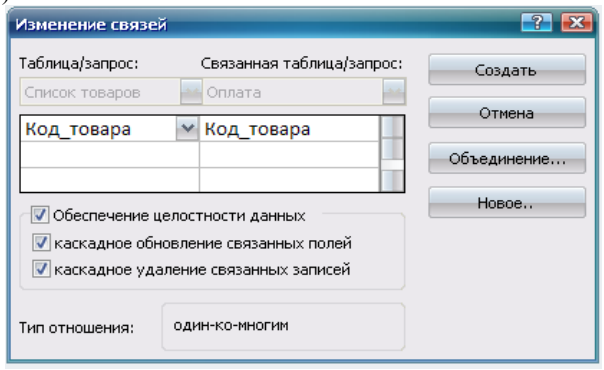


Рис. 3. Окно изменения связей между таблицами

В этом окне следует выбрать опцию Обеспечение целостности данных, после чего выбрать пункты Каскадное обновление связанных полей и Каскадное удаление связанных записей, тем самым, обеспечив требование целостности по ссылкам. Для сохранения связи нажмите кнопку Создать, в результате чего в окне Схема данных будет отображена связь между выбранными таблицами (рис. 4.).

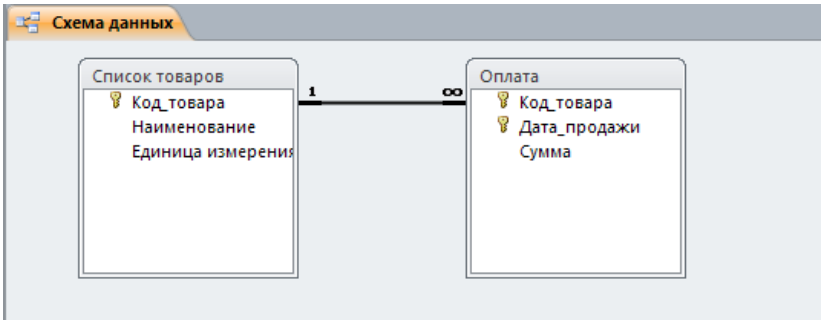


Рис. 4. Определение связей между таблицами

В СУБД MS Access допускаются следующие типы связей:

«один-ко-многим» является наиболее часто используемым типом связи между таблицами. Например, между таблицами «Список Товаров» и «Оплата» существует отношение «один-ко-многим»: товар одного наименования может продаваться различным покупателям, но каждая оплата была произведена за определенный товар.

«многие-ко-многим» реализуется только с помощью третьей таблицы, первичный ключ которой состоит из ключевых полей тех таблиц, которые необходимо связать. Например, между таблицами «Сотрудники» и «Должности» имеется отношение «многие-ко-многим», поскольку один сотрудник может занимать несколько должностей и на одну должность может занимать несколько сотрудников, такая связь может быть реализована с помощью дополнительной таблицы «Занимаемые должности».

«один-к-одному». В этом случае каждая запись в одной таблице может быть связана только с одной записью в другой таблице и наоборот. Этот тип связи используют редко, поскольку такие данные могут быть помещены в одну таблицу. Например, такую связь используют для разделения очень больших по структуре таблиц, для отделения части таблицы по соображениям защиты и т.п.

Следует помнить, что нельзя изменить тип данных для поля, которое связывает таблицу с другой таблицей. Предварительно нужно удалить установленную связь.

Лабораторная работа № 2 «Создание SQL и QBE запросов в СУБД MS Access»

Тема: Создание запросов в СУБД MS Access. Создание запросов с помощью визуального средства построителя запросов и с помощью языка SQL.

Раздел дисциплины: Языки управления и манипулирования данными

Цель работы: создать запросы в среде MS Access.
Продолжительность: 8 часов.

Типы запросов, создаваемых в Microsoft Access

В среде MS ACCESS можно создавать следующие типы запросов:

- запросы на выборку;
- запросы с параметрами;
- перекрестные запросы;
- запросы на изменение (запросы на создание таблицы, удаление, обновление, добавление записей);
- запросы SQL (запросы на объединение, запросы к серверу, управляющие запросы, подчиненные запросы).

Наиболее часто используемым запросом является запрос на выборку, возвращающий данные из одной или нескольких таблиц, а также результаты, которые при желании пользователь может изменить. Запрос на выборку можно использовать для группировки записей, для вычисления сумм, средних значений, пересчета и других действий.

Запрос с параметрами – это запрос, содержащий в себе условие для возвращения записей или значение, которое должно содержаться в поле таблицы. Например, можно создать запрос, в результате которого выводится приглашение на ввод временного интервала. В результате будут возвращены все записи, находящиеся между двумя указанными датами.

Запросы на изменение – это запросы, при запуске которых за одну операцию вносятся изменения в несколько записей. Существует четыре типа запросов на изменение: на удаление, на обновление и добавление записей, а также на создание таблицы.

Запрос на удаление удаляет группу записей из одной или нескольких таблиц. С помощью запроса на удаление можно удалить только всю запись, а не содержимое отдельных полей внутри нее.

Запрос на удаление позволяет удалить записи как из одной таблицы, так и из нескольких таблиц со связями «один-к-одному» или с «один-ко-многим», если при определении связей было установлено каскадное удаление.

Запрос на обновление записей вносит общие изменения в группу записей одной или нескольких таблиц. Например, можно с помощью одного запроса увеличить на 30 процентов стипендию студентов 3-го курса. Запрос на обновление записей позволяет изменять данные только в существующих таблицах.

Создание запросов с помощью построителя запросов

В СУБД MS Access существует специальное средство построения запросов, которое позволяет создавать простые запросы. Первые три запроса, создаваемые в рамках данной лабораторной работы, должны быть реализованы с помощью построителя запросов в режиме конструктора (рис. 5).

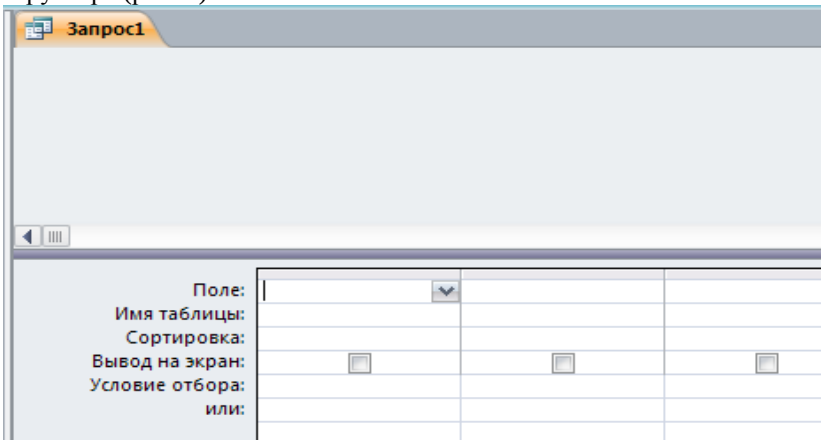


Рис. 5. Бланк построителя запросов

Для создания нового запроса в окне базы данных (рис. 1) перейдите на вкладку Запросы и нажмите кнопку Создание запроса в режиме конструктора. В появившемся окне (рис. 6) выберите таблицу (таблицы) – источник запроса. Если запрос уже открыт, то для перехода в режим конструктора следует нажать кнопку Вид  на панели инструментов.

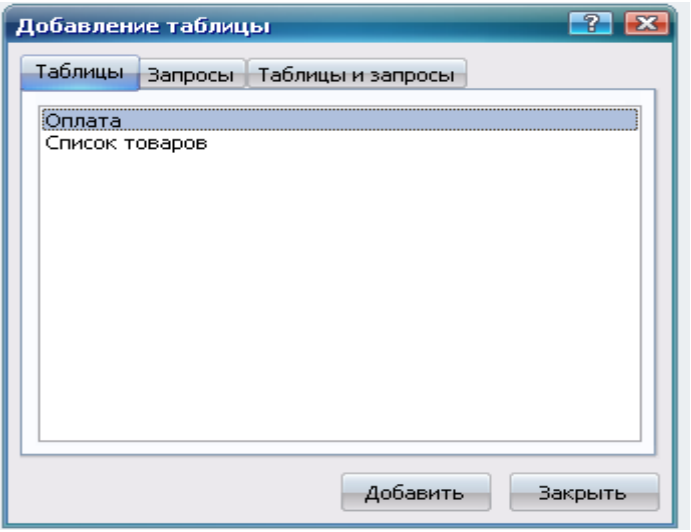


Рис. 6. Добавление таблицы

По умолчанию, любой запрос, создаваемый с помощью конструктора, является запросом на выборку. Для изменения типа запроса в меню Запрос следует выбрать тип создаваемого запроса либо на панели инструментов нажать кнопку Тип запроса .

Для сохранения запроса необходимо нажать кнопку на панели инструментов и ввести имя запроса, под которым он будет сохранен.

Создание запроса на выборку

При создании запроса на выборку, необходимо определить поля, которые будут содержаться в результирующем наборе данных, для этого в строке Имя таблицы (рис.7) выбрать название таблицы, а в строке Поле выбрать названия полей. Для сортировки записей в строке Сортировка можно указать тип сортировки: по возрастанию или по убыванию значений.

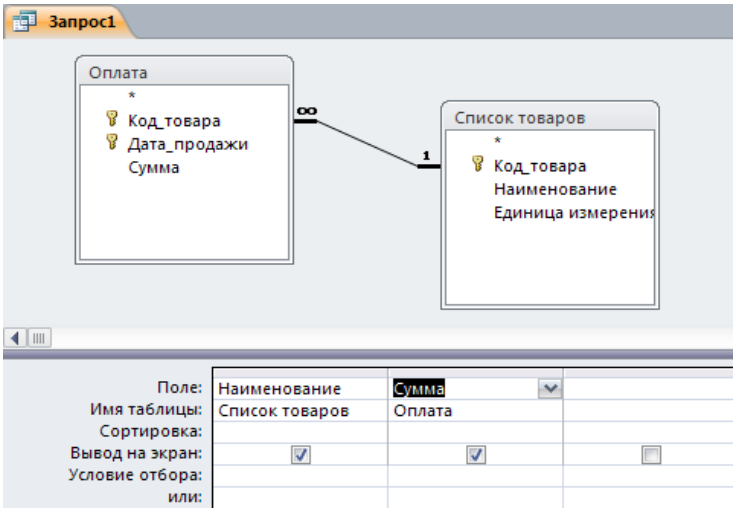


Рис. 7. Запрос на выборку

Связи между таблицами в окне построителя запросов определяются по тому же принципу, что и при определении связей в схеме данных. Для запуска запроса нажмите кнопку Запуск на панели инструментов.

Создание запроса на выборку с параметрами

Различают два типа запросов с параметрами: с приглашением на ввод условий отбора и с явным указанием условия отбора

Запрос с параметрами отображает одно или несколько определенных диалоговых окон, выводящих приглашение пользователю ввести условия отбора.

Для создания запроса с параметрами создайте новый запрос на выборку, после чего, в режиме конструктора для каждого поля, для которого предполагается использовать параметр, введите в ячейку строки Условие отбора текст приглашения, заключенный в квадратные скобки. Это приглашение будет выводиться при запуске запроса. Текст подсказки должен отличаться от имени поля, но может включать его (рис. 8), введенное в окне приглашения значение будет являться значением параметра.

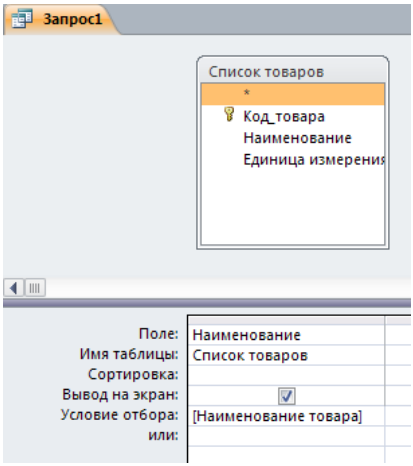


Рис. 8. Запрос на выборку с параметрами

При запуске запроса (рис.8) будет выведена подсказка Наименование товара. Для явного указания условия отбора, в построителе запроса, текстовый параметр необходимо заключить в кавычки: "Сахар", значение числового параметра указывается без дополнительных символов.

Создание запроса на обновление данных

Для создания запроса на обновление создайте запрос, выбрав таблицу или таблицы, включающие записи, которые необходимо обновить, и поля, которые должны быть использованы в условиях отбора. В режиме конструктора запроса нажмите стрелку рядом с кнопкой Тип запроса на панели инструментов и выберите команду Обновление.

Выберите поля, значения которых необходимо обновить, после чего в строку Обновление введите выражение или значение, которое должно быть использовано для изменения полей (рис. 9)

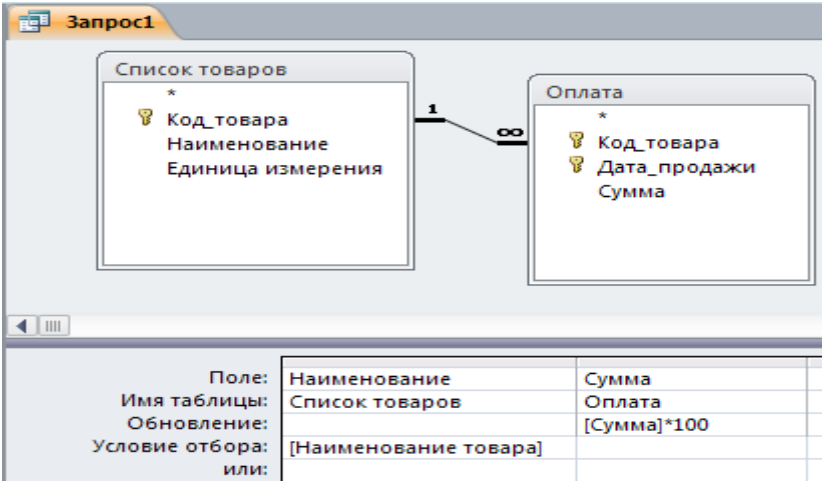


Рис. 9. Запрос на обновление

В строке Условие отбора укажите условие отбора – значения каких записей следует изменить, в противном случае значения выбранных полей будут изменены во всех записях таблицы.

Для просмотра обновляемых записей нажмите кнопку Вид на панели инструментов. Выводимый список будет содержать значения, которые будут изменены. Для возврата в режим конструктора запроса снова нажмите кнопку Вид на панели инструментов. Для запуска запроса нажмите кнопку Запуск на панели инструментов. Чтобы остановить выполнение запроса, нажмите клавиши Ctrl+Break.

Создание запроса на удаление записей

Для создания запроса на удаление создайте запрос, выбрав таблицу, содержащую записи, которые необходимо удалить.

В режиме конструктора запроса нажмите стрелку рядом с кнопкой Тип запроса на панели инструментов и выберите команду Удаление либо выберите тип запроса в меню Запрос.

Следует помнить, что при удалении записей с помощью запроса на удаление отменить данную операцию невозможно. Следовательно, прежде чем выполнить данный запрос, необходимо просмотреть выбранные для удаления данные. Для этого на панели инструментов нажмите кнопку Вид и просмотрите в режиме таблицы удаляемые записи.

Для избежания удаления всех записей из таблицы в условии отбора следует указать значение условия для выборки удаляемых записей.

В результате выполнения запроса на удаление будут удалены записи из подчиненных таблиц, для которых установлено «Каскадное удаление связанных записей».

Создание SQL-запросов

Для описания команд, используемых в SQL-запросах будем использовать следующий синтаксис написания SQL-предложений:

– в описании команд слова, написанные прописными латинскими буквами, являются зарезервированными словами SQL;

– фрагменты SQL-предложений, заключенные в фигурные скобки и разделенные символом «|», являются альтернативными. При формировании соответствующей команды для конкретного случая необходимо выбрать одну из них;

– фрагмент описываемого SQL-предложения, заключенный в квадратные скобки **[ ],** имеет необязательный характер и может не использоваться;

– многоточие перед закрывающейся скобкой, говорит о том, что фрагмент, указанный в этих скобках, может быть повторен.

На рисунке 10 схематично представлен SQL-запрос на выборку данных из таблицы «Студент»

Рис. 10. Схематичное представление SQL-запроса

Для создания нового SQL-запроса в окне базы данных нажмите кнопку Создание запроса в режиме конструктора и, не выбирая таблицы, для запроса нажмите кнопку Вид  на панели инструментов. Для запуска запроса нажмите кнопку Запуск  на панели инструментов.

Создание новой таблицы

Инструкция **CREATE TABLE** создает новую таблицу и используется для описания ее полей и индексов. Если для поля добавлено ограничение NOT NULL, то при добавлении новых записей это поле не должно содержать не существующих (NULL) данных. Синтаксис:

CREATE TABLE таблица (поле_1 тип [(размер)]

[NOT NULL] [индекс_1] [, поле_2 тип [(размер)]

[NOT NULL] [индекс_2] [, ...]] [, CONSTRAINT составной Индекс [, ...]]),

где таблица – имя создаваемой таблицы;

поле_1, поле_2 – имена одного или нескольких полей, создаваемых в новой таблице. Таблица должна содержать хотя бы одно поле;

тип – тип данных поля в новой таблице;

размер – размер поля в символах (только для текстовых и двоичных полей);

индекс_1, индекс_2 – предложение CONSTRAINT, предназначенное для создания простого индекса;

составной Индекс – предложение CONSTRAINT, предназначенное для создания составного индекса.

В следующем примере представлено создание новой таблицы «Студент»:

CREATE TABLE Студент (Код_студента AUTOINCREMENT PRIMARY KEY, Номер_зачетной_книжки INTEGER , ФИО_студента TEXT(50), Место_рождения TEXT (50));

В результате выполнения этого запроса в БД СУБД MS Access будет создана таблица «Студент» представленная в схеме БД следующим образом (рис. 11).

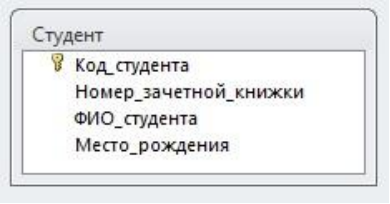


Рис. 11. Таблица «Студент» в схеме данных

Предложение CONSTRAINT используется в инструкциях ALTER TABLE и CREATE TABLE для создания или удаления индексов (индексы – это специальные объекты в БД, создаваемые с целью повышения быстродействия выполнения запросов, оптимизации хранения и доступа к данным, об индексах будет рассказано в следующем разделе учебного пособия). Существуют два типа предложений

CONSTRAINT: для создания простого индекса – по одному полю и для создания составного индекса – по нескольким полям Синтаксис:

простой индекс:

CONSTRAINT имя {PRIMARY KEY|UNIQUE | NOT NULL]
[ON UPDATE {CASCADE | SET NULL}]
[ON DELETE {CASCADE | SET NULL}]}

составной индекс:

CONSTRAINT имя
{PRIMARY KEY (ключевое_1[, ключевое_2 [, ...]]) |
UNIQUE (уникальное_1[, уникальное_2 [, ...]]) |
NOT NULL (непустое_1[, непустое_2 [, ...]]) |
FOREIGN KEY (ссылка_1[, ссылка_2 [, ...]])
REFERENCES внешняя Таблица [(внешнее Поле_1 [, внешнее Поле_2 [, ...]])]
[ON UPDATE {CASCADE | SET NULL}]
[ON DELETE {CASCADE | SET NULL}]} ,

где имя — имя индекса, который следует создать;

ключевое_1, ключевое_2 – имена одного или нескольких полей, которые следует назначить ключевыми;

уникальное_1, уникальное_2 – имена одного или нескольких полей, которые следует включить в уникальный индекс;

непустое_1, непустое_2 – имена одного или нескольких полей, в которых запрещаются значения Null;

ссылка_1, ссылка_2 – имена одного или нескольких полей, включенных во внешний ключ, которые содержат ссылки на поля в другой таблице;

внешняя Таблица – имя внешней таблицы, которая содержит поля, указанные с помощью аргумента внешнееПоле;

внешнее Поле_1, внешнее Поле_2 – имена одного или нескольких полей во внешней Таблице, на которые ссылаются поля, указанные с помощью аргумента ссылка_1, ссылка_2. Это предложение можно опустить, если данное поле является ключом внешней Таблицы.

ON UPDATE, ON DELETE обеспечивают автоматическое указание каскадного обновления связанных полей и каскадного удаления связанных записей в схеме БД (CASCADE) или обнуление соответствующих значений полей, являющихся внешними ключами (SET NULL). Данные ключевые слова актуальны, если в СУБД MS Access включить поддержку ANSI SQL, в противном случае в результате выполнения запроса будет выдана ошибка синтаксиса.

Предложение CONSTRAINT позволяет создать для поля индекс одного из двух описанных ниже типов:

1) уникальный индекс, использующий для создания зарезервированное слово UNIQUE. Это означает, что в таблице не может быть двух записей, имеющих одно и то же значение в этом поле. Уникальный индекс создается для любого поля или любой группы полей. Если в таблице определен составной уникальный индекс, то комбинация значений включенных в него полей должна быть уникальной для каждой записи таблицы, хотя отдельные поля и могут иметь совпадающие значения;
2) ключ таблицы, состоящий из одного или нескольких полей, использующий зарезервированные слова PRIMARY KEY. Все значения ключа таблицы должны быть уникальными и не значениями Null. Кроме того, в таблице может быть только один ключ.

Для создания внешнего ключа можно использовать зарезервированную конструкцию **FOREIGN KEY**. Если ключ внешней таблицы состоит из нескольких полей, необходимо использовать предложение CONSTRAINT. При этом следует перечислить все поля, содержащие ссылки на поля во внешней таблице, а также указать имя внешней таблицы и имена полей внешней таблицы, на которые ссылаются поля, перечисленные выше, причем в том же порядке. Однако, если последние поля являются ключом внешней таблицы, то указывать их необязательно, поскольку ядро базы данных считает, что в качестве этих полей следует использовать поля, составляющие ключ внешней таблицы.

В следующем примере создается таблица «Задолженность_за_обучение» с внешним ключом «Код_студента», связанным с полем «Код_студента», в таблице «Студент»:

```
CREATE TABLE Задолженность_за_обучение
(Код_задолженности AUTOINCREMENT PRIMARY KEY,
Код_студента INTEGER, Сумма_задолженности MONEY,
CONSTRAINT f1_i FOREIGN KEY (Код_студента)
REFERENCES Студент (Код_студента)
ON UPDATE CASCADE ON DELETE CASCADE );
```

Внешний вид схемы БД, состоящей из таблиц «Студент» и «Задолженность_за_обучение», представлен на рисунке 12. Здесь необходимо отметить, что дополнительные ключевые слова ON UPDATE CASCADE ON DELETE CASCADE позволили обеспечить автоматическое указание каскадного обновления связанных полей и каскадного удаления связанных записей в схеме БД.

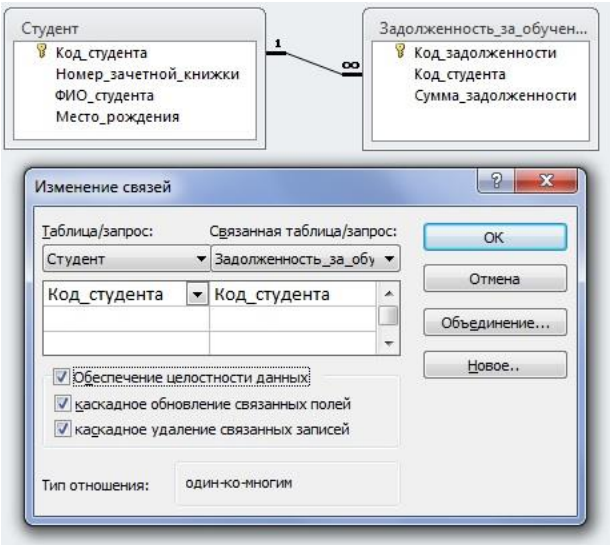


Рис. 12. Схема данных после создания таблицы «Задолженность_за_обучение»

Создание индекса с помощью инструкции CREATE INDEX

CREATE INDEX создает новый индекс для существующей таблицы. Синтаксис команды:

CREATE [UNIQUE] **INDEX** индекс
ON таблица (поле [ASC|DESC][, поле [ASC|DESC], ...])
[WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }]

где индекс – имя создаваемого индекса;

таблица – имя существующей таблицы, для которой создается индекс;

поле – имена одного или нескольких полей, включаемых в индекс. Для создания простого индекса, состоящего из одного поля, вводится имя поля в круглых скобках сразу после имени таблицы. Для создания составного индекса, состоящего из нескольких полей, перечисляются имена всех этих полей. Для расположения элементов индекса в убывающем порядке используется зарезервированное слово DESC; в противном случае будет принят порядок по возрастанию.

Чтобы запретить совпадение значений индексированных полей в разных записях, используется зарезервированное слово UNIQUE. Необязательное предложение WITH позволяет задать условия на значения. Например:

с помощью параметра DISALLOW NULL можно запретить значения Null в индексированных полях новых записей;

параметр IGNORE NULL позволяет запретить включение в индекс записей, имеющих значения Null в индексированных полях;

зарезервированное слово PRIMARY позволяет назначить индексированные поля ключом. Такой индекс по умолчанию является уникальным, следовательно, зарезервированное слово UNIQUE можно опустить.

В следующем примере создается уникальный индекс, не допускающий ввод повторяющихся значений в поле «Номер зачетной книжки» в таблице «Студент»:

CREATE UNIQUE INDEX New_index ON

Студент (Номер_зачетной_книжки) WITH IGNORE NULL

Ключевые слова IGNORE NULL позволяют добавлять в таблицу записи по студентам без обязательного ввода номера зачетной книжки (рис.13). Нажатие пиктограммы «Индексы» в конструкторе таблиц открывает одноименное окно с перечнем индексов для данной таблице, где отображается созданный в результате выполнения запроса уникальный индекс New_index.

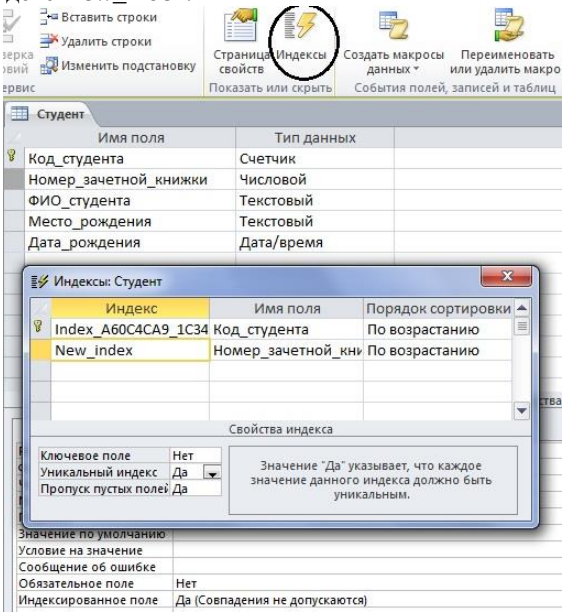


Рис. 13 Результат создания уникального индекса

Создание запроса на удаление таблицы/индекса

Инструкция **DROP** удаляет существующую таблицу из базы данных или удаляет существующий индекс из таблицы. Синтаксис:

DROP {TABLE таблица | INDEX индекс ON таблица}

где таблица – имя таблицы, которую следует удалить или из которой следует удалить индекс;

индекс – имя индекса, удаляемого из таблицы.

Прежде чем удалить таблицу или удалить из нее индекс, необходимо ее закрыть. Следует отметить, что таблица и индекс удаляются из базы данных безвозвратно.

В следующем примере удалим созданный в предыдущем примере индекс «New_index»:

DROP INDEX New_index ON Студент;

Создание запроса на добавление записей

Инструкция **INSERT INTO** добавляет запись или несколько записей в заданную таблицу.

Синтаксис команды:

а) запрос на добавление нескольких записей:

INSERT INTO назначение [(поле_1[, поле_2[, ...]])]

SELECT [источник.]поле_1[, поле_2[, ...]

FROM выражение

б) запрос на добавление одной записи:

INSERT INTO назначение [(поле_1[, поле_2[, ...]])]

VALUES (значение_1[, значение_2[, ...])

где назначение — имя таблицы или запроса, в который добавляются записи;

источник – имя таблицы или запроса, откуда копируются записи;

поле_1, поле_2 – имена полей для добавления данных, если они следуют за аргументом «Назначение»; имена полей, из которых берутся данные, если они следуют за аргументом источник;

выражение – имена таблицы или таблиц, откуда вставляются данные. Это выражение может быть именем отдельной таблицы или результатом операции INNER JOIN, LEFT JOIN или RIGHT JOIN, а также сохраненным запросом;

значение_1, значение_2 – значения, добавляемые в указанные поля новой записи. Каждое значение будет вставлено в поле, занимающее то же положение в списке: значение_1 вставляется в поле_1 в новой записи, значение_2 – в поле_2 и т.д. Каждое значение текстового поля следует заключать в кавычки (' '), для разделения значений используются запятые.

Инструкцию INSERT INTO можно использовать для добавления одной записи в таблицу с помощью запроса на добавление одной записи, описанного выше. В этом случае инструкция должна содержать имя и значение каждого поля записи. Нужно определить все поля записи, в которые будет помещено значение, и значения для этих полей. Если поля не определены, в недостающие столбцы будет вставлено значение по умолчанию или значение Null. Записи добавляются в конец таблицы.

Инструкцию INSERT INTO можно также использовать для добавления набора записей из другой таблицы или запроса с помощью предложения SELECT ... FROM, как показано выше в запросе на добавление нескольких записей. В этом случае предложение SELECT определяет поля, добавляемые в указанную таблицу Назначение. Инструкция INSERT INTO является необязательной, однако, если она присутствует, то должна находиться перед инструкцией SELECT.

Запрос на добавление записей копирует записи из одной или нескольких таблиц в другую таблицу. Таблицы, которые содержат добавляемые записи, не изменяются.

Вместо добавления существующих записей из другой таблицы, можно указать значения полей одной новой записи с помощью предложения VALUES. Если список полей опущен, предложение VALUES должно содержать значение для каждого поля таблицы; в противном случае инструкция INSERT не будет выполнена. Можно использовать дополнительную инструкцию INSERT INTO с предложением VALUES для каждой добавляемой новой записи.

В следующем примере добавим новую запись в таблицу «Студент»:

```
INSERT INTO Студент (Номер_зачетной_книжки,
ФИО_студента, Место_рождения, Дата_рождения )
VALUES (201454321, 'Иванов Иван Петрович', 'г. Томск',
'12.02.1996');
```

Заметим, что поскольку поле «Код_студента» имеет тип данных «Счетчик» и формируется автоматически, то в перечне новых значений для полей мы его опускаем. На рисунке 14 представлена таблица «Студент» с новой записью.

Студент

Код_студента	Номер_зачетной_книжки	ФИО_студента	Место_рождения	Дата_рождения
1	201454321	Иванов Иван Петро	г. Томск	12.02.1996

Рис. 14 Результат добавления записи в таблицу «Студент»

Аналогично можно добавить данные в таблицу «Задолженность_за обучение».

Создание запроса на обновление данных

Инструкция **UPDATE** создает запрос на обновление, который изменяет значения полей указанной таблицы на основе заданного условия отбора.

Синтаксис команды:

UPDATE таблица
SET новое Значение
WHERE условие Отбора;

где таблица — имя таблицы, данные в которой следует изменить;

новое Значение — выражение, определяющее значение, которое должно быть вставлено в указанное поле обновленных записей;

условие Отбора — выражение, отбирающее записи, которые должны быть изменены.

При выполнении этой инструкции будут изменены только записи, удовлетворяющие указанному условию. Инструкцию UPDATE особенно удобно использовать для изменения сразу нескольких записей или в том случае, если записи, подлежащие изменению, находятся в разных таблицах. Одновременно можно изменить значения нескольких полей. Следующая инструкция SQL увеличивает сумму задолженности всех студентов, по которым есть сведения в таблице «Задолженность за обучение» на 10%:

UPDATE Задолженность_за_обучение
SET Сумма_задолженности = Сумма_задолженности * 1.1;

Создание запроса на выборку записей

Инструкция **SELECT**. При выполнении инструкции SELECT СУБД находит указанную таблицу или таблицы, извлекает заданные столбцы, выделяет строки, соответствующие условию отбора, и сортирует или группирует результирующие строки в указанном порядке в виде набора записей.

Синтаксис команды:

SELECT [предикат] { * | таблица.* | [таблица.]поле_1
[AS псевдоним_2] [, [таблица.]поле_2[AS псевдоним_2] [, ...]]}
FROM выражение [, ...]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]

где предикат – один из следующих предикатов отбора: ALL, DISTINCT, DISTINCTROW, TOP. Данные ключевые слова используются для ограничения числа возвращаемых записей. Если они отсутствуют, по умолчанию используется предикат ALL;

* указывает, что результирующий набор записей будет содержать все поля заданной таблицы или таблиц. Следующая инструкция отбирает все поля из таблицы «Студенты»: SELECT * FROM Студенты;

таблица – имя таблицы, из которой выбираются записи;

поле_1, поле_2 – имена полей, из которых должны быть отобраны данные;

псевдоним_1, псевдоним_2 – ассоциации, которые станут заголовками столбцов вместо исходных названий полей в таблице;

выражение – имена одной или нескольких таблиц, которые содержат необходимые для отбора записи;

предложение GROUP BY в SQL-предложении объединяет записи с одинаковыми значениями в указанном списке полей в одну запись. Если инструкция SELECT содержит статистическую функцию SQL, например Sum или Count, то для каждой записи будет вычислено итоговое значение;

предложение HAVING определяет, какие сгруппированные записи, выданные в результате выполнения запроса, отображаются при использовании инструкции SELECT с предложением GROUP BY. После того как записи результирующего набора будут сгруппированы с помощью предложения GROUP BY, предложение HAVING отберет те из них, которые удовлетворяют условиям отбора, указанным в предложении HAVING;

предложение ORDER BY позволяет отсортировать записи, полученные в результате запроса, в порядке возрастания или убывания

на основе значений указанного поля или полей.

Следует отметить, что инструкции SELECT не изменяют данные в базе данных.

Для более наглядного представления результатов выполнения запросов на выборку добавим в нашу базу данных таблицы «Дисциплины» и «Успеваемость» и добавим поле «Номер_группы» в таблицу «Студенты» (рис. 15), а также заполним эти таблицы.

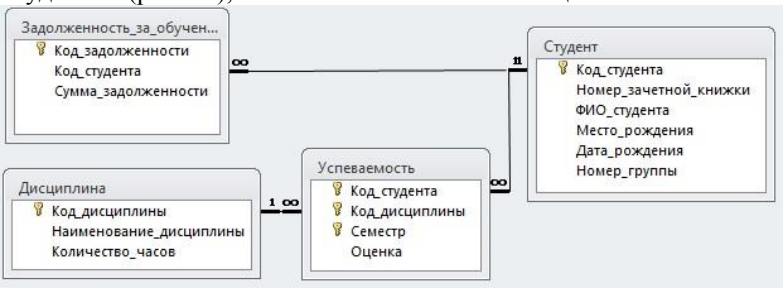


Рис. 15 Обновленная схема базы данных

На рисунке 16 приведено содержимое таблиц «Студент», «Успеваемость» и «Дисциплина»

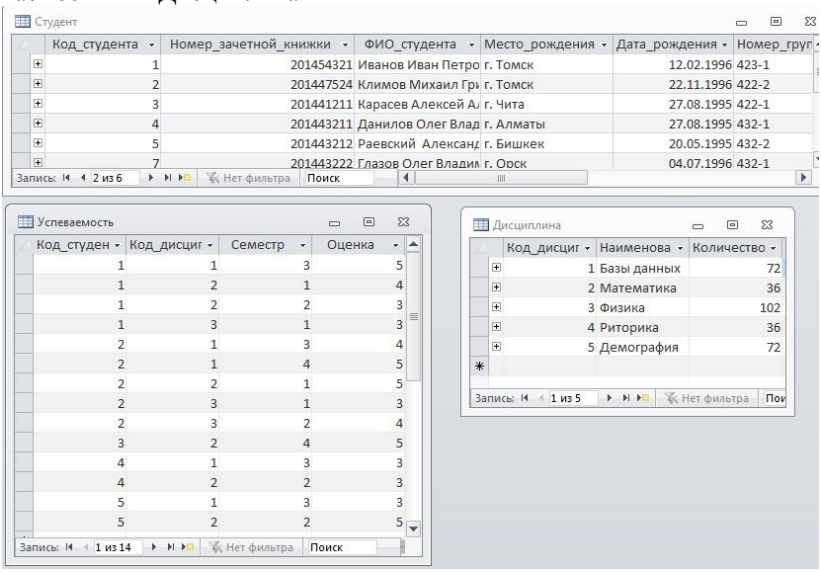


Рис. 16 Содержимое таблиц базы данных

В следующем примере представлен простейший запрос на выборку записей из таблицы студенты, в котором отбираются студенты, обучающиеся в группе 432-1:

```
SELECT Студент.Номер_зачетной_книжки,
Студент.ФИО_студента, Студент.Номер_группы
FROM Студент
WHERE Студент.Номер_группы='432-1';
```

На рисунке 17 представлен результат выполнения этого запроса, были найдены 2 записи, удовлетворяющие критерию запроса.

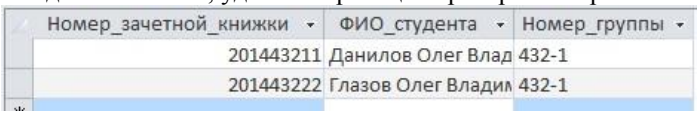


Рис. 17 Результат выполнения запроса

Помимо обычных знаков сравнения (=,<,>,<=,>=, <>) в языке SQL в условии отбора используются ряд ключевых слов:

Is not null — выбрать только непустые значения;

Is null — выбрать только пустые значения;

Between … And определяет принадлежность значения выражения указанному диапазону.

Синтаксис:

выражение [Not] Between значение_1 And значение_2 ,

где выражение — выражение, определяющее поле, значение которого проверяется на принадлежность к диапазону;

значение_1, значение_2 – выражения, задающие границы диапазона.

Если значение поля, определенного в аргументе выражение, попадает в диапазон, задаваемый аргументами значение_1 и значение_2 (включительно), то оператор Between...And возвращает значение True; в противном случае возвращается значение False. Логический оператор Not позволяет проверить противоположное условие: что выражение находится за пределами диапазона, заданного с помощью аргументов значение_1 и значение_2.

Оператор Between...And часто используют для проверки: попадает ли значение поля в указанный диапазон чисел.

Если выражение, значение_1 или значение_2 имеет значение Null, оператор Between...And возвращает значение Null.

Создание запроса на выборку с внутренним соединением

Операция **INNER JOIN** объединяет записи из двух таблиц, если связующие поля этих таблиц содержат одинаковые значения.

Синтаксис операции:

FROM таблица_1 INNER JOIN таблица_2 ON таблица_1.поле_1 оператор таблица_2.поле_2

где таблица_1, таблица_2 — имена таблиц, записи которых подлежат объединению;

поле_1, поле_2 — имена объединяемых полей. Поля должны иметь одинаковый тип данных и содержать данные одного рода,

однако эти поля могут иметь разные имена;

оператор — любой оператор сравнения: "=," "<," ">," "<=," ">=," или "<>".

Операцию INNER JOIN можно использовать в любом предложении FROM. Это самые обычные типы связывания. Они объединяют записи двух таблиц, если связующие поля обеих таблиц содержат одинаковые значения.

В следующем примере составим запрос, выдающий сведения об успеваемости студентов по всем изученным ими дисциплинам:

```
SELECT Студент.ФИО_студента as ФИО, Дисциплина.
Наименование_дисциплины as Предмет, Успеваемость. Семестр,
Успеваемость.Оценка
FROM Студент INNER JOIN
(Дисциплина INNER JOIN Успеваемость ON Дисциплина.
Код_дисциплины = Успеваемость.Код_дисциплины)
```

ON Студент.Код_студента = Успеваемость.Код_студента;

На рисунке 18 представлен результат выполнения этого запроса.

ФИО_студента	Наименова	Семестр	Оценка
Иванов Иван Петрович	Базы данных	3	5
Иванов Иван Петрович	Математика	1	4
Иванов Иван Петрович	Математика	2	3
Иванов Иван Петрович	Физика	1	3
Климов Михаил Григорьевич	Базы данных	3	4
Климов Михаил Григорьевич	Базы данных	4	5
Климов Михаил Григорьевич	Математика	1	5
Климов Михаил Григорьевич	Физика	1	3
Климов Михаил Григорьевич	Физика	2	4
Карасев Алексей Александрови	Математика	4	5
Данилов Олег Владимирович	Базы данных	3	3
Данилов Олег Владимирович	Математика	2	3
Раевский Александр Иванович	Базы данных	3	3
Раевский Александр Иванович	Математика	2	5

Запись: 2 из 14 | Нет фильтра | Поиск

Рис. 18 Результат выполнения запроса с внутренним соединением

Создание запроса на выборку с внешним соединением

Операции **LEFT JOIN, RIGHT JOIN** объединяют записи исходных таблиц при использовании в любом предложении FROM.

Операция LEFT JOIN используется для создания внешнего соединения, при котором все записи из первой (левой) таблицы включаются в результирующий набор, даже если во второй (правой) таблице нет соответствующих им записей.

Операция RIGHT JOIN используется для создания внешнего объединения, при котором все записи из второй (правой) таблицы включаются в результирующий набор, даже если в первой (левой) таблице нет соответствующих им записей.

Синтаксис операции:

FROM таблица_1 [**LEFT | RIGHT**] **JOIN** таблица_2

ON таблица_1.поле_1 оператор таблица_2.поле_2

В следующем примере в результате выполнения запроса будут найдены все студенты и их успеваемость, при этом в результирующий набор данных будут также включены студенты, чьи сведения об успеваемости отсутствуют:

SELECT Студент.ФИО_студента, Дисциплина.
Наименование_дисциплины, Успеваемость.Семестр, Успеваемость.
Оценка

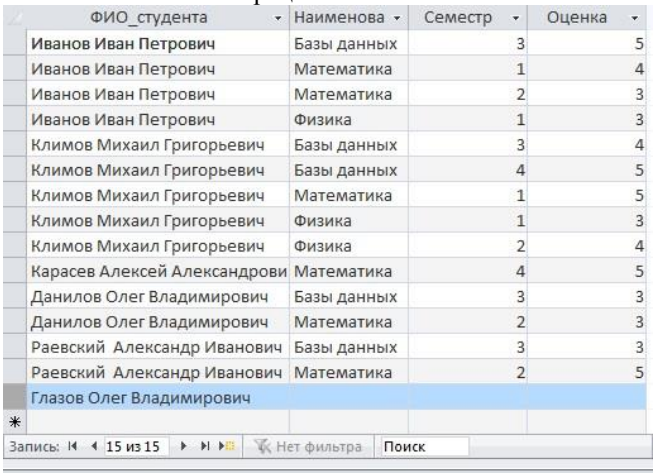
FROM Студент LEFT JOIN

(Дисциплина RIGHT JOIN Успеваемость ON Дисциплина.Код_дисциплины = Успеваемость.Код_дисциплины)

ON Студент.Код_студента = Успеваемость.Код_студента;

На рисунке 19 представлен результат выполнения этого запроса, где видно, что по сравнению с предыдущим примером добавлена еще одна запись о студенте Глазове Олеге Владимировиче, .

Важно отметить, что операции LEFT JOIN или RIGHT JOIN могут быть вложены в операцию INNER JOIN, но операция INNER JOIN не может быть вложена в операцию LEFT JOIN или RIGHT JOIN.



ФИО_студента	Наименова	Семестр	Оценка
Иванов Иван Петрович	Базы данных	3	5
Иванов Иван Петрович	Математика	1	4
Иванов Иван Петрович	Математика	2	3
Иванов Иван Петрович	Физика	1	3
Климов Михаил Григорьевич	Базы данных	3	4
Климов Михаил Григорьевич	Базы данных	4	5
Климов Михаил Григорьевич	Математика	1	5
Климов Михаил Григорьевич	Физика	1	3
Климов Михаил Григорьевич	Физика	2	4
Карасев Алексей Александрови	Математика	4	5
Данилов Олег Владимирович	Базы данных	3	3
Данилов Олег Владимирович	Математика	2	3
Раевский Александр Иванович	Базы данных	3	3
Раевский Александр Иванович	Математика	2	5
Глазов Олег Владимирович			

Запись: 15 из 15 Нет фильтра Поиск

Рис. 19 Результат выполнения запроса с внешним соединением

Создание перекрестного запроса

В Jet SQL существует такой вид запросов как перекрестный. В таком запросе отображаются результаты статистических функций – суммы, средние значения и др., а также количество записей. При этом подсчет выполняется по данным одного из полей таблицы. Результаты группируются по двум наборам данных, один из которых расположен в левом столбце таблицы, а другой в заголовке таблицы. Например, при необходимости вычислить средний балл студентов за семестр, обучающихся в разных группах, необходимо реализовать перекрестный запрос, результат выполнения которого будет представлен в виде таблицы, где заголовками строк будут служить название предмета, заголовками столбцов – номера групп, а в полях таблицы будет рассчитан средний балл студентов группы по каждому предмету.

Для создания перекрестного запроса необходимо использовать следующую инструкцию:

TRANSFORM статистическая_функция

инструкция_SELECT

PIVOT поле [IN (значение_1[, значение_2[, ...]])],

где статистическая_функция — статистическая функция SQL, обрабатывающая указанные данные;

инструкция_SELECT — запрос на выборку;

поле — поле или выражение, которое содержит заголовки столбцов для результирующего набора;

значение_1, значение_2 — фиксированные значения, используемые при создании заголовков столбцов.

Составим SQL-запрос, реализующий описанный выше пример:

TRANSFORM Avg(Успеваемость.Оценка)

SELECT Дисциплина.Наименование_дисциплины as [Предмет]

FROM Студент INNER JOIN

(Дисциплина INNER JOIN Успеваемость ON Дисциплина.Код_дисциплины = Успеваемость.Код_дисциплины) ON Студент.Код_студента = Успеваемость.Код_студента

GROUP BY Дисциплина.Наименование_дисциплины, Успеваемость.Код_дисциплины

PIVOT Студент.Номер_группы;

В результате выполнения такого перекрестного SQL-запроса формируется следующий набор данных (рис. 20):

Предмет	422-1	422-2	423-1	432-1	432-2
Базы данных		4,5	5	3	3
Математика	5	5	3,5	3	5
Физика		3,5	3		

Запись: 2 из 3 Нет фильтра Поиск

Рис. 20 Результат выполнения перекрестного запроса

Таким образом, когда данные сгруппированы с помощью перекрестного запроса, можно выбирать значения из заданных столбцов или выражений и определять как заголовки столбцов. Это позволяет просматривать данные в более компактной форме, чем при работе с обычным запросом на выборку. Отметим, что перекрестные запросы удобно использовать для формирования статистических отчетов.

Создание запроса на удаление записей

Инструкция **DELETE** создает запрос на удаление записей из одной или нескольких таблиц, перечисленных в предложении FROM и удовлетворяющих предложению WHERE.

Синтаксис команды:

DELETE [Таблица.*]

FROM таблица

WHERE условие Отбора

где Таблица — необязательное имя таблицы, из которой удаляются записи;

таблица — имя таблицы, из которой удаляются записи;

условие Отбора — выражение, определяющее удаляемые записи.

С помощью инструкция DELETE можно осуществлять удаление большого количества записей. Данные из таблицы также можно удалить и с помощью инструкции DROP, однако при таком удалении теряется структура таблицы. Если же применить инструкцию DELETE, удаляются только данные. При этом сохраняются структура таблицы и все остальные ее свойства, такие, как атрибуты полей и индексы.

В следующем примере из таблицы «Абитуриент» будет удален абитуриент Авдеев Н.В.:

```
DELETE
FROM Абитуриент
WHERE ФИО_Абитуриента = 'Авдеев Н.В.'
```

Запрос на удаление удаляет записи целиком, а не только содержимое указанных полей. Нельзя восстановить записи, удаленные с помощью запроса на удаление. Чтобы узнать, какие записи будут удалены, необходимо посмотреть результаты запроса на выборку, использующего те же самые условие отбора в предложении Where, а затем выполнить запрос на удаление.

Порядок выполнения работы

В ходе выполнения лабораторной работы необходимо создать с помощью построителя запросов:

1. Запрос на выборку,
2. Запрос на выборку с параметрами;
3. Запрос на обновление данных;
4. Запрос на удаление записей.

Следующие должны быть реализованы на языке SQL без помощи построителя запросов:

5. Используя инструкцию CREATE TABLE, создайте запрос на создание новой таблицы для выбранной ранее предметной области, содержащую пять полей, различных типов данных, определив в запросе первичный ключ и проиндексировав соответствующие поля, используя предложение CONSTRAINT.

   Для запуска запроса нажмите кнопку Запуск  на панели инструментов. После чего создайте запрос на создание еще одной таблицы, содержащей внешний ключ по отношению к первичному ключу предыдущей таблицы. Запустите запрос, после чего проверьте, отразились ли изменения в схеме данных.

6. Используя команду CREATE INDEX, создайте запрос на создание нового индекса, используя различные условия на значения индексов (IGNORE NULL, DISSALLOW NULL, PRIMARY), а также типы сортировки.
7. Используя команду INSERT INTO, создайте запросы на добавление группы записей (из дополнительной таблицы) и одной записи в существующую таблицу.
8. Используя команду UPDATE, создайте запрос на обновление данных в созданных ранее таблицах.
9. Используя команду SELECT, создайте запрос на выборку записей из двух (или более) таблиц, используя правила внешнего и внутреннего соединения, а также различные условия отбора и сортировки.
10. Используя команду TRANSFORM, создайте перекрестный запрос.
11. Используя команду DROP, создайте запросы на удаление таблицы и индекса, созданных ранее в БД.

Сохраните все созданные запросы в базе данных.

Лабораторная работа № 3 «Создание экранных форм в СУБД MS Access»

Тема: Создание форм в СУБД MS Access. Создание экранных форм и их использование для ввода данных.

Раздел дисциплины: Реляционная модель.

Цель работы: создать ленточную, табличную и сложную формы в базе данных MS Access, используя в качестве источника записей созданные ранее таблицы и запросы.

Продолжительность: 6 часов.

Формы MS Access

Формы являются типом объектов базы данных, и используются для отображения и ввода данных в таблицы БД. Форму можно также использовать как кнопочную форму, открывающую другие формы или отчеты БД, а также как пользовательское диалоговое окно.

Обычно для формы выбирается источник записей. В базе данных Microsoft Access источником записей может быть таблица, запрос или инструкция SQL. В проекте Microsoft Access источником записей может быть таблица, представление, инструкция SQL или сохраненная процедура. Другие выводящиеся в форме сведения, такие как заголовок, дата и номера страниц и др., сохраняются в макете формы.

Связь между формой и ее источником записей создается при помощи графических объектов, которые называют элементами управле-

ния – это объекты графического интерфейса пользователя (такие как поле, флажок, полоса прокрутки или кнопка), позволяющий пользователям управлять приложением. Элементы управления используются для отображения данных или параметров, для выполнения действий, либо для упрощения работы с интерфейсом пользователя. Наиболее часто используемым для вывода и ввода данных типом элементов управления является поле.

Существует несколько типов форм: формы можно открывать в виде таблицы, линейной формы и в режиме простой формы. В этих режимах пользователи могут динамически изменять макет формы для изменения способа представления данных. Существует возможность упорядочивать заголовки строк и столбцов, а также применять фильтры к полям. При каждом изменении макета сводная форма немедленно выполняет вычисления заново в соответствии с новым расположением данных.

Форму можно создать тремя различными способами:

– При помощи *автоформы* на основе таблицы или запроса. С помощью автоформ можно создавать формы, в которых выводятся все поля и записи базовой таблицы или запроса. Если выбранный источник записей имеет связанные таблицы или запросы, то в форме также будут присутствовать все поля и записи этих источников записей.

– При помощи *мастера* на основе одной или нескольких таблиц или запросов. Мастер задает подробные вопросы об источниках записей, полях, макете, требуемых форматах и создает форму на основании полученных ответов.

– Вручную в режиме *конструктора*. Сначала создается базовая форма, которая затем изменяется в соответствии с требованиями в режиме конструктора.

В ходе выполнения лабораторной работы должны быть созданы *ленточная, табличная и сложная (содержащая подчиненную)* формы.

Ленточная форма – это форма, в которой поля, образующие одну запись, расположены в одной строке; их подписи выводятся один раз в верхней части (заголовке) формы.

Табличная форма – это форма, в которой поля записей расположены в формате таблицы, где каждой записи соответствует одна строка, а каждому полю один столбец. Имена полей служат заголовками столбцов.

Подчиненной формой называют форму, вставленную в другую форму. Первичная форма называется *главной* формой, а форма внутри формы называется *подчиненной*. Комбинацию «форма/подчиненная

форма» часто называют также иерархической формой или комбинацией «родительской» и «дочерней» форм.

Подчиненные формы особенно удобны для вывода данных из таблиц или запросов, связанных с отношением «один-ко-многим». Например, можно создать форму с подчиненной формой для вывода данных из таблицы «Типы» и из таблицы «Товары». Данные в таблице «Типы» находятся на стороне «один» отношения. Данные в таблице «Товары» находятся на стороне «многие» отношения — каждый тип может иметь несколько товаров. В главной форме отображаются данные на стороне отношения «один». В подчиненной форме отображаются данные на стороне отношения «многие».

Главная форма и подчиненная форма в этом типе форм связаны таким образом, что в подчиненной форме выводятся только те записи, которые связаны с текущей записью в главной форме. Например, когда главная форма отображает тип «Напитки», подчиненная форма отображает только те товары, которые входят в тип «Напитки».

Порядок выполнения работы

Создание ленточной табличной и макета сложной формы следует осуществлять с помощью мастера форм. Мастер задает подробные вопросы об источниках записей, полях, макете, требуемых форматах и создает форму на основании полученных ответов.

Для создания ленточной формы с помощью мастера форм в окне базы данных перейдите на вкладку Формы и выберите пункт Создание формы с помощью мастера. В предложенном окне (рис. 21) выберите таблицу или созданный ранее запрос на выборку, который будет использоваться в качестве источника записей формы, выберите поля таблицы/запроса, которые будут доступны для редактирования в создаваемой форме. Затем нажмите кнопку Далее.

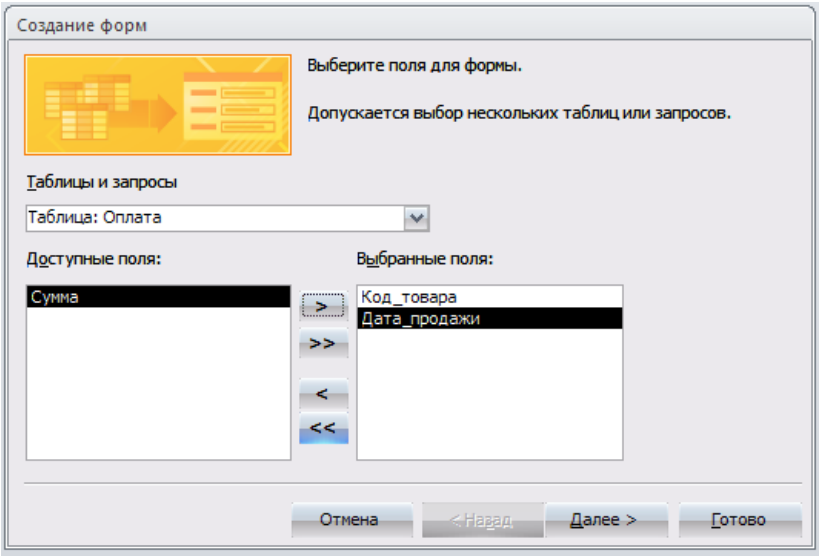


Рис. 21. Первое окно мастера форм

В следующем окне (рис. 22) выберите тип формы Ленточный, нажмите кнопку Далее и следуйте дальнейшим указаниям мастера форм.

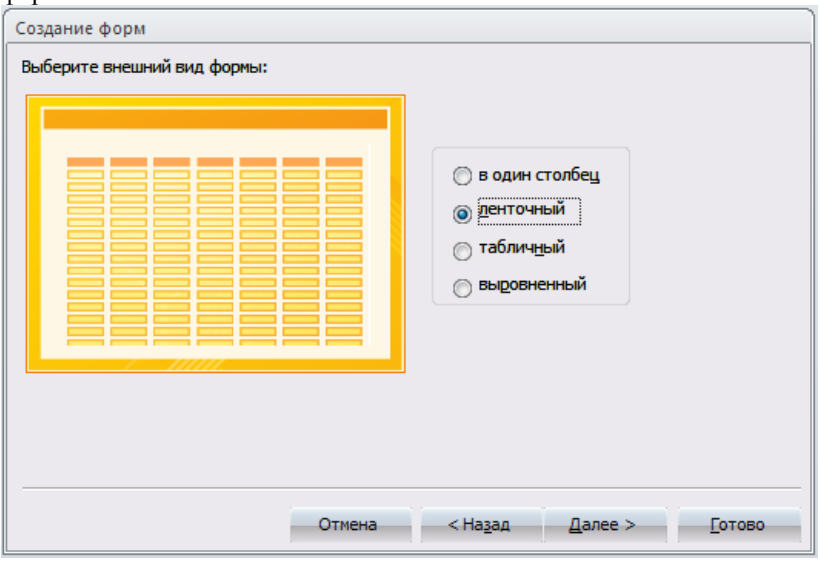


Рис. 22 Второе окно мастера форм

Для создания табличной формы с помощью мастера, следуйте предложенным выше инструкциям, а в окне (рис. 22.) выберите тип формы *табличный*. При этом рекомендуется табличную форму создавать с учетом того, что она может являться подчиненной формой в сложной форме, т.е. источником записей должна быть таблица, содержащая внешний ключ, т.е. связанная с другой таблицей М:1.

Для создания главной формы в окне (рис. 22.) выберите тип формы *в один столбец*, учитывая, что источником записей должна быть таблица, соединенная в схеме данных с подчиненной таблицей с типом связи 1:М.

Созданные формы будут отображены в окне базы данных в разделе Формы, их макет можно изменить в режиме конструктора для чего необходимо выбрать нужную форму и нажать кнопку Конструктор в окне БД.

В созданной ленточной форме создайте кнопки для навигации по записям и кнопку закрытия формы. Для этого откройте форму в режиме конструктора и в панели инструментов выберите объект Кнопка (рис. 23) и поместите её в область примечания формы.

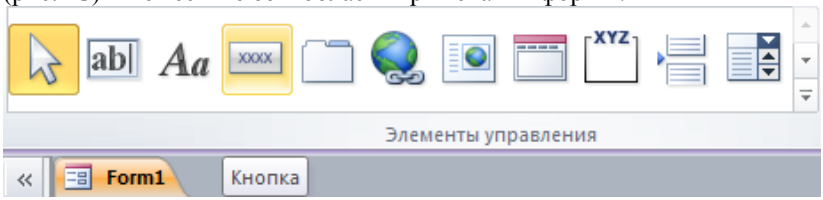


Рис. 23. Панель элементов

После этого откроется окно мастера кнопок (рис. 24) в котором необходимо выбрать категорию *переходы по записям* и в действиях – *первая запись* для перехода к первой записи набора данных, затем нажмите кнопку *Далее* и следуйте дальнейшим инструкциям мастера.

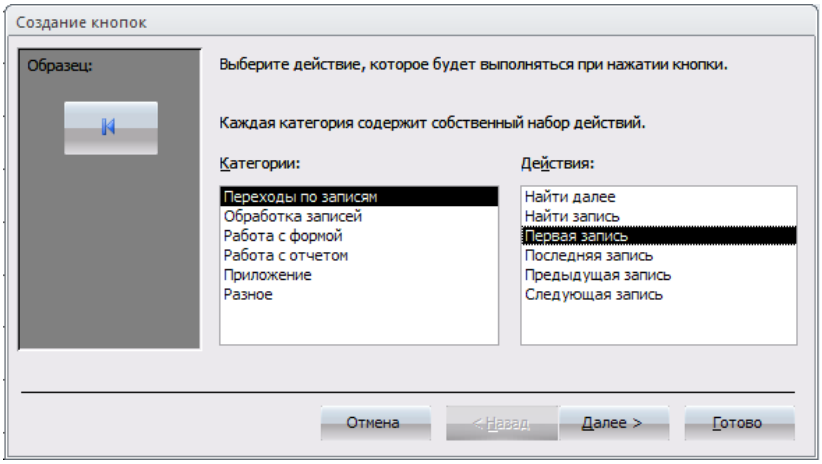


Рис. 24. Мастер кнопок

Аналогично создайте кнопки для перехода на последнюю, предыдущую и следующую запись. При создании кнопки для закрытия формы в мастере кнопок выберите категорию *работа с формой* и действие *закрытие формы*.

Создать подчиненную форму можно либо на основе имеющихся в БД таблиц и запросов либо на основе созданных ранее форм, учитывая при этом, что и главная, и подчиненная формы должны иметь общие поля, необходимые для установления связи между ними.

Для создания подчиненной формы откройте созданную ранее главную форму в режиме конструктора и в панели инструментов выберите объект подчиненная форма и вставьте его в главную форму, поле чего откроется окно мастера подчиненных форм (рис. 25)

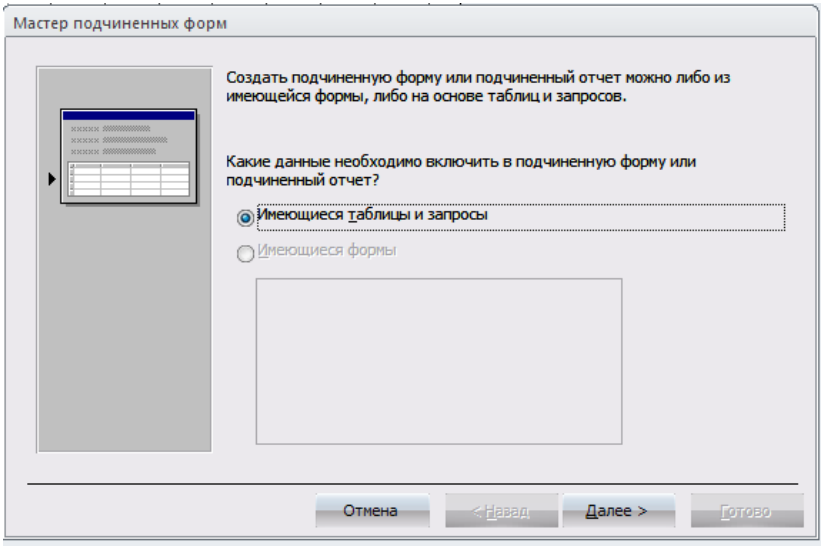


Рис. 25. Мастер подчиненных форм

Выберите из списка предложенных форм созданную ранее форму в табличном виде, при условии, что она удовлетворяет всем требованиям для подчиненных форм, в противном случае создайте подчиненную форму на основе имеющихся таблиц и запросов. Нажмите кнопку *Далее* и следуйте указаниям мастера.

В главной форме создайте кнопки перехода по записям и кнопку закрытия формы, так же как и для ленточной формы, кроме того, создайте кнопку *Добавить новую запись*, по нажатию которой добавлялась бы пустая запись в главную форму, для этого в мастере кнопок выберите категорию *обработка записей* и действие *добавить запись*. Создайте также кнопку, нажатие которой вызовет открытие созданной ранее ленточной формы, для этого в мастере кнопок выберите категорию *работа с формой* и действие *открытие формы*.

4 Самостоятельная работа

Согласно рабочим программам самостоятельная работа составляет:

для направления Бизнес-информатика составляет 251 час,

для направления программная инженерия – 160 часов,

для направления государственное и муниципальное управление – 119 часов.

Проверка самостоятельной работы и уровень освоения заявленных в рабочих программах компетенций осуществляется путем сдачи контрольных работ, защиты лабораторных работ и курсового проекта (при наличии такового).

Для изучения тем теоретической части курса, отводимых на самостоятельную подготовку и указанных в рабочих программах, а также и при подготовке к экзамену и/или зачету по дисциплине, рекомендуется ознакомиться со всеми разделами [1].

Кроме этого, рекомендуется повторить разделы предложенной литературы [2-5], посвященные обоснованию концепции баз данных, концепции модели данных, проектированию данных, языкам управления и манипулирования данными, построению пользовательских приложений.

5 Рекомендуемая литература

1. Сенченко П. В. Организация баз данных: учеб. пособие / П.В. Сенченко. — Томск: факультет дистанционного обучения ТУСУРа, 2015. — 170 с. ил. [Электронный ресурс]: сайт кафедры АОИ. – URL: http://aoi.tusur.ru/upload/methodical_materials/BD_posobie2015_file__614_3112.pdf

2. Сенченко, П.В. Организация баз данных: Учебное пособие/ П. В. Сенченко; Федеральное агентство по образованию, Томский государственный университет систем управления и радиоэлектроники. – Томск: ТУСУР, 2004. - 184 с.: ил.. - Библиогр.: с. 183-184. - ISBN 5-86889-224-0: УДК 681.3.016(075.8)

3. Дейт, К. Дж. Основы будущих систем баз данных. Третий манифест: Детальное исследование влияния теории типов на реляционную модель данных, включая полную модель наследования типов : Пер. с англ. / К. Дж. Дейт, Хью Дарвен; Ред. пер. С. Д. Кузнецов, Пер. С. Д. Кузнецов, Пер. Т. А. Кузнецова. - 2-е изд. - М. : Янус-К, 2004. - 655

4. Кузнецов С.Д. Основы современных баз данных. Информационно-аналитические материалы Центра Информационных технологий. М.– режим доступа к сайту http://citforum.ru/database/osbd/contents.shtml свободный (дата обращения: 19.11.2015)

5. Кириллов В.В. Основы проектирования реляционных баз данных. Учебное пособие режим доступа к сайту http://citforum.ru/database/dbguide/index.shtml свободный (дата обращения: 19.11.2015)