Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

ОРГАНИЗАЦИЯ БАЗ ДАННЫХ

Методические указания к лабораторным работам, курсовому проекту и организации самостоятельной работы для студентов направлений подготовки
«Программная инженерия»
«Бизнес-информатика»
(уровень бакалавриата)

Томск – 2022

Сенченко П.В.

Организация баз данных: Методические указания к лабораторным работам, курсовому проекту и организации самостоятельной работы для студентов направлений подготовки «Программная инженерия», «Бизнес-информатика» (уровень бакалавриата) / П.В. Сенченко. – Томск, 2022. – 80 с.

СОДЕРЖАНИЕ

1 Введение

Выполнение лабораторных работ, курсового проекта и самостоятельная работа направлены на приобретение навыков разработки баз данных, создания пользовательских SQL-запросов в среде СУБД MS Access и элементов пользовательского интерфейса, разработки моделей данных и программной документации студентами направления подготовки бакалавров «Программная инженерия» и «Бизнес-информатика».

В результате изучения дисциплины студент должен:

Знать:

- историю развития концепции баз данных;
- основные функции современных систем управления базами данных (СУБД);
- методы управления транзакциями;
- классификацию и характеристики моделей данных, лежащих в основе баз данных;
- теорию реляционных баз данных;
- операции реляционной алгебры и реляционное исчисление;
- целостную часть реляционной модели данных;
- методы проектирования реляционных баз данных с использованием нормализации;
- основы построения языков манипулирования данными SQL и QBE;
- синтаксис основных команд языка SQL;
- основные элементы и принципы построения моделей «Сущность-связь»;
- физическую организацию данных;
- принципы построения индексов;
- архитектуры представления баз данных (файл-серверную и клиент-серверную);
- современные тенденции в развитии концепции баз данных.
- объектно-ориентированных подход при организации баз данных.

Уметь:

- производить моделирование предметной области, уметь строить для нее ER-диаграмму и отображать ER-диаграмму в схему реляционной базы данных;
- разрабатывать все виды запросов на языке SQL и QBE;
- разрабатывать информационные системы для работы со сложно-структурированными базами данных: экранные формы, отчеты, разрабатывать для конкретного применения все виды запросов в выбранном диалекте языка SQL;

Владеть:

- методикой проектирования баз данных на основе нормализации отношений.
- методикой проектирования БД на основе разработки ER-модели предметной области.
- как минимум одним средством автоматизированного проектирования ER-диаграмм;
- навыками разработки сложных баз данных и пользовательских приложений с использованием функциональных возможностей современных СУБД (MS Access).

2 Методические указания к проведению лабораторных работ

2.1 Лабораторная работа «Организация хранения данных в MS Access»

Цель работы

Научиться разрабатывать структуру базы данных (БД) для выбранной предметной области, содержащую не менее восьми взаимосвязанных таблиц.

Форма проведения

Выполнение индивидуального задания.

Форма отчетности

На проверку должна быть представлена БД, содержащая не менее восьми взаимосвязанных таблиц, в каждой и з которых должно быть не менее 3-х записей.

Теоретические основы

Организация базы данных в среде MS Access

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

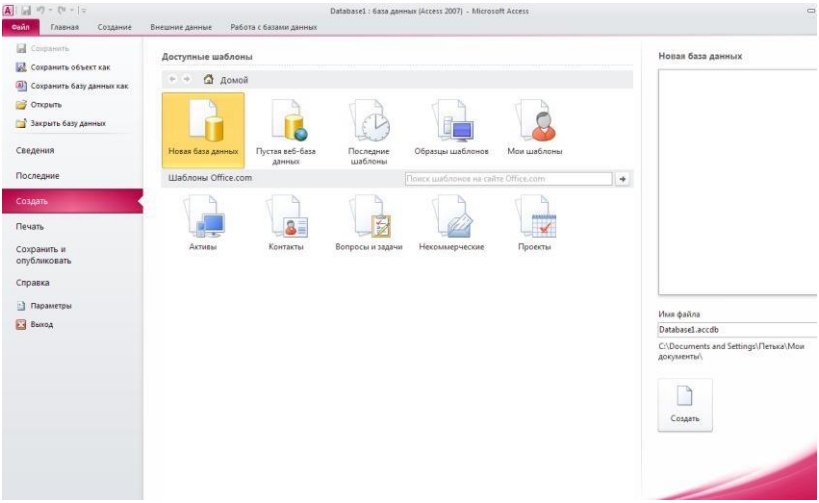


Рисунок 1 – Окно БД MS Access

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

Ниже представлены характеристики БД в СУБД MS Access:

– размер файла базы данных MS Access (.mdb или accdb для версии 2006 и выше) – 2 Гбайт за вычетом места, необходимого системным объектам;

– число объектов в базе данных – 768;

– модули (включая формы и отчеты, свойство Наличие модуля (HasModule) которых 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 ито говые 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 (Таблица 1):

Таблица 1 – Варианты предметных областей

№	Название предметной области АИС
1.	Автосалон
2.	Агентство недвижимости
3.	Аэропорт
4.	Банк
5.	Библиотека
6.	Гостиница
7.	Деканат
8.	Документооборот предприятия
9.	Магазин продовольственных товаров
10.	Музей
11.	Научная организация
12.	Отдел кадров
13.	Поликлиника
14.	Развлекательный центр
15.	Ресторан
16.	Сервисный центр
17.	Спортивный клуб
18.	Супермаркет
19.	Турфирма
20.	Университет

Проанализируйте объекты выбранной предметной области и создайте не менее 8-ми взаимосвязанных таблиц в соответствии с порядком выполнения работы, учитывая, что все таблицы должны быть нормализованы по 3НФ (см. учебное пособие).

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 (Рисунок 2).

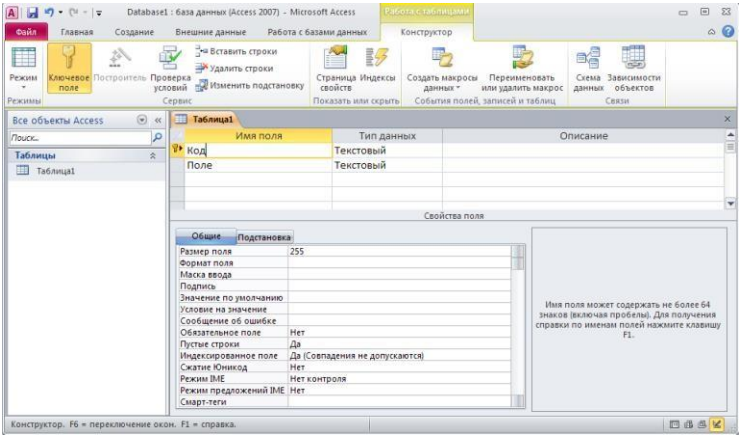


Рисунок 2 – Новая таблица в режиме конструктора

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 (MEMO) 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 форматов для различных типов данных представлено в таблице 2.

Таблица 2 – Описание форматов данных

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, 19 января 2022 г.
средний формат	19 – янв – 22
краткий формат	19.01.22
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.

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» (Рисунок 3).

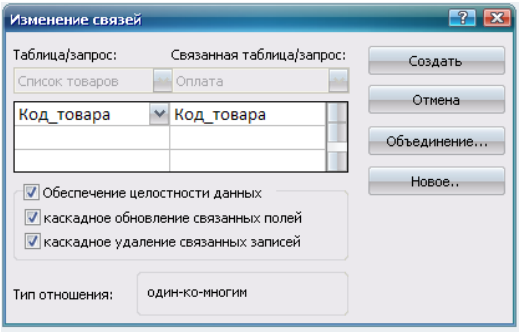


Рисунок 3 – Окно изменения связей между таблицами

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 (Рисунок 4).

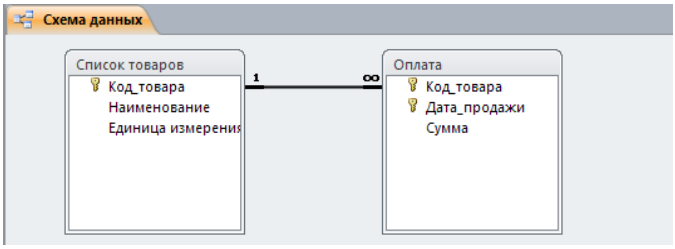


Рисунок 4 – Определение связей между таблицами

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 (промежуточной) таблицы, 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.

После создания схемы базы данных необходимо заполнить созданные таблицы, для чего выберите нужную таблицу и нажмите кнопку «Открыть» в окне «Базы данных».

2.2 Лабораторная работа «Создание элементов пользовательского интерфейса в среде MS Access»

Цель работы

Научиться создавать ленточную, табличную и сложную формы, а также отчеты в среде MS Access, используя в качестве источника записей созданные ранее таблицы.

Форма проведения

Выполнение индивидуального задания.

Форма отчетности

На проверку должны быть представлены созданные экранные и отчетные формы.

Теоретические основы

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 MS Access источником записей может быть таблица, запрос или инструкция SQL. В проекте MS 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 управления

– 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 различными способами:

1. 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.
2. 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.
3. 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».

Отчеты MS Access

Отчет – это гибкое и эффективное средство для организации данных при выводе на печать. С помощью отчета имеется возможность вывести данные на печать в том виде, в котором они требуется конкретному пользователю.

Основные данные для формирования отчета берется из базовой таблицы, запроса или инструкции SQL, являющихся источниками данных для отчета. Другие сведения (заголовки, примечания отчетов, количество страниц и другая

сопроводительная информация) вводятся при разработке отчета.

Пользователь имеет возможность разработать отчет самостоятельно или создать отчет с помощью мастера. Мастер по разработке отчетов MS Access выполняет всю рутинную работу и позволяет быстро разработать макет отчета. Работа мастера отчетов аналогична Работе других мастеров в среде MS Access. После создания основной части отчета разработчик может переключиться в режим конструктора и внести изменения в стандартный макет.

Данные в отчете могут быть сгруппированы по различным полям, например, сведения о продаже товара могут быть сгруппированы по месяцам, либо по наименованию товара. При этом в отчете можно указать способ сортировки выводимых на печать данных. Кроме этого, в отчете, по тому же принципу, как и в форме, можно создать подчиненный отчет.

При включении в главный отчет подчиненного отчета, содержащего данные, относящиеся к данным в главном отчете, необходимо установить связь подчиненного отчета с главным. Связь обеспечивает соответствие записей, выводящихся в подчиненном отчете, записям в главном отчете.

При создании подчиненного отчета с помощью мастера или путем переноса с помощью мыши отчета или таблицы в другой отчет, MS Access автоматически выполняет синхронизацию главного и подчиненного отчета в следующих случаях:

Отчеты базируются на таблицах, между которыми установлены связи в окне Схема данных. При создании отчетов на основе запроса или запросов синхронизация отчета с подчиненным отчетом выполняется автоматически, если связи установлены для базовых таблиц запроса или запросов. Если связи базовых таблиц установлены корректно, MS Access выполнит синхронизацию главного и подчиненного отчетов автоматически.

Главный отчет базируется на таблице, содержащей ключевое поле, а подчиненный отчет базируется на таблице, содержащей поле с тем же именем и с тем же или совместимым типом данных. Это же условие должно выполняться для базовых таблиц запросов, если отчеты базируются на запросах.

Перед созданием отчета убедитесь, что на компьютере настроен используемый по умолчанию принтер.

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 (Рисунок 5) 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».

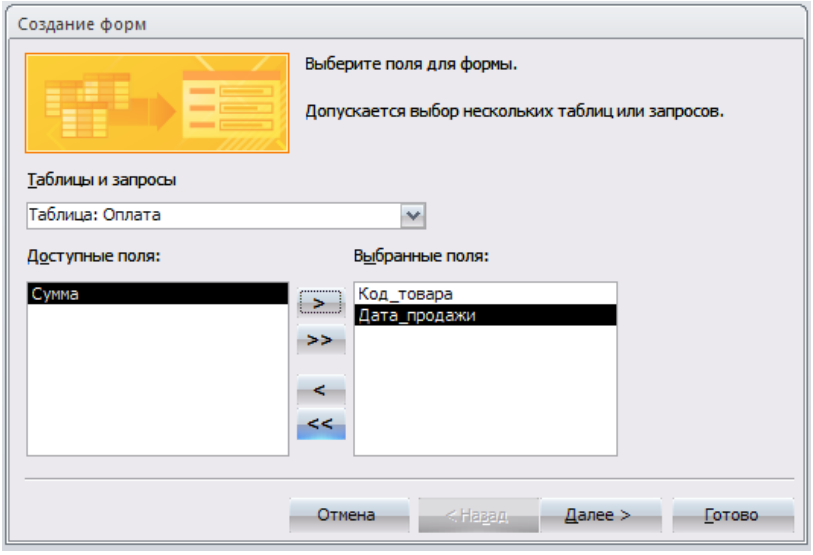


Рисунок 5 – Первое окно мастера форм

В следующем окне (Рисунок 6) выберите тип формы Ленточный, нажмите кнопку «Далее» и следуйте дальнейшим указаниям мастера форм.

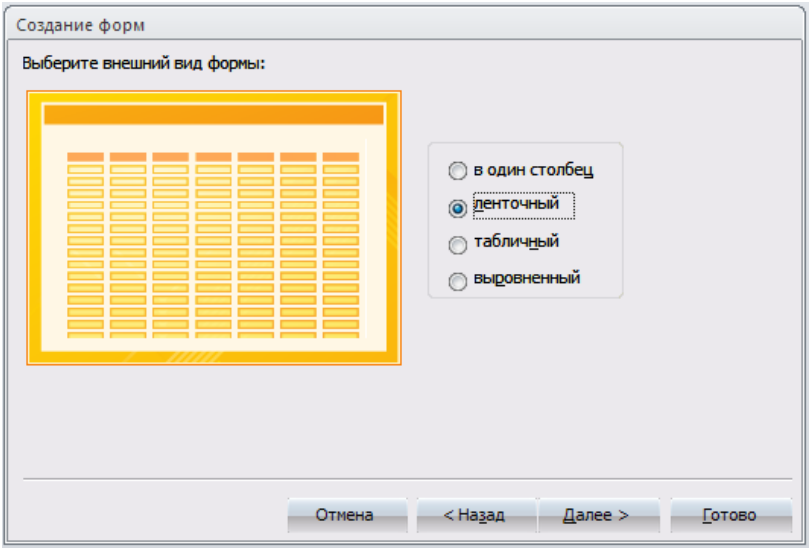


Рисунок 6 – Второе окно мастера форм

Для создания табличной формы с помощью мастера, следуйте предложенным выше инструкциям, а в окне (Рисунок 6) 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 (Рисунок 6) 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» (Рисунок 7) и поместите её в область примечания формы.

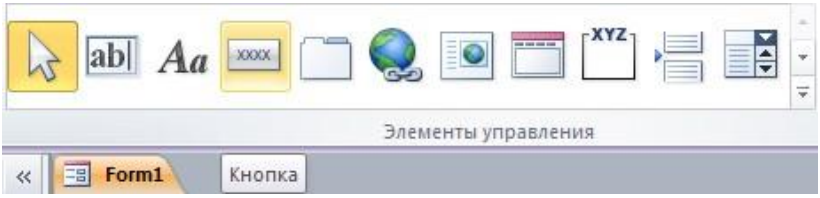


Рисунок 7 – Панель элементов

После этого откроется окно мастера кнопок (Рисунок 8) 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.

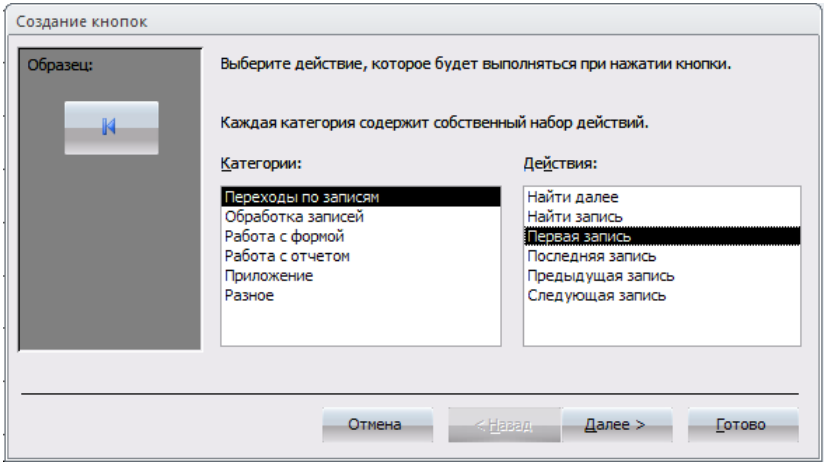


Рисунок 8 – Мастер кнопок

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 (Рисунок 9).

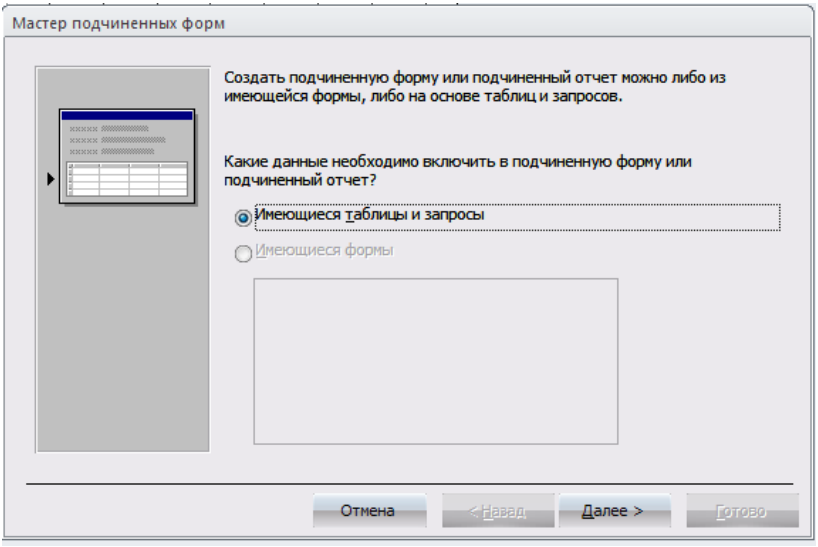


Рисунок 9 – Мастер подчиненных форм

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, также, 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»*.

Далее в ходе выполнения работы необходимо создать **отчет**, содержащий подчиненный отчет.

Для создания отчета в окне базы данных выберите вкладку «Отчеты» и нажмите кнопку «Создание отчета с помощью мастера».

В первом окне мастера отчетов выберите имя таблицы или запроса,

содержащих данные, по которым строится отчет. MS Access по умолчанию использует эту таблицу или запрос как базовый источник данных для отчета. Для удобства используйте тот же источник записей, что и ранее для главной формы.

В следующем окне (Рисунок 10) установите уровни группировки данных (если необходимо). Нажмите кнопку «Далее».

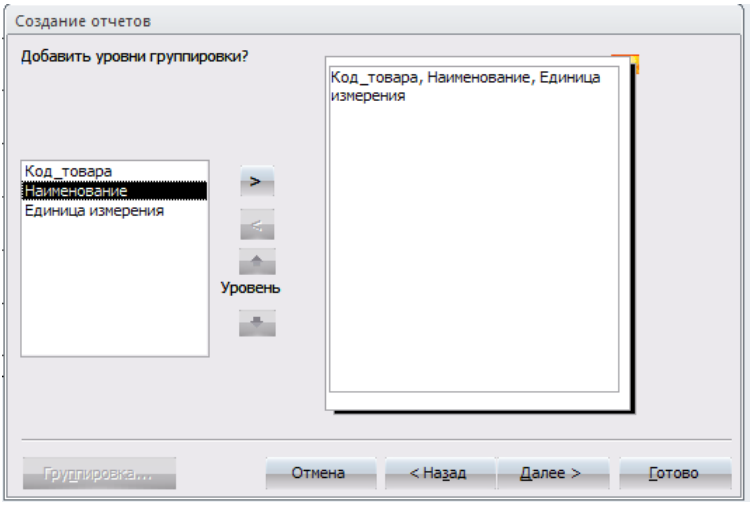


Рисунок 10 – Мастер отчетов, окно группировки

В следующем окне выберите способ сортировки данных в отчете, нажмите кнопку «Далее» и следуйте дальнейшим инструкциям мастера отчетов.

Для изменения макета отчета откройте его в режиме конструктора (Рисунок 11).

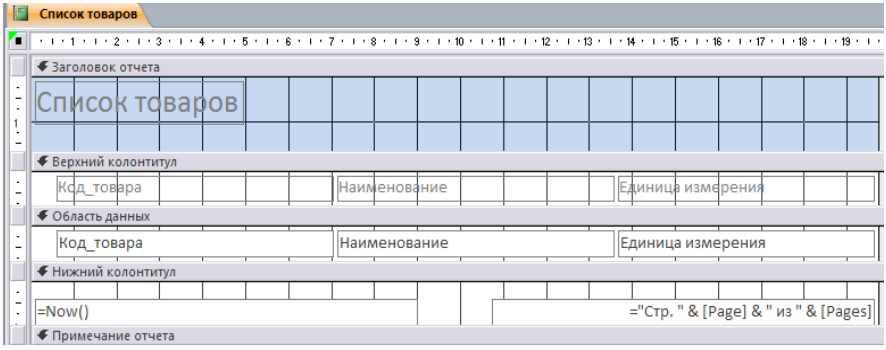


Рисунок 11 – Отчет, открытый в режиме конструктора

Чтобы изменить внешний вид отчета, нажмите кнопку «Автоформат» на панели инструментов и выберите новый внешний вид для отчета.

Для изменения внешнего вида одного элемента управления, например, надписи, выделите его, после чего на панели инструментов «Форматирование»

выберите другой шрифт, размер шрифта или другие параметры.

Для изменения формата отображения данных в элементе управления, например, в поле, убедитесь, что данный элемент выделен и нажмите кнопку «Свойства» на панели инструментов для вывода окна свойств.

При необходимости добавьте в отчет другие поля из таблицы, являющейся источником данных отчета следующим образом: нажмите кнопку «Список полей» на панели инструментов для отображения списка всех полей базовой таблицы и с помощью мыши «перенесите» выбранное поле в отчет.

Создайте заголовок и примечание отчета, куда добавьте надпись, содержащую Ваши ФИО и № группы. Чтобы создать надпись, нажмите кнопку «Надпись» Aa на панели элементов. Затем выберите в отчете место, куда ее следует поместить, введите нужный текст и нажмите клавишу «Enter».

Поменяйте порядок сортировки и группировки данных в отчете, для этого нажмите кнопку «Сортировка и группировка» на панели инструментов для вывода диалогового окна (Рисунок 12).

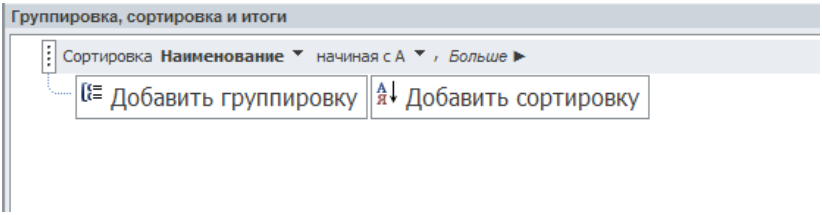


Рисунок 12 – Окно сортировки и группировки данных

Добавьте новые поля для группировки и сортировки, создайте за головки и примечание групп.

Для создания подчиненного отчета нажмите кнопку «Подчиненная форма/отчет» на панели элементов, затем установите указатель в отчете на том месте, куда требуется поместить подчиненный отчет, и нажмите кнопку мыши. Выполняйте инструкции, выводящиеся в диалоговых окнах мастера.

После нажатия кнопки «Готово» элемент управления «Подчиненная форма/отчет» будет вставлен в главный отчет. Кроме того, будет создан отдельный отчет, выводящийся как подчиненный отчет.

При включении в главный отчет подчиненного отчета, содержащего данные, относящиеся к данным в главном отчете, необходимо установить связь подчиненного отчета с главным. Связь обеспечивает соответствие записей, выводящихся в подчиненном отчете, записям в главном отчете.

При создании подчиненного отчета с помощью мастера или путем переноса с помощью мыши отчета или таблицы в другой отчет, MS Access автоматически выполняет синхронизацию главного и подчиненного отчета.

Также, как и при создании подчиненных форм, следует учитывать, что отчеты базируются на таблицах, между которыми установлены связи в окне Схема данных. Для подчиненного отчета используйте источник записей такой

же, как и для подчиненной формы. Помните, что при создании отчетов на основе запроса или запросов синхронизация отчета с подчиненным отчетом выполняется автоматически, если связи установлены для базовых таблиц запроса или запросов. Если связи базовых таблиц установлены корректно, MS Access выполнит синхронизацию главного и подчиненного отчетов автоматически.

Главный отчет основан на таблице, содержащей ключевое поле, а подчиненный отчет базируется на таблице, содержащей поле с тем же именем и с тем же или совместимым типом данных, являющимся внешним ключом к первой таблице. Это же условие должно выполняться для базовых таблиц запросов, если отчеты базируются на запросах.

При связывании главного и подчиненного отчетов MS Access использует свойства «Основные поля» (LinkMasterFields) и «Подчиненные поля» (LinkChildFields) элемента управления «Подчиненная форма/отчет». Если по каким-либо причинам MS Access не связывает главный и подчиненный отчет автоматически, пользователь имеет возможность задать значения этих свойств самостоятельно.

Для открытия отчета в сложной форме создайте кнопку открытия отчета для просмотра, используя для этого мастер кнопок.

2.3 Лабораторная работа «Создание SQL-запросов среде MS Access»

Цель работы

Научиться создавать SQL-запросы различного назначения.

Форма проведения

Выполнение индивидуального задания.

Форма отчетности

На проверку должны быть представлены созданные SQL-запросы.

Теоретические основы

Типы запросов, создаваемых в MS 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 четыре запроса, создаваемые в рамках данной лабораторной работы, должны быть реализованы с помощью построителя запросов в режиме конструктора (Рисунок 13).

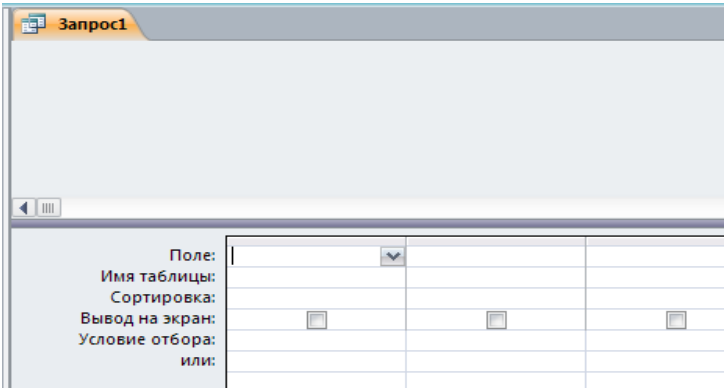


Рисунок – 13 Бланк построителя запросов

Для создания нового запроса в окне базы данных (Рисунок 1) перейдите на вкладку «Запросы» и нажмите кнопку «Создание запроса в режиме конструктора». В появившемся окне (Рисунок 14) выберите таблицу (таблицы) – источник запроса. Если запрос уже открыт, то для перехода в режим конструктора следует нажать кнопку «Вид»  на панели инструментов.

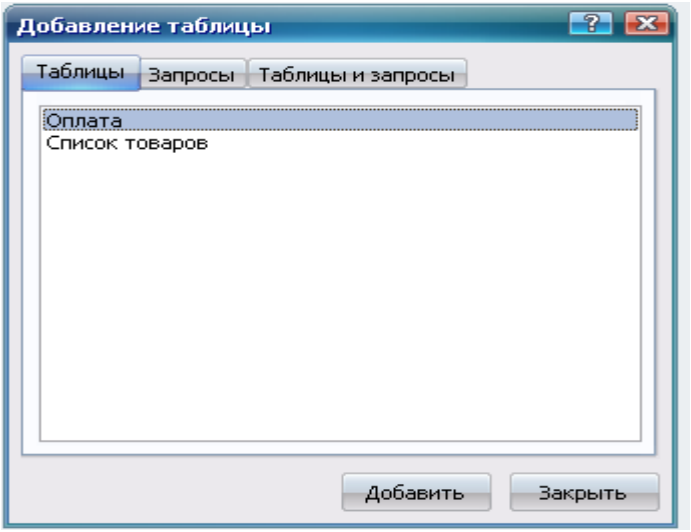


Рисунок – 14 Добавление таблицы

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» (Рисунок 15) 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.

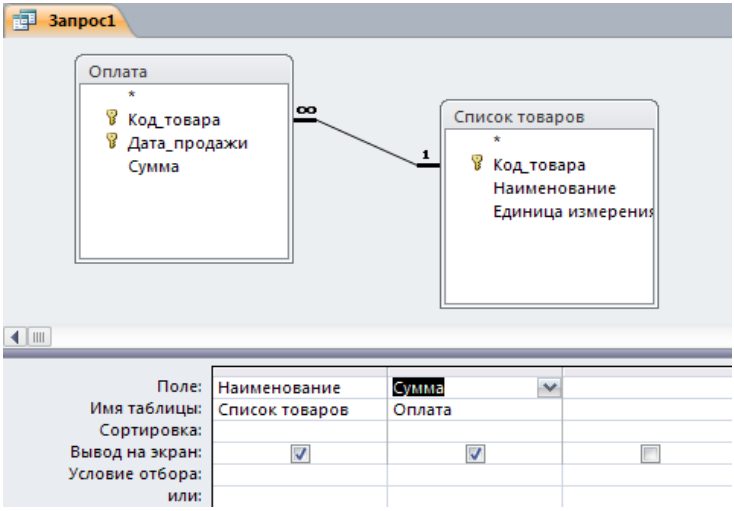


Рисунок – 15 Запрос на выборку

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 (Рисунок 16), введенное в окне приглашения значение будет являться значением параметра.

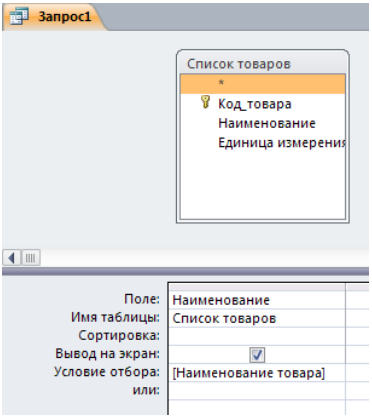


Рисунок – 16 Запрос на выборку с параметрами

При запуске запроса (Рисунок 16) 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 (Рисунок 17).

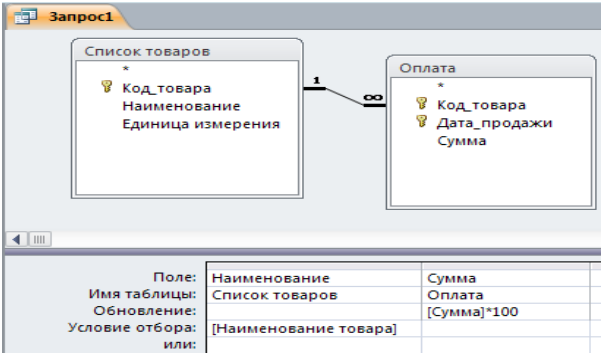


Рисунок – 17 Запрос на обновление

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 18 схематично представлен SQL-запрос на выборку данных из таблицы «Студент».

Рисунок – 18. 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 (Рисунок 19).

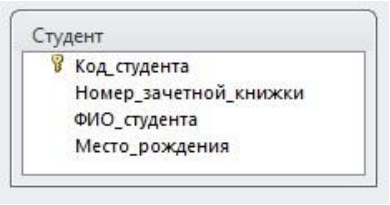


Рисунок – 19 Таблица «Студент» в схеме данных

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 20. 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.

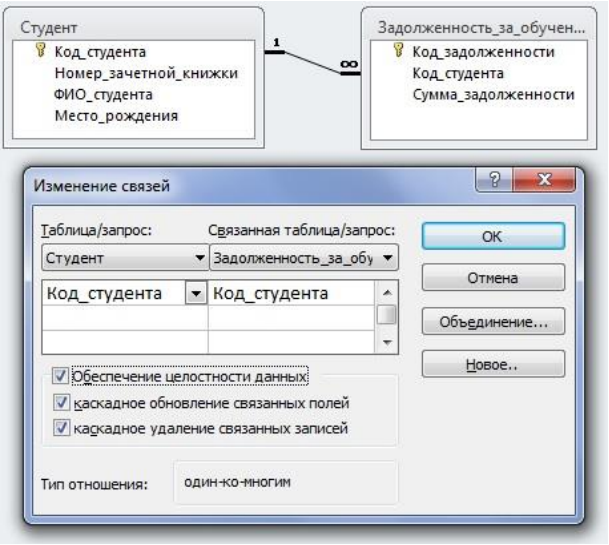


Рисунок – 20 Схема данных после создания таблицы «Задолженность_за_обучение»

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 (Рисунок 21). 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.

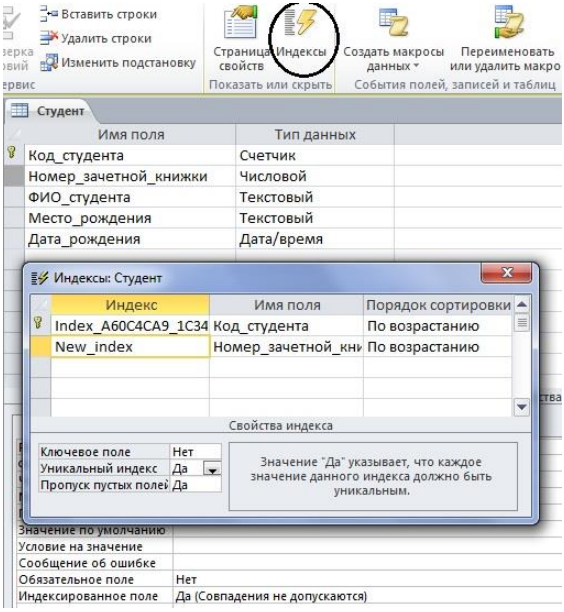


Рисунок – 21 Результат создания уникального индекса

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

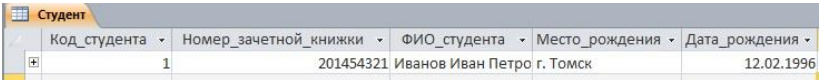
указать значения полей одной новой записи с помощью предложения VALUES. Если список полей опущен, предложение VALUES должно содержать значение для каждого поля таблицы; в противном случае инструкция INSERT не будет выполнена. Можно использовать дополнительную инструкцию INSERT INTO с предложением VALUES 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 22 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

Рисунок – 22 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» (Рисунок 23), а также заполним эти таблицы.

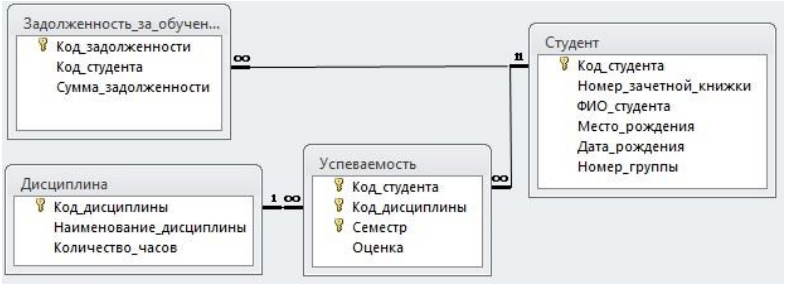


Рисунок – 23 Обновленная схема базы данных

На рисунке 24 приведено содержимое таблиц «Студент», «Успеваемость» и «Дисциплина»

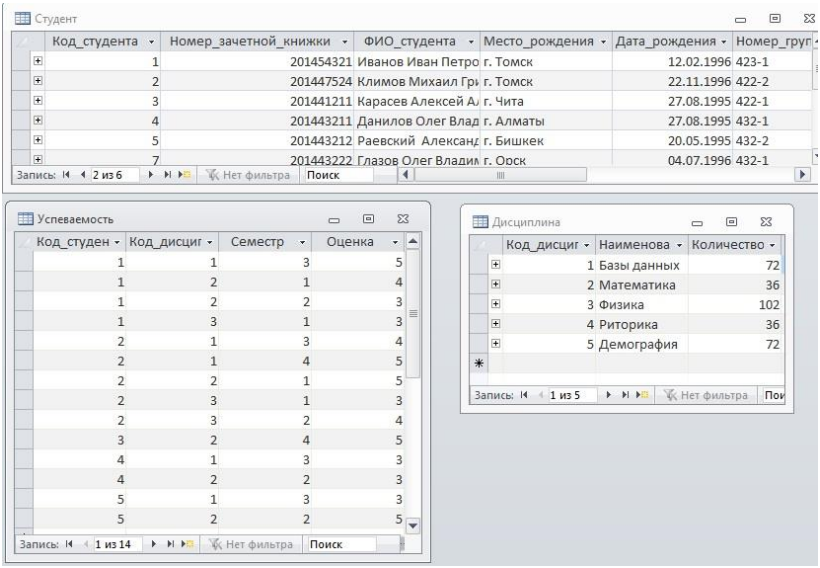
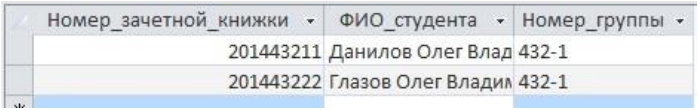


Рисунок – 23 Содержимое таблиц базы данных

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 24 представлен результат выполнения этого запроса, были найдены 2 записи, удовлетворяющие критерию запроса.



Номер_зачетной_книжки	ФИО_студента	Номер_группы
201443211	Данилов Олег Влад	432-1
201443222	Глазов Олег Владим	432-1

Рисунок – 24 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 25 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

Рисунок – 25 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

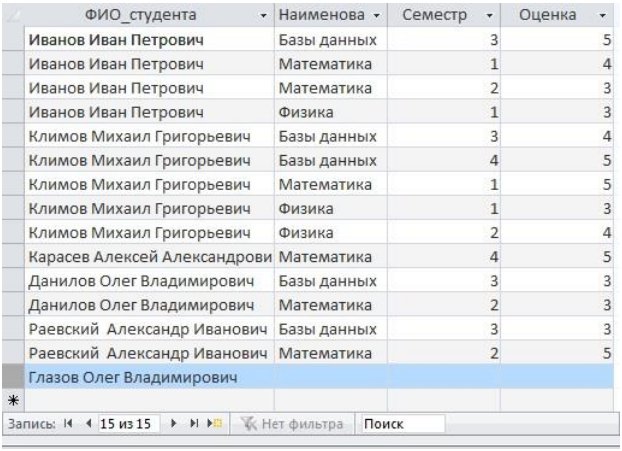
FROM Студент LEFT JOIN
(Дисциплина RIGHT JOIN Успеваемость ON

Дисциплина.Код_дисциплины = Успеваемость.Код_дисциплины)

ON Студент.Код_студента = Успеваемость.Код_студента;

На рисунке 26 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

Рисунок – 26 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 (Рисунок 27):

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

Рисунок – 27 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 запросы 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.

2.4 Лабораторная работа «Создание концептуальной модели данных»

Цель работы

Научиться проектировать концептуальную модель, выбранной ранее предметной области в среде автоматизированного проектирования.

Форма проведения

Выполнение индивидуального задания.

Форма отчетности

На проверку должна быть представлена созданная ER-диаграмма выбранной предметной области.

Теоретические основы

Сущности и атрибуты

Каждая сущность является множеством подобных индивидуальных объектов, называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от всех остальных экземпляров. Атрибут выражает определенное свойство объекта. С точки зрения БД (физическая модель) сущности соответствует таблица, экземпляру сущности – строка в таблице, а атрибуту – колонка таблицы.

Построение модели данных предполагает определение сущностей и атрибутов, т.е. необходимо определить, какая информация будет храниться в конкретной сущности или атрибуте. Сущность можно определить как объект, событие или концепцию, информация о которой должна сохраняться. Сущности должны иметь наименование с четким смысловым значением, именоваться существительным в единственном лице, не носить «технических» наименований и быть достаточно важными для того, чтобы их моделировать. Именование сущности в единственном числе облегчает в дальнейшем чтение модели. Фактически имя сущности дается по имени ее экземпляра.

Каждая сущность должна быть полностью определена с помощью текстового описания. Каждый атрибут хранит информацию об определенном свойстве сущности, а каждый экземпляр сущности должен быть уникальным. Атрибут или группа атрибутов, которые идентифицируют сущность, называется первичным ключом. При установлении связей между сущностями атрибуты первичного ключа родительской сущности мигрируют в качестве внешних ключей в дочернюю сущность.

Очень важно дать атрибуту правильное имя. Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение. Соблюдение этого правила позволяет частично решить проблему нормализации данных уже на этапе определения атрибутов.

Связи

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой. Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение построенной модели данных.

Различают зависимые и независимые сущности. Тип сущности определяется ее связью с другими сущностями. Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности атрибуты помечаются как внешний ключ (FK).

При установлении неидентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь служит для связывания независимых сущностей.

Имя связи – фраза, характеризующая отношение между родительской и дочерней сущностями. Для связи один-ко-многим идентифицирующей или не идентифицирующей достаточно указать имя, характеризующее отношение от родительской к дочерней сущности.

Тип связи (идентифицирующая/неидентифицирующая). Для неидентифицирующей связи можно указать обязательность. В случае обязательной связи атрибут внешнего ключа получит признак NOT NULL, несмотря на то, что внешний ключ не войдет в состав первичного ключа дочерней сущности. В случае необязательной связи внешний ключ может принимать значение NULL. Необязательная неидентифицирующая связь помечается прозрачным ромбиком со стороны родительской сущности.

Правила ссылочной целостности – логические конструкции, которые выражают бизнес-правила использования данных и представляют собой правила вставки, замены и удаления.

Информацию о предметной области суммируют составлением спецификаций по сущностям, атрибутам и отношениям с использованием графических диаграмм, в чем и заключается процесс моделирование данных.

Порядок выполнения работы

Для запуска пакета MS Visio в меню Windows найдите соответствующий ярлык (иконку). Для создания концептуальной модели данных необходимо выбрать меню Файл/ Создать. Далее появится панель с категориями шаблонов (Рисунок 24), в котором надо выбрать Программы и базы данных.

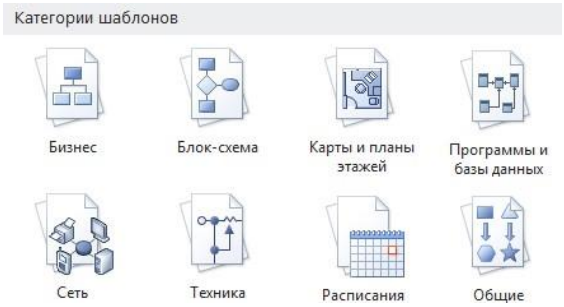


Рисунок 24 – Панель выбора категории шаблонов

Далее появится панель с группой шаблонов из выбранной категории (Рисунок 25), в котором необходимо выбрать шаблон «Схема модели базы данных».

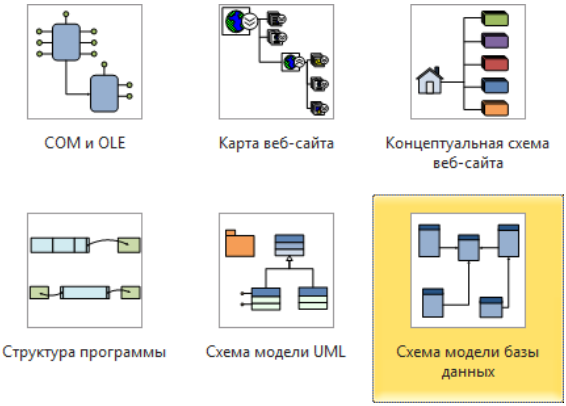


Рисунок 25 – Панель выбора шаблона

После двойного нажатия на выбранный шаблон или иконки «Создать» (Рисунок 26) появится окно, в котором создается ER-диаграмма.

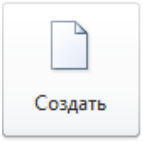


Рисунок 26 – Иконка создания шаблона

Создание сущностей

Прежде чем приступить к созданию концептуальной модели, зайдите на вкладку «База данных», выберите пункт на панели инструментов «Показать параметры» и выставьте следующие настройки (рисунки 27 – 29).

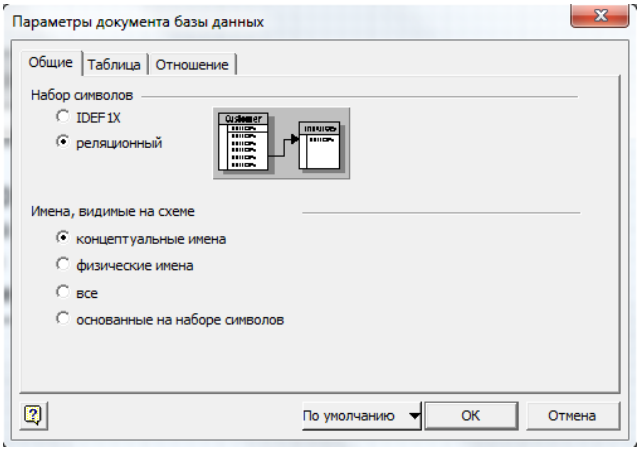


Рисунок 27 – Параметры документа (вкладка «Общие»)

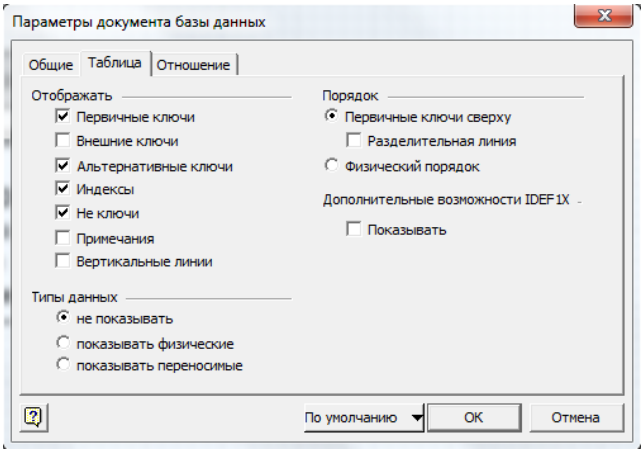


Рисунок 28 – Параметры документа (вкладка «Таблица»)

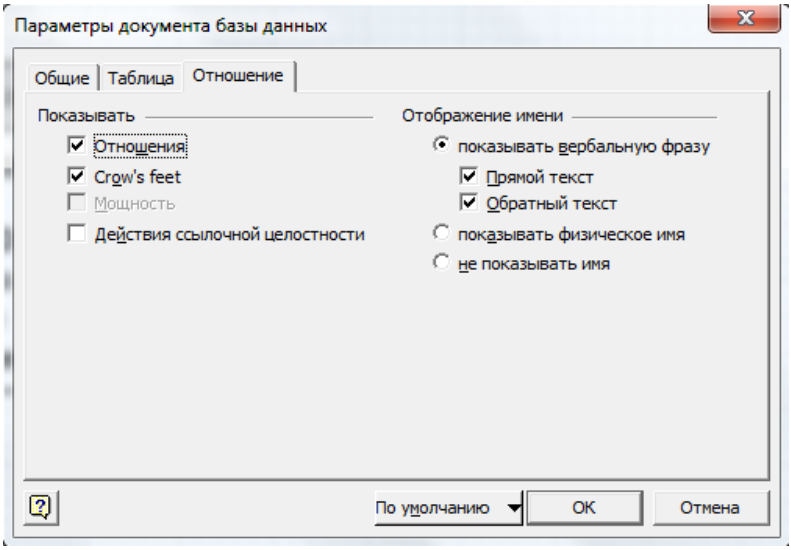


Рисунок 29 – Параметры документа (вкладка «Отношение»)

Далее на панели «Фигуры» (Рисунок 30) в группе фигур выберите вкладку «Отношение сущности».

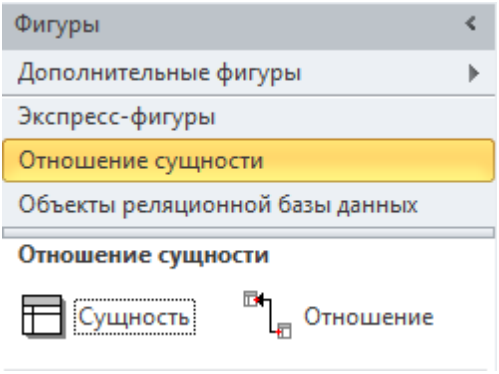


Рисунок 30 – Панель Фигуры

Для создания сущности, выберите и «перетащите» элемент «Сущность» с панели элементов (вкладка «Отношение сущности») на рабочее поле в нужном месте (Рисунок 31).

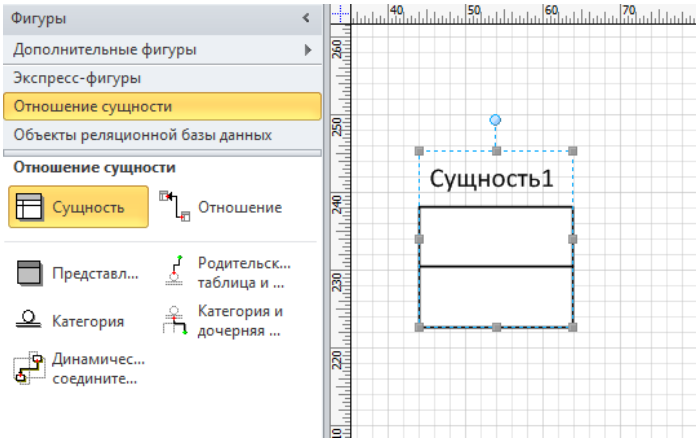


Рисунок 31 – Создание сущности

Для изменения свойства созданной сущности нажмите левую кнопку мышки, находясь на ней, и на панели «Свойства базы данных» в группе «Категории» (Рисунок 32) отобразится список из вкладок со свойствами сущности.

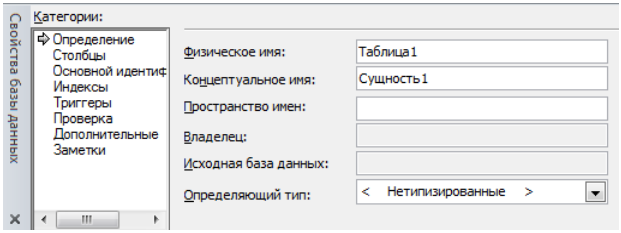


Рисунок 32 – Панель свойств сущности

В отобразившейся панели «Свойства базы данных» в разделе «Категории» восемь вкладок.

Вкладка «Определение» позволяет ввести следующие параметры:

- Физическое имя – имя таблицы в точном соответствии с его отображением в базе данных;
- Концептуальное имя – имя таблицы в точном соответствии с его отображением в концептуальной модели;
- Пространство имен – значение, используемое дополнительно для различения элементов моделей с идентичными именами, например, элементов баз данных Oracle;
- Владелец – отображает сведения о владельце выбранной таблицы, полученные из центральной СУБД при реконструировании. Сведения доступны только для чтения;
- Исходная база данных – отображает базу данных, из которой была извлечена выбранная таблица. Эти сведения получены из центральной СУБД при

реконструировании и доступны только для чтения;

- Определяющий тип – содержит список всех составных типов данных, позволяющий создать типизованную таблицу для объектно-реляционной базы данных. Поле «Определяющий тип» доступно, только в том случае, если таблица пуста;
- Синхронизация имен при вводе – если установлен этот флажок, то при изменении одного из имен – концептуального или физического – соответствующим образом будет изменено и другое имя.

Вкладка «Столбцы» содержит таблицу (Рисунок 33)

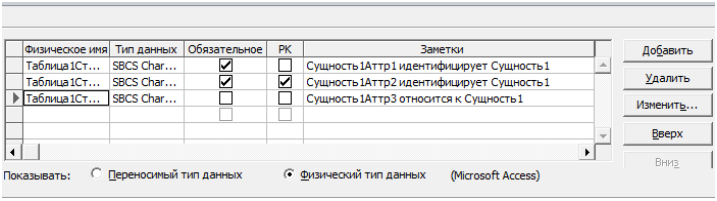


Рисунок 33 – Таблица ввода атрибутов сущности

На данной вкладке необходимо определить атрибуты (поля будущей таблицы) сущности:

- Физическое имя – имя атрибута, которое будет использоваться при генерации физической модели;
- Тип данных – тип данных атрибута, который может быть выбран из выпадающего списка, щелкнув в поле «Тип данных»;
- Обязательное – установите этот флажок, чтобы указать, что атрибут не может содержать пустые значения;
- PK – установите этот флажок, чтобы указать, что атрибут является первичным ключом для таблицы (в физической модели данных атрибут будет являться первичным ключом или его составной частью);
- Добавление – выберите (щелкните клавишей мыши), чтобы добавить атрибут;
- Удаление – щелкните, чтобы удалить выбранный атрибут.
- Изменение – щелкните, чтобы открыть диалоговое окно «Свойств столбца (атрибута)», в котором можно задать дополнительные параметры для выбранного атрибута.
- Вверх – щелкните, чтобы изменить порядок атрибутов в этом списке и на диаграмме;
- Вниз – щелкните, чтобы изменить порядок атрибутов в этом списке и на диаграмме;
- Переносимый тип данных – щелкните, чтобы показать типы данных, которые не зависят от драйвера;
- Физический тип данных – щелкните, чтобы показать типы данных, используемые в выбранном драйвере.

Более полную информацию по свойствам атрибута можно получить, нажав кнопку «Изменить», описанную выше. Здесь можно задавать

концептуальное имя, значения атрибута по умолчанию, включать синхронизацию имен при вводе во вкладе «Определение», просмотреть драйвер по умолчанию, задать длину значения атрибута, задать тип, размер, категорию атрибута во вкладе «Тип данных». Также можно выбрать тип-семейства во вкладке «Семейство», определять верхние и нижние границы диапазона значений, а также собственный список значений во вкладке «Проверка», просмотреть дополнительную информацию во вкладе «Дополнительные», вводить заметки во вкладке «Заметки».

Вкладка «Основной идентификатор» (Рисунок 34) – содержит автоматически заполняемый список первичных идентификаторов сущности. С помощью параметров данной вкладки можно изменить, определить или удалить первичные ключи из списка имеющихся атрибутов, а также указать, следует ли создать индекс по первичным ключам. При первом просмотре данной категории отображаются все сведения о таблице, имеющие отношение к первичному ключу. Впоследствии в этой вкладке можно создать первичный ключ либо изменить его состав или физическое имя, а также создать индекс для первичного ключа.

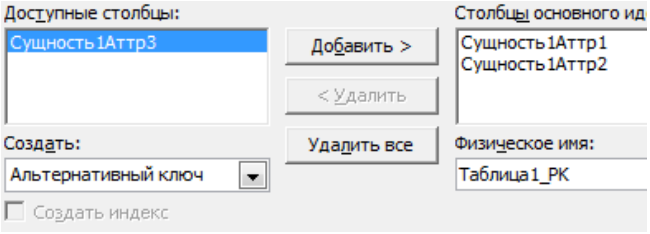


Рисунок 34 – Вкладка Основной идентификатор

С помощью параметров вкладки «Индексы» можно определить индекс и изменить его свойства для выбранной таблицы. Например, можно изменить имя индекса, его параметры, порядок столбцов в индексе и указать расширенные атрибуты индекса. Расширенные атрибуты индекса зависят от драйвера и поддерживаются не всеми СУБД. В Visio уникальные индексы автоматически создаются на основе первичных ключей.

С помощью параметров вкладок «Триггеры» и «Проверка» можно добавлять, изменять, удалять и просматривать код триггеров и проверки соответственно. Параметры в этих вкладках зависят от конкретной СУБД.

Вкладка «Заметки» – носит описательный характер для улучшения понимания модели.

Домен

Домен – это множество допустимых значений атрибута определенного типа данных. Домен определяется заданием стандартного типа данных, к которому относятся элементы домена и заданием произвольного логического выражения, применяемому к этому типу данных.

Для создания домена необходимо на панели «Фигуры» (Рисунок 30) из списка фигур выбрать вкладку «Объекты реляционной базы данных». Далее

выберите и перетащите элемент «Тип» с панели элементов (вкладка «Объекты реляционной базы данных») на рабочее поле в нужном месте.

Чтобы определить значение «Тип» как «Домен», щелкните по нему и в списке «Категории» на панели «Свойства базы данных» выберите вариант «Определение». В разделе «Составной тип» выберите параметр «Домен», чтобы указать, что этот тип является псевдонимом другого типа (неявное представление того же типа). В поле с меткой «Имя» указывается имя домена. Выберите параметр для «Тип-семейство» псевдонима, чтобы указать, является ли значение атрибута набором из единственного значения, множества, списка или нескольких множеств (Рисунок 35). В реляционных базах данных все типы набора атрибутов являются единственными значениями. Объектно-реляционные базы данных позволяют указывать дополнительные типы наборов.

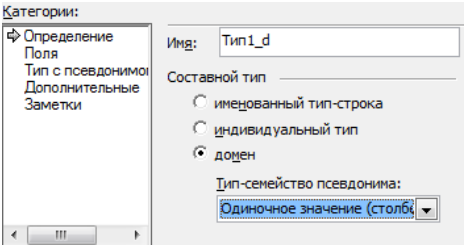


Рисунок 35 – Настройка домена

После создания нового домена его имя появится в списке «Тип данных» при задании свойств атрибутов сущностей (Рисунок 36).

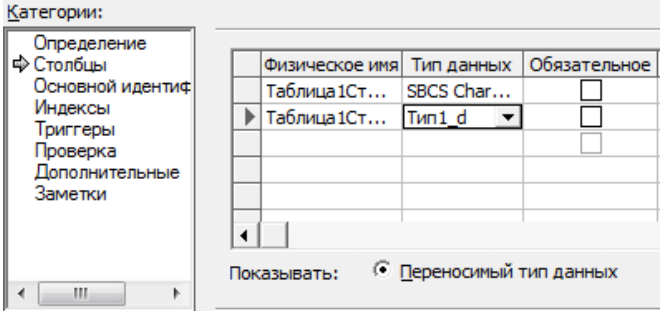


Рисунок 36 – Определение домена для атрибутов сущности

Следует отметить, что при создании атрибутов сущностей в концептуальной модели нет необходимости создавать атрибуты, являющиеся внешними ключами сущностей.

После того, как созданы все необходимые сущности и атрибуты, необходимо определить связи между ними.

Установка связей

Для установки связи между двумя сущностями, необходимо, нажать на кнопку «Отношение» в разделе «Фигуры» (Рисунок 37).

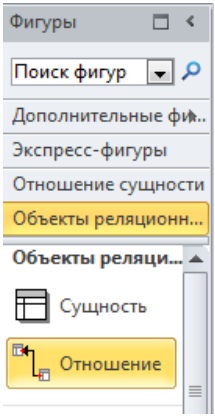


Рисунок 37 – Отношение (связь)

Соедините две сущности связью – от старшей к подчиненной. Связь установлена (Рисунок 38).

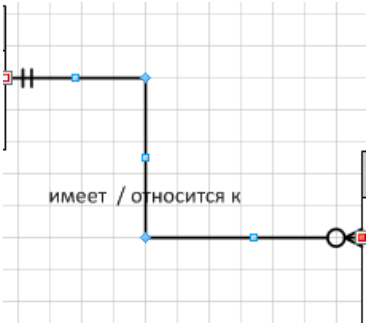


Рисунок 38 – Установленная связь

Для изменения свойств связи, необходимо дважды щелкнуть левой кнопкой мыши на линию связи (или нажать правую кнопку мыши и выпавшем меню выбрать пункт «Свойства базы данных») и на панели

«Свойства базы данных» в группе «Категории» отобразится список из четырех вкладок со свойствами связи.

Вкладка «Определение» – позволяет связывать/отсоединять атрибуты разных сущностей. Выделите атрибуты в разделе «Родительский элемент» и «Дочерний элемент» и нажмите кнопку «Связать» (Отсоединить). Имя роли внешнего ключа – носит описательный характер для улучшения понимания модели.

Вкладка «Имя» (текст роли считывается в направлении «предок-потомок») позволяет задать следующие параметры:

- Глагольные фразы – описывает родительскую роль (этот текст не может использоваться для именования каких-либо объектов базы данных);
- Обратная фраза – описывает дочернюю роль (этот текст не может

использоваться для именования каких-либо объектов базы данных);

- Физическое имя – имя связи в точном соответствии с его отображением в физической базе данных (это значение используется по-разному в зависимости от конечной СУБД)
- Заметки – носит описательный характер для улучшения понимания модели.

Вкладка «Прочее» позволяет задать мощность (один-к-одному, один-ко-многим и т.д.) и тип отношения (связи) – идентифицирующее/ неидентифицирующее. В Visio по умолчанию используется неидентифицирующее отношение. Кроме того, при не идентифицирующем отношении указывается, является ли наличие родительской сущности обязательным (т.е. может ли существовать экземпляр дочерней сущности, если не существует экземпляра родительской). Если наличие родительского объекта является необязательным, графически это отобразится в виде не закрашенного ромба со стороны родительской сущности.

Вкладка «Действия» ссылочной целостности позволяет задавать сценарии (без действий, каскад, и т.д.) при обновлении/ удалении родительского объекта.

Если размерность связи не устраивает, то дважды щелкните связь. На панели «Свойства базы данных» в группе «Категории2 выберите вариант «Прочее». В группе «Мощность» выберите размерность, наиболее подходящую к связи. Для отношения «один-ко-многим» наилучшим вариантом будет 0 или более, или 1 или более (Для отношения «один-к-одному» – 0 или 1 или ровно 1).

Обратите внимание, что в Visio для атрибутов, между которыми создаются отношения, должен совпадать тип данных и название (с учетом регистра). Иначе будет создано дополнительное поле с внешним ключом. В СУБД (например, Access) такое условие не является обязательным.

После того, как созданы все сущности, указаны атрибуты и установлены все связи, необходимо проверить правильность построения концептуальной модели. На этом этапе рекомендуется проанализировать сущность на соответствие 3 нормальной форме (НФ).

Работа считается полностью выполненной, если создано не менее 8-ми взаимосвязанных сущностей и при проверке модели нет ненормализованных по 3-НФ сущностей.

Допускается создание ER-модели в среде автоматизированного проектирования, отличной от MS Visio.

2.5 Лабораторная работа «Реконструкция схемы базы данных»

Цель работы

Научиться реконструировать существующую базу данных в модель данных в пакете VISIO.

Форма проведения

Выполнение индивидуального задания.

Форма отчетности

На проверку должна быть представлена созданная ER-диаграмма выбранной предметной области.

Теоретические основы

Если существует база данных, которую нужно представить в виде модели данных для лучшего понимания ее структуры или применения в качестве основы создания новой модели, то для извлечения схемы или структуры базы данных можно воспользоваться мастером реконструирования.

Элементы, извлекаемые мастером реконструирования

Элементы определения схемы, которые может извлечь мастер, зависят от сочетания различных факторов, например, от возможностей системы управления базами данных (СУБД) и драйвера ODBC. Во время работы мастера вы увидите, какие элементы доступны для извлечения, и сможете выбрать нужные. Например, можно выбрать только пять таблиц из десяти и два представления из четырех.

В мастере можно задать автоматическое создание документа в дополнение к списку реконструируемых элементов в окне Таблицы и представления. Если вы решили не создавать документ автоматически, то можете перетащить элементы из окна Таблицы и представления на страницу документа, чтобы построить модель базы данных вручную.

Извлечь можно такие элементы (при условии, что они доступны в целевой СУБД):

- таблицы;
- представления;
- первичные ключи;
- внешние ключи;
- индексы;
- триггеры (включая код);
- предложения проверки (включая код); хранимые процедуры (включая код).

Порядок выполнения работы

Для запуска пакета MS Visio в меню Windows найдите соответствующий ярлык (иконку). Для создания концептуальной модели данных необходимо выбрать меню «Файл»/ «Создать». Для реконструирования схемы базы данных необходимо выбрать «Файл»/ «Создать». Далее появится панель с категориями шаблонов (рисунок 39), в котором надо выбрать «Программы и базы данных».

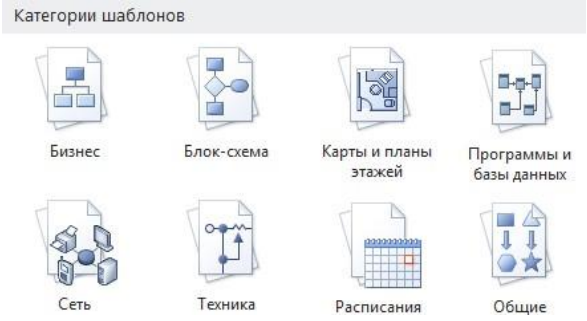


Рисунок 39 – Панель выбора категории шаблонов

Далее появится панель с группой шаблонов из выбранной категории (Рисунок 40), в котором надо выбрать Схема модели базы данных.

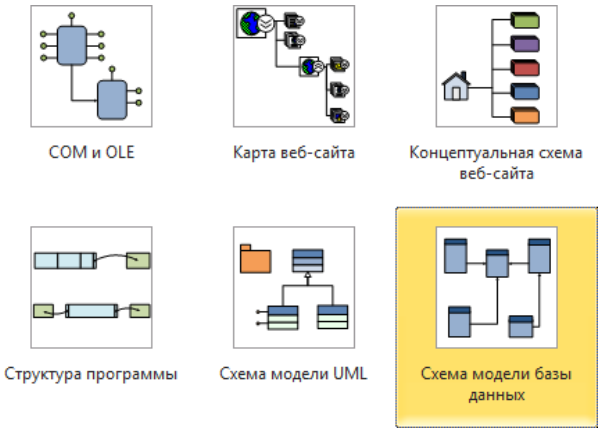


Рисунок 40 – Панель выбора шаблона

После двойного нажатия на выбранный шаблон или иконки «Создать» (Рисунок 41) появится окно, в котором создается ER-диаграмма.

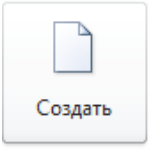


Рисунок 41 – Иконка создания шаблона

Прежде чем приступить к реконструкции, на вкладке «База данных» в группе «Управление» щелкните «Драйверы баз данных» и выставьте следующие настройки (Рисунки 42, 43).

В открывшемся окне в группе Драйвер по умолчанию для Visio выберите MS Access (Рисунок 42) и нажмите на кнопку «Настройка».

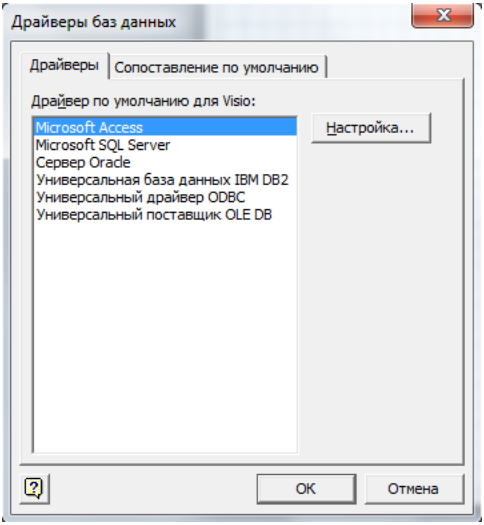


Рисунок 42 – Драйвер по умолчанию для Visio

В открывшемся окне (Рисунок 43) в закладке «Драйверы ODBC» в группе «Выберите нужные драйверы ODBC» отметьте MS Access Driver (*.mdb, *.accdb) и нажмите на кнопку «ОК» в окне «Установка MS Access» и затем в окне «Драйверы баз данных».

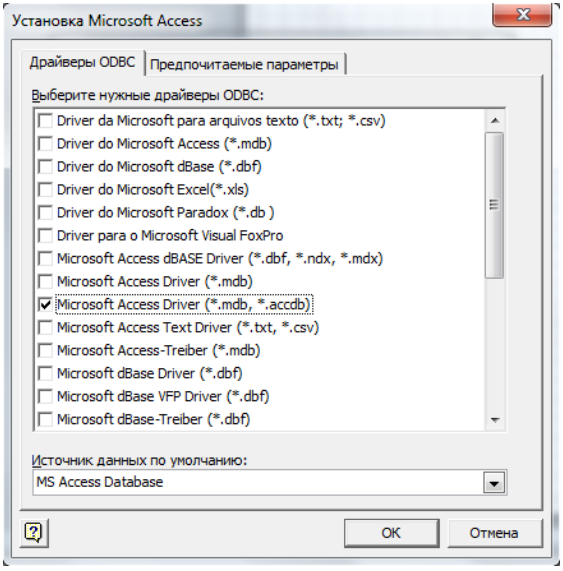


Рисунок 43 – Установка MS Access

Реконструкция существующей базы данных

На вкладке «База данных» в группе «Модель» нажмите кнопку

«Реконструирование». Откроется окно «Мастер реконструирования» (Рисунок 44).

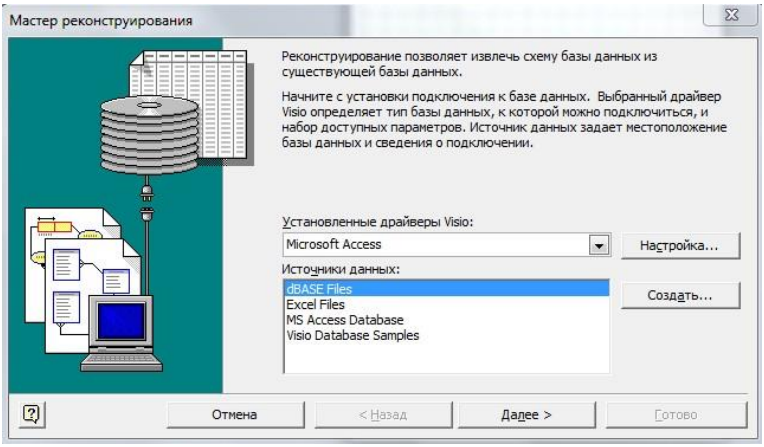


Рисунок 44 – Мастер реконструирования

На первом экране мастера реконструирования выберите драйвер MS Access в списке «Установленные драйверы Visio». Далее в качестве «Источника данных» укажите MS Access Database (Рисунок 45) и нажмите кнопку «Далее».

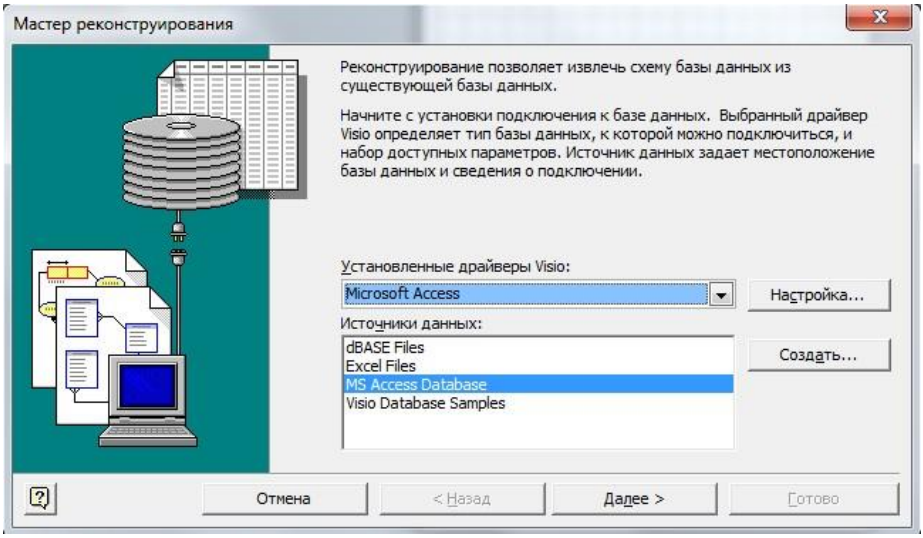


Рисунок 45 – Первый экран мастера реконструирования

Откроется окно Подключение источника данных (Рисунок 46). Укажите Логин пользователя и Пароль и нажмите «ОК». Если источник данных не защищен паролем, нажмите кнопку «ОК».

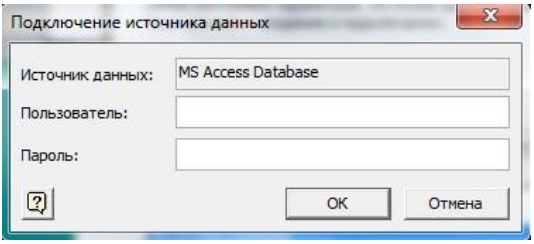


Рисунок 46 – Подключение источника данных

Откроется окно «Выбор баз данных» (Рисунок 47). С помощью проводника найдите файл базы данных, щелкните по нему и нажмите кнопку «ОК».

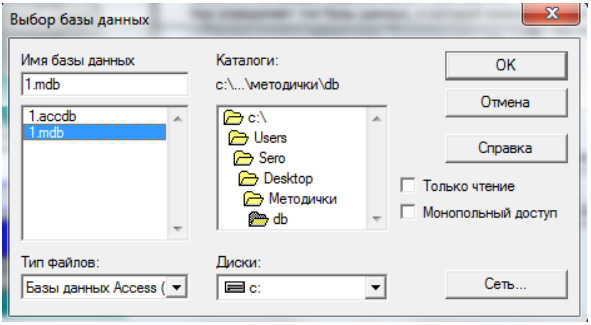


Рисунок 47 – Выбор базы данных

Откроется второе окно мастера реконструирования (Рисунок 48). Установите флажки для тех типов данных, которые нужно извлечь, и нажмите кнопку «Далее». Некоторые элементы могут быть затенены, т. е. недоступны, потому что не каждая система поддерживает все типы элементов, которые может извлечь мастер.

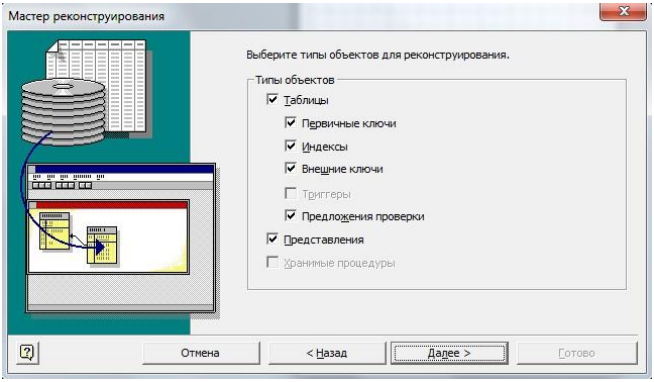


Рисунок 48 – Выбор объектов

Откроется третье окно мастера реконструирования (Рисунок 49). Установите флажки для таблиц (и представлений, если они есть), которые нужно извлечь, или нажмите «Выделить все», чтобы извлечь все элементы, и нажмите кнопку «Далее» (Если в третьем окне мастера реконструирования вы установили флажок «Хранимые процедуры», откроется окно с выбором процедур, которые нужно извлечь, или щелкните «Выделить все», чтобы извлечь их все, а затем нажмите кнопку «Далее»).

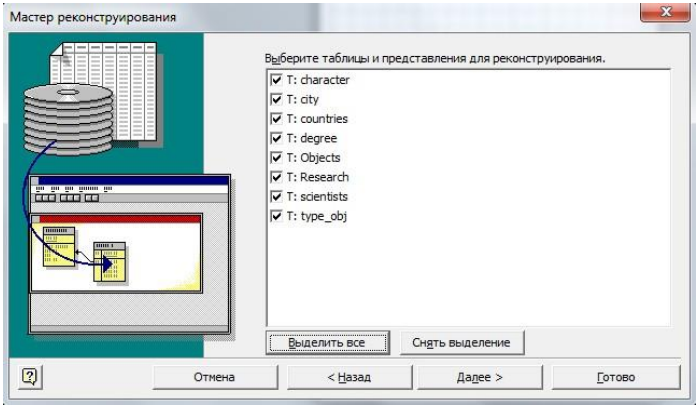


Рисунок 49 – Выбор таблиц и представлений

Откроется следующее окно мастера реконструирования (Рисунок 50). Укажите, нужно ли автоматически добавлять реконструируемые элементы на текущую страницу. Вы можете задать автоматическое создание документа в дополнение к списку реконструируемых элементов в окне «Таблицы и представления». Если вы решили не создавать документ автоматически, то можете перетащить элементы из окна «Таблицы» и представления на страницу документа, чтобы построить модель базы данных вручную. Нажмите кнопку

«Далее».

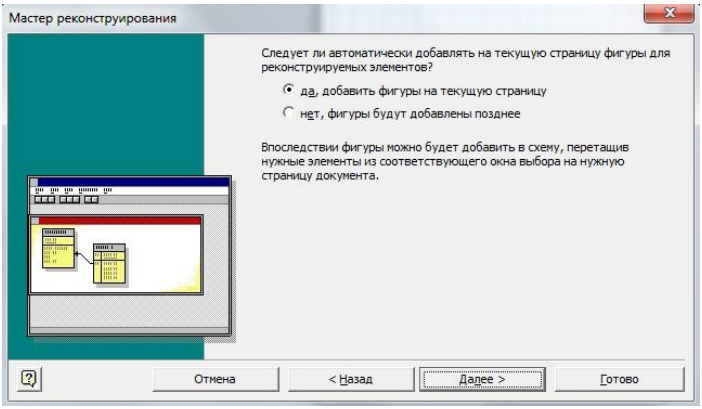


Рисунок 50 – Добавление фигур на страницу

Откроется последнее окно мастера реконструирования (Рисунок 51). Просмотрите выбранные параметры, чтобы убедиться в том, что будут извлечены все необходимые данные, и нажмите кнопку «Готово».

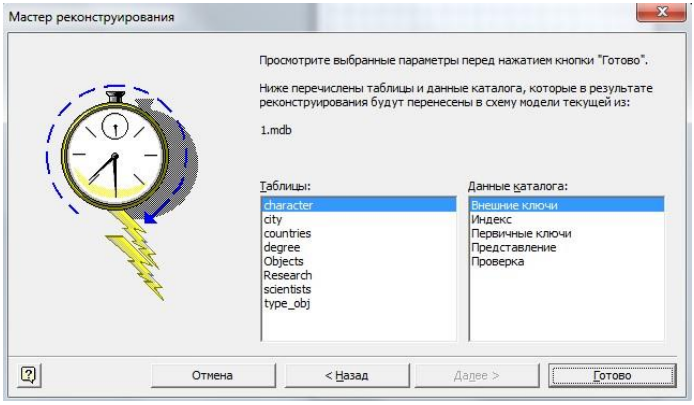


Рисунок 51 – Просмотр выбранных параметров

Если используется универсальный драйвер ODBC, иногда появляется сообщение об ошибке, которое означает, что реконструированная информация может быть неполной. В большинстве случаев это не указывает на реальную проблему, так что можно нажать кнопку «ОК», чтобы продолжить работу мастера.

Мастер извлекает выбранные данные и отображает замечания о процедуре извлечения в окне вывода. т.е. должна получиться схема модели базы данных.

Работа считается выполненной, если на основе созданной ранее БД, получена модель данных по изложенной выше методике.

Допускается проведение реконструкции схемы БД в среде автоматизированного проектирования, отличной от MS Visio. В качестве альтернативы, студент может провести проектирование физической модели данных и схемы БД для созданной ранее ER-диаграммы.

2.6 Лабораторная работа «Сравнение информационных систем по критерию функциональной полноты»

Цель работы

Овладение навыками сравнения информационных систем.

Форма проведения

Выполнение индивидуального задания.

Форма отчетности

На проверку должен быть представлен отчет, оформленный по требованиям ОС ТУСУР 01-2021 https://regulations.tusur.ru/documents/70

Теоретические основы

Методика сравнения информационных систем

Необходимость проведения сравнительного анализа программных продуктов возникает как перед потенциальным пользователем в случае приобретения системы, так и перед разработчиком при создании собственной системы с целью изучения уже существующих наработок в этой предметной области. Для такого анализа необходимы или рабочие копии конкурирующих продуктов, или, по крайней мере, их демонстрационные версии или описания, если ничего больше достать не удастся. Составляется перечень их функций, сильных и слабых сторон и тех характеристик, которые отмечаются в прессе и профессиональных изданиях как достоинства и недостатки этих продуктов. Производится классификация продуктов.

Продукты разделяются по занимаемым ими сегментам рынка или по специфическим назначениям. Затем составляется детальный отчет обо всех продуктах, включая и те, появление которых на рынке только предполагается. В отчет включается четко структурированное описание каждого продукта, и такое же описание составляется для будущего продукта компании.

На основании отобранных таким образом данных можно ответить на ключевой вопрос проводимого анализа — какая из систем является предпочтительной в использовании.

Ниже приводится методика выбора (оценки) автоматизированных информационных систем, основанная на проверке соответствия функциональной полноты системы требованиям пользователя или некоторому эталону [2].

Пусть $Z = \{Zi\}$ $(i = 1, 2, \ldots, n)$ — множество сравниваемых АИС;

$R = \{Rj\}$ $(j = 1, 2, \ldots, m)$ — множество, составляющее словарь реализуемых АИС функций $\{Zi\}$.

Исходная информация представляется в виде таблицы $\{Xij\}$, элементы которой определяются следующим образом:

$$X_{ij} = \begin{cases} 1, \text{если } j-\text{я функция реализуетс я } i-\text{й АИС;} \\ 0, \text{если не реализуетс я.} \end{cases}$$

Выделим системы Zi и Zk (i, k =1, 2, …, n) и введем следующие обозначения:

$P_{ik}^{(11)}$ — число функций, выполняемых и Zi и Zk, то есть

$P_{ik}^{(11)}$ =| Zi ∩ Zk| — мощность пересечения множеств Zi = {Xij} и Zk = {Xkj} (j∈ m; x|xij ∧ xkj = 1);

$P_{ik}^{(10)}$ — число функций, выполняемых Zi, но не реализу-емых Zk, то есть

$P_{ik}^{(10)}$ = |Zi\Zk| — мощность разности множеств Zi= {Xij} и Zk ={Xkj};

$P_{ik}^{(01)}$ — число функций, выполняемых Zk но не реализу-емых Zi, то есть

$P_{ik}^{(01)}$ = |Zk\Zi| — мощность разности множеств Zk и Zi;

$P_{ik}^{(00)}$ = |Zi ∪ Zk| — мощность объединения множеств Zi и Zk, то есть

$P_{ik}^{(00)} = P_{ik}^{(11)} + P_{ik}^{(10)} + P_{ik}^{(01)}$.

Для оценки того, какая часть (доля) функций, выполняемых АИС Zi, реализуется также АИС Zk можно использовать следующую величину:

$$H_{ik} = P_{ik}^{(11)} / (P_{ik}^{(11)} + P_{ik}^{(10)}), \quad (0 \le H_{ik} \le 1).$$

Взаимосвязь между АИС Zi и Zk оценивается по значениям $P_{ik}^{(11)}$ и $G_{ik} = P_{ik}^{(11)} / P_{ik}^{(00)}$, $(0 \le G_{ik} \le 1)$, где G_{ik} — «мера подобия».

Выбирая различные пороговые значения матриц G и H, можно построить логические матрицы поглощения (включения) G0, H0. Например, элементы матрицы H0 получим следующим образом:

$$H_{ik}^{0} = \begin{cases} 1, \text{если } H_{ik}^{0} \ge \varepsilon_h, i \ne k; \\ 0, \text{если } H_{ik}^{0} < \varepsilon_h, \text{или } i = k. \end{cases}$$

$$G_{ik}^{0} = \begin{cases} 1, \text{если } G_{ik}^{0} \ge \varepsilon_g, i \ne k; \\ 0, \text{если } G_{ik}^{0} < \varepsilon_g \text{ или } i = k. \end{cases}$$

Граф, построенный по логическим матрицам G0 и H0, дает наглядное представление о взаимосвязи между сравниваемыми АИС (по выполняемым функциям).

Строку с перечнем функций, которые в идеале должна выполнять система, обозначим через Ze.

Дополнив таблицу {Xij} (i ∈ n, j ∈ m) строкой Xej (j ∈ m), рассчитаем матрицы P(01), P(11) и, выделив строки, у которых $P_{ej}^{(10)} = 0$, получим перечень АИС, полностью удовлетворяющих требованиям к функциональной полноте программного средства.

Приведем пример экспериментального исследования методики. Для этого определим функции и параметры информационных систем автоматизированного документооборота, наиболее широко представленных на рынке.

Характеристики сравниваемых ниже систем определялись на основе материалов открытой печати, изданий по компьютерной тематике (Мир ПК, Открытые Системы, Computerworld Россия, PC Week/RE, КомпьютерПресс и др.) материалов конференций, выставок, семинаров; рекламных материалов фирм-производителей; материалов, размещаемых в сети Интернет.

В таблице 3 перечислены параметры и функции систем, а также параметры и функции так называемой системы эталона, наличие которых в системе делопроизводства и документооборота способствует полной автоматизации этих процессов в организации.

Таблица 3 – Сводная таблица параметров и функций систем автоматизации документооборота и делопроизводства

№	Параметры	Системы автоматизации делопроизводства и документооборота				
		КОРД	Дело	LanDocs	Золушка	Система эталон
	Виды документов, регистрируемых в системе					
1.	Входящие	1	1	1	1	1
2.	Исходящие	1	1	1	1	1
3.	Внутренние	1	1	1	1	1
4.	Обращения граждан	1	1	0	1	1
	Общие реквизиты регистрационной карточки					
5.	Регистрационный номер документа	1	1	1	1	1
6.	Дата регистрации	1	1	1	1	1
7.	Код рубрики темы	1	1	0	1	1
8.	Краткое содержание документа	1	1	1	1	1
9.	Номер дела	1	1	1	1	1
10.	Ключевые слова	0	0	0	1	0
11.	Реквизиты резолю-ции по документу	1	1	1	1	1
12.	Реквизиты конт-рольной службы	1	1	1	1	1
13.	Реквизиты архивного хранения	1	1	1	0	1
	Реквизиты организации-корреспондента					

№	Параметры	Системы автоматизации делопроизводства и документооборота				
		КОРД	Дело	LanDocs	Золушка	Система эталон
14.	Наименование органи-зации-корреспондента	1	1	0	1	1
15.	Исходящий номер	1	1	1	1	1
16.	Исходящая дата	1	1	1	1	1
17.	Подпись	1	1	1	1	1
	Регистрация входящих документы					
18.	Кому адресован	1	1	0	1	1
19.	Вид доставки	1	1	1	0	1
20.	Отметка о наличии приложений (связан-ные документы)	1	1	1	1	1
21.	Признак повторности	1	1	1	1	1
22.	Тип документа	1	0	0	0	1
	Регистрация сопроводительные документы					
23.	Аннотация	1	1	1	1	1
24.	Корреспондент	1	1	1	1	1
25.	Исходящий номер	1	1	1	1	1
26.	Исходящая дата	1	1	1	1	1
27.	Кто подписал	1	1	0	1	1
28.	Исполнитель	1	0	0	1	1
	Регистрация писем и обращений граждан					
29.	Корреспондент	1	1	0	0	1
30.	Признак коллективности	1	1	0	0	1
	Регистрация исходящих документов					
31.	Кому адресован	1	1	0	1	1
32.	Кто подписал	1	1	0	1	1
33.	Подразделение- автор	1	1	0	1	1
34.	ФИО исполнителя	1	1	0	1	1
35.	Ссылка на номер входящего документа	1	1	1	1	1
36.	Ссылка на документ	1	0	0	1	1
37.	Вид отправки	1	0	1	0	1
	Контроль исполнения документов					
38.	Сведения о исполнителе	1	1	1	1	1
39.	Гриф утверждения	1	1	1	0	1
40.	Текст задания	1	1	1	0	1
41.	Контролер	1	0	1	1	1
42.	Выделение ответственного исполнителя	1	1	1	0	1
43.	Методы предупреж-дающего контроля и механизм	1	0	0	0	1

№	Параметры	Системы автоматизации делопроизводства и документооборота				
		КОРД	Дело	LanDocs	Золушка	Система эталон
	поддержки принятия решений					
	Сроки исполнения документов					
44.	Поступление к исполнению	1	1	1	1	1
45.	Плановый срок	1	1	1	1	1
46.	Фактический срок	1	1	1	1	1
47.	Напоминание для просроченных	1	1	0	1	1
	Поиск документов					
	Поиск по атрибутам регистрационной карточки					
48.	Группа документов	1	1	1	1	1
49.	Дата документа	1	1	0	1	1
50.	Тематический рубрикатор	1	1	0	1	1
51.	Фильтры поиска	1	1	0	1	1
52.	Критерии поиска для входящих	1	1	0	1	1
53.	Критерии поиска для исходящих	1	1	0	1	1
	Поиск по регистрационным номерам					
54.	Группа документов	1	1	1	1	0
55.	Номер документа	1	1	1	1	1
56.	Год регистрации	1	1	0	1	1
57.	Подразделение	0	0	0	1	0
	Формирование отчетов					
58.	Сведения о документообороте за заданный период времени	1	1	0	1	1
59.	Сводка об исполнении контрольных документов	1	1	1	1	1
60.	Справка-напоминание об исполнении контрольных документов	1	1	0	1	1

По вышеописанному алгоритму рассчитаем следующие матрицы:

$$P^{(01)} = \begin{vmatrix} 0 & 0 & 0 & 2 & 0 \\ 6 & 0 & 2 & 5 & 5 \\ 24 & 20 & 0 & 22 & 24 \\ 10 & 7 & 6 & 0 & 10 \\ 2 & 1 & 2 & 3 & 0 \end{vmatrix}; \quad P^{(10)} = \begin{vmatrix} 0 & 6 & 24 & 10 & 2 \\ 0 & 0 & 20 & 7 & 1 \\ 0 & 2 & 0 & 6 & 2 \\ 2 & 5 & 22 & 0 & 3 \\ 0 & 5 & 24 & 10 & 0 \end{vmatrix};$$

$$P^{(11)} = \begin{vmatrix} 56 & 52 & 32 & 41 & 56 \\ 52 & 52 & 32 & 45 & 51 \\ 32 & 32 & 34 & 28 & 32 \\ 41 & 45 & 28 & 50 & 47 \\ 56 & 51 & 32 & 47 & 56 \end{vmatrix}; \quad P^{(00)} = \begin{vmatrix} 56 & 58 & 56 & 53 & 58 \\ 58 & 52 & 54 & 57 & 57 \\ 56 & 54 & 34 & 56 & 58 \\ 53 & 57 & 56 & 50 & 60 \\ 58 & 57 & 58 & 60 & 56 \end{vmatrix}.$$

При использовании порогового значения ε_h = 0,8 получим логическую матрицу поглощения H^0.

$$H = \begin{vmatrix} 1 & 0{,}9 & 0{,}7 & 0{,}84 & 0{,}97 \\ 1 & 1 & 0{,}6 & 0{,}87 & 0{,}98 \\ 1 & 0{,}94 & 1 & 0{,}82 & 0{,}94 \\ 0{,}96 & 0{,}9 & 0{,}56 & 1 & 0{,}92 \\ 1 & 0{,}91 & 0{,}57 & 0{,}82 & 1 \end{vmatrix}; \quad H^0 = \begin{vmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{vmatrix}.$$

При использовании порогового значения ε_g = 0,75 получим логическую матрицу подобия G^0.

$$G = \begin{vmatrix} 1 & 0{,}9 & 0{,}58 & 0{,}77 & 0{,}97 \\ 0{,}9 & 1 & 0{,}6 & 0{,}8 & 0{,}9 \\ 0{,}58 & 0{,}6 & 1 & 0{,}5 & 0{,}55 \\ 0{,}77 & 0{,}8 & 0{,}5 & 1 & 0{,}77 \\ 0{,}97 & 0{,}9 & 0{,}55 & 0{,}77 & 1 \end{vmatrix}; \quad G^0 = \begin{vmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{vmatrix}.$$

По матрицам G0 и H0 построим графы подобия (рис. 52) и поглощения (рис. 53), соответственно.

Из полученных графов можно сделать вывод, что при выбранных коэффициентах подобия и поглощения системами, в наибольшей мере отвечающими требованиям к технологии документооборота и делопроизводства, являются системы «КОРД» и «Дело». Однако при этом необходимо отметить, что в данном случае были выбраны средние коэффициенты подобия и поглощения (εg = 0,75 и εh = 0,8).

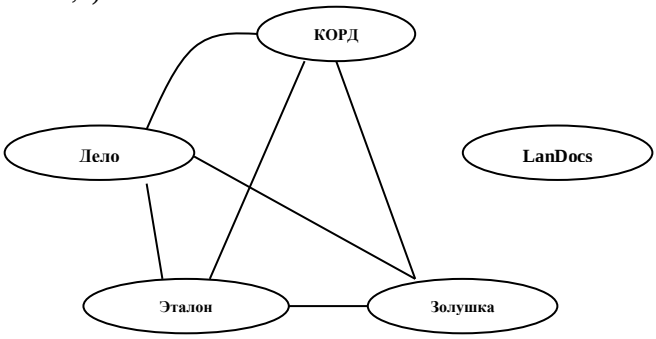


Рисунок – 52 Граф подобия

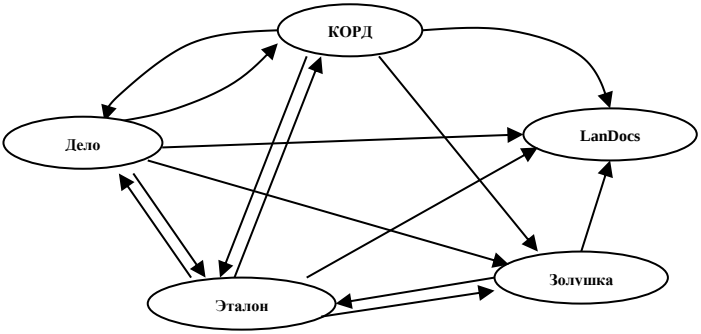


Рисунок – 53 Граф поглощения

В принципе допускается варьирование коэффициентов $0{,}5 \leq \varepsilon g \leq 1$ и $0{,}5 \leq \varepsilon h \leq 1$. Явно видно, что при использовании максимальных коэффициентов условия подобия и поглощения не соблюдаются, а при наименьших значениях предложенных коэффициентов все рассматриваемые системы в целом могут быть использованы для автоматизации документооборота и делопроизводства. С целью выбора системы, в наибольшей мере отвечающей требованиям потребителя, необходимо выбирать коэффициенты подобия и поглощения близкими к единице.

Применение рассмотренной выше методики позволяет проводить сравнительный анализ любых однотипных автоматизированных информационных систем и делать вывод о предпочтении использования системы и ее соответствии требованиям пользователя или системе эталону.

Порядок выполнения работы.

1. Выбрать для сравнения не менее трех информационных систем. Определить функции эталонной системы. Выявить базовые функции систем, сгруппировав их по общему назначению.
2. Построить таблицу, содержащую перечень функций с отметкой о наличии конкретной функции в системе. Построить матрицы по описанной выше методике.
3. Построить матрицы подобия и поглощения, выбрав оптимальные коэффициенты подобия и поглощения. Построить графы подобия и поглощения.
4. Сделать выводы о предпочтительном использовании той или иной системы.
5. Результаты выполнения данной работы необходимо представить в виде отчет, оформленного по требованиям ОС ТУСУР 01-2013 http://www.tusur.ru/export/sites/ru.tusur.new/ru/education/documents/inside/tech_01-2013_new.pdf

3 Указания к выполнению курсового проекта[1]

Цели работы

Освоение методики проектирования концептуальной информационной модели предметной области, создание физической структуры базы данных, разработка пользовательского приложения. Закрепление теоретических знаний по курсу организация баз данных.

Задачи курсового проекта:

– формализовать исходное описание предметной области;

– построить концептуальную информационную модель, используя методику, изученную в рамках теоретического курса;

– сгенерировать физическую структуру базы данных;

– реализовать пользовательское приложение, представляющее собой информационную систему, взаимодействующую с разработанной БД и демонстрирующее накопленные студентом знания по курсу «Организация баз данных».

Средства выполнения и форма отчетности

Проектирование модели предметной области выполняется средствами автоматизированного проектирования и графического представления (MS Visio и др.). При отсутствии средств моделирования концептуальную и физическую модель необходимо реализовать средствами MS Word.

База данных разрабатывается в среде любой современной СУБД (MS Access и др.).

Пользовательское приложение может быть создано либо средствами выбранной СУБД, либо с помощью любых языков программирования.

Результаты выполнения работы представляются в пояснительной записке, подготовленной в среде MS Word. Пользовательское приложение и базу данных необходимо представить вместе с пояснительной запиской к курсовому проекту.

Варианты предметных областей, для которых должна быть создана база данных, представлены в таблице 1.

Порядок выполнения работы

1. Разработка технического задания (ТЗ)

Каждый студент получает для работы вариант предметной области (Таблица 1). В ходе выполнения курсового проекта необходимо провести анализ предметной области и создать ТЗ на разрабатываемую информационную систему.

Техническое задание является основным документом, в соответствии с которым проводят создание программного продукта (в т.ч. АИС и др. программных изделий) и приемку его заказчиком.

ТЗ создается в соответствии с ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению».

Данный стандарт устанавливает порядок построения и оформления технического задания на разработку программы или программного изделия для

[1] Для студентов, обучающихся на направлении «Программная инженерия»

вычислительных машин, комплексов и систем независимо от их назначения и области применения. Стандарт полностью соответствует СТ СЭВ 1627-79.

Техническое задание оформляют в соответствии с ГОСТ 19.106-78. Номера листов (страниц) проставляются в верхней части листа над текстом. Лист утверждения и титульный лист оформляют в соответствии с ГОСТ 19.104-78.

Информационную часть (аннотацию и содержание), лист регистрации изменений допускается в документ не включать.

Для внесения изменений или дополнений в техническое задание на последующих стадиях разработки программы или программного изделия выпускают дополнение к нему. Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

Техническое задание должно содержать следующие разделы:

- введение;
- основания для разработки;
- назначение разработки;
- требования к программе или программному изделию;
- требования к программной документации;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки;
- в техническое задание допускается включать приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

В разделе «Введение» указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие.

В разделе «Основания для разработки» должны быть указаны:

- документ (документы), на основании которых ведется разработка;
- организация, утвердившая этот документ, и дата его утверждения;
- наименование и (или) условное обозначение темы разработки.

В разделе «Назначение разработки» должно быть указано функциональное и эксплуатационное назначение программы или программного изделия.

Раздел «Требования к программе или программному изделию» должен содержать следующие подразделы:

- требования к функциональным характеристикам;
- требования к надежности;
- условия эксплуатации;
- требования к составу и параметрам технических средств;
- требования к информационной и программной совместимости;
- требования к маркировке и упаковке;

– требования к транспортированию и хранению;
– специальные требования.

В подразделе «Требования к функциональным характеристикам» должны быть указаны требования к составу выполняемых функций, организации входных и выходных данных, временным характеристикам и т. п.

В подразделе «Требования к надежности» должны быть указаны требования к обеспечению надежного функционирования (обеспечения устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа и т.п.).

В подразделе «Условия эксплуатации» должны быть указаны условия эксплуатации (температура окружающего воздуха, относительная влажность и т.п. для выбранных типов носителей данных), при которых должны обеспечиваться заданные характеристики, а также вид обслуживания, необходимое количество и квалификация персонала.

В подразделе «Требования к составу и параметрам технических средств» указывают необходимый состав технических средств с указанием их основных технических характеристик.

В подразделе «Требования к информационной и программной совместимости» должны быть указаны требования к информационным структурам на входе и выходе и методам решения, исходным кодам, языкам программирования и программным средствам, используемым программой. При необходимости должна обеспечиваться защита информации и программ.

В подразделе «Требования к маркировке и упаковке» в общем случае указывают требования к маркировке программного изделия, варианты и способы упаковки.

В подразделе «Требования к транспортированию и хранению» должны быть указаны для программного изделия условия транспортирования, места хранения, условия хранения, условия складирования, сроки хранения в различных условиях.

В разделе «Требования к программной документации» должен быть указан предварительный состав программной документации и, при необходимости, специальные требования к ней.

В разделе «Технико-экономические показатели» должны быть указаны: ориентировочная экономическая эффективность, предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами.

В разделе «Стадии и этапы разработки» устанавливают необходимые стадии разработки, этапы и содержание работ (перечень программных документов, которые должны быть разработаны, согласованы и утверждены), а также, как правило, сроки разработки и определяют исполнителей.

В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.

В приложениях к техническому заданию, при необходимости, приводят:

– перечень научно-исследовательских и других работ, обосновывающих разработку;

- схемы алгоритмов, таблицы, описания, обоснования, расчеты и другие документы, которые могут быть использованы при разработке;
- другие источники разработки.

Разработанное техническое задание должно быть утверждено руководителем курсового проекта.

2. Создание концептуальной информационной модели предметной области

Концептуальная модель представляется в виде набора ER-диаграмм. Осуществляется формализация исходного описания в виде набора сущностей с последующим их преобразованием и связыванием в концептуальную модель.

Процесс проектирования сопровождается составлением ряда сущностей, необходимыми пояснениями – обоснованиями принимаемых решений

Проектирование концептуальной модели предметной области целесообразно производить с помощью специального средства проектирования (например, MS Visio).

Основные этапы проектирования концептуальной модели:

1. Первичный анализ информационных потребностей пользователей, выделение объектов предметной области и формирование исходных сущностей:
   - анализ информационных документов;
   - анализ конкретных информационных потребностей (запросов) пользователей.
2. Проектирование исходных сущностей:
   - определение атрибутов сущностей и их типов данных;
   - нормализация сущностей до 3 НФ.
3. Связывание сущностей в концептуальную информационную модель:
   - определение уникальных идентификаторов сущностей (первичных ключей);
   - определение связей между сущностями.

Ограничения концептуальной модели:

- предметная область должна быть описана 8-10 взаимосвязанными сущностями;
- каждая сущность должна содержать не менее 3 атрибутов (не считая справочников-классификаторов);
- в каждой сущности должен быть определен уникльный идентификатор сущности.

3. Создание физической модели данных

На основе спроектированной концептуальной модели может быть создана физическая модель данных, свойственная для конкретной СУБД.

При формировании физической модели определяются внешние ключи в связываемых отношениях. Добавляются промежуточные таблицы связи, с целью исключения связей многие-ко-многим (M:M).

Большинство автоматизированных средств проектирования позволяют произвести автоматическую генерацию физической модели на основе созданной концептуальной. При отсутствии таковых средств физическая модель создается

вручную с последующим ее отражением в структурной части базы данных конкретной СУБД.

4. Создание пользовательского приложения

Приложение, работающее с созданной базой данных должно обеспечивать выполнение следующих функций:

- ввод информации в БД;
- удаление информации из БД;
- редактирование внесенной информации;
- выборка (поиск) данных по таблицам БД с использованием различных критериев;
- формирование отчетов и вывод информации из базы данных на экран и на принтер;

Добавление, замена и удаление информации должны производиться в экранных формах разрабатываемого пользовательского приложения.

5. Подготовка и проведение приемочных испытаний

Приемочные испытания информационной системы, разработанной в ходе выполнения курсового проекта, проводятся в соответствии с разработанной программой и методикой проведения приемочных испытаний.

Цель составление программы и методики проведения испытаний является – подготовка документа, на основании которого будет подтверждено соответствие характеристик созданной информационной системы всем требованиям, заданным ТЗ, в условиях, максимально приближенных к условиям реальной эксплуатации (применения, использования), а также для подтверждения эксплуатационной пригодности комплекса.

Программа и методика приемочных испытаний создаются в соответствии с 19.301-79 «Программа и методика испытаний. Требования к содержанию и оформлению». Структура и оформление документа устанавливается в соответствии с ГОСТ 19.105-78.

Составление информационной части (аннотации и содержания) является необязательным.

Документ «Программа и методика испытаний» должен содержать следующие разделы:

- объект испытаний;
- цель испытаний;
- требования к программе;
- требования к программной документации;
- состав и порядок испытаний;
- методы испытаний.

В зависимости от особенностей документа допускается вводить дополнительные разделы.

Содержание разделов

В разделе «Объект испытаний» указывают наименование, область применения и обозначение испытуемой программы.

В разделе «Цель испытаний» должна быть указана цель проведения

испытаний.

В разделе «Требования к программе» должны быть указаны требования, подлежащие проверке во время испытаний и заданные в техническом задании на программу.

В разделе «Требования к программной документации» должны быть указаны состав программной документации, предъявляемой на испытания, а также специальные требования, если они заданы в техническом задании на программу.

В разделе «Средства и порядок испытаний» должны быть указаны технические и программные средства, используемые во время испытаний, а также порядок проведения испытаний.

В разделе «Методы испытаний» должны быть приведены описания используемых методов испытаний. Методы испытаний рекомендуется по отдельным показателям располагать в последовательности, в которой эти показатели расположены в разделах «Требования к программе» и «Требования к программной документации».

В методах испытаний должны быть приведены описания проверок с указанием результатов проведения испытаний (перечней тестовых примеров, контрольных распечаток тестовых примеров и т. п.).

В приложение к документу могут быть включены тестовые примеры, контрольные распечатки тестовых примеров, таблицы, графики и т. п.

Необходимо разработать программу и методики приемочных испытаний.

Провести приемочные испытания программного продукта в соответствии с разработанной ранее программой и методикой приемочных испытаний.

Результат испытаний необходимо представить в виде акта и протоколов испытаний (Приложение 3) и предоставить их до защиты курсового проекта.

6. Разработка эксплуатационной документации

К эксплуатационной документации на программный продукт относятся следующие документы:

- Ведомость эксплуатационных документов.
- Описание применения.
- Формуляр.
- Руководство оператора (пользователя).
- Руководство системного программиста.
- Руководство программиста.

Данные документы создаются в соответствии с действующими ГОСТами серии 19.50х.хх.

В рамках выполнения курсового проекта необходимо разработать руководство оператора (пользователя) в соответствии с ГОСТ 19.505-79. Данный стандарт устанавливает требования к содержанию и оформлению программного документа «Руководство оператора», определённого ГОСТ 19.101-77.

Структуру и оформление документа устанавливают в соответствии с ГОСТ 19.105-78.

Составление информационной части (аннотации и содержания) является обязательным.

Руководство оператора должно содержать следующие разделы:

- назначение программы;
- условия выполнения программы;
- выполнение программы;
- сообщения оператору.

В зависимости от особенностей документы допускается объединять отдельные разделы или вводить новые.

В разделе «Назначение программы» должны быть указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.

В разделе «Условия выполнения программы» должны быть указаны условия, необходимые для выполнения программы (минимальный и (или) максимальный состав аппаратурных и программных средств и т.п.).

В разделе «Выполнение программы» должна быть указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузки и управляет выполнением программы, а также ответы программы на эти команды.

В разделе «Сообщения оператору» должны быть приведены тексты сообщений, выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия оператора (действия оператора в случае сбоя, возможности повторного запуска программы и т.п.).

Допускается содержание разделов иллюстрировать поясняющими примерами, таблицами, схемами, графиками.

В приложения к руководству оператора допускается включать различные материалы, которые нецелесообразно включать в разделы руководства.

7. Оформление пояснительной записки

Пояснительная записка оформляется в соответствие с действующими стандартами ТУСУР. Текст стандарта доступен на официальном портале ТУСУР.

Пояснительная записка к курсовому проекту должна включать: титульный лист, лист задания на курсовую работу, содержание, введение, основную часть, заключение, список использованных литературных источников, приложение.

Титульный лист оформляется согласно действующим стандартам.

Введение должно содержать цель выполняемой курсового проекта, основные принципы, положенные в основу ее проведения, область применения.

В основной части должен быть отражен процесс и результат проектирования базы данных и пользовательского приложения. Основная часть должна содержать:

- описание предметной области;
- описание и обоснование выбранного средства реализации (СУБД, средства проектирования, программной среды написания приложения);
- концептуальную информационную модель предметной области с полным описанием выделенных сущностей;
- физическую модель базы данных;

– описание пользовательского приложения.

Заключение должно содержать краткие выводы по результатам выполненной работы.

Список использованных литературных источников оформляется согласно действующим стандартам.

В приложении приводятся: техническое задание на разработку информационной системы, экранные и отчетные формы приложения, тексты SQL-запросов, создаваемых в информационной системе, руководство оператора и другая информация.

4 Методические указания по организации самостоятельной работы

4.1 Общие положения

Целями самостоятельной работы является систематизация, расширение и закрепление теоретических знаний, приобретение навыков научно-исследовательской и производственно-технологической деятельности.

Самостоятельная работа по дисциплине «Организация баз данных» включает следующие виды активности студента:

- проработка лекционного материала;
- изучение тем (вопросов) теоретической части дисциплины, вынесенных для самостоятельной подготовки;
- подготовка к лабораторным работам;
- выполнение индивидуального задания;
- подготовка к экзамену.

4.2 Проработка лекционного материала

Для проработки лекционного материала студентам рекомендуется воспользоваться конспектом, сопоставить записи конспекта с соответствующими разделами методического пособия [1].

Целесообразно ознакомиться с информацией, представленной в файлах, содержащих презентации лекций, предоставляемых преподавателем.

Для проработки лекционного материала студентам, помимо конспектов лекций, рекомендуются следующие главы учебно-методического пособия [1] по разделам курса:

- обоснование концепции баз данных – глава 1;
- модели данных – глава 2;
- реляционная модель – глава 3;
- технология проектирования реляционных баз данных – глава 4;
- языки управления и манипулирования данными – глава 5;
- физическая организация баз данных – глава 6;
- системы управления базами данных – глава 7;

При изучении учебно-методического пособия [1] студенту рекомендуется самостоятельно ответить на вопросы, приводимые в конце каждой главы. Рекомендуется сформулировать вопросы преподавателю и задать их либо посредством электронной образовательной среды вуза, либо перед началом следующей лекции.

4.3 Изучение тем (вопросов) теоретической части дисциплины, вынесенных для самостоятельной подготовки

4.3.1 Реляционное исчисление

Перечень вопросов, подлежащих изучению

- исчисление кортежей;
- исчисление доменов;
- понятие переменной с определенной для нее областью допустимых

значений;
- понятие правильно построенной формулы.

Методические рекомендации по изучению

Рекомендуется представить в виде выражений реляционного исчисления запросы на выборку, созданные при помощи языка SQL, реализованные в ходе выполнения лабораторной работы «Создание SQL-запросов».

Рекомендуемые источники

Для подготовки к изучению реляционного исчисления необходимо ознакомиться с материалом, изложенным в главе 3.4.3 учебного пособия [1].

4.3.2 Дополнительные элементы ER-модели

Перечень вопросов, подлежащих изучению

- изучение понятия домены;
- изучение супертипов сущностей;
- изучение подтипов сущностей;
- этапы получения схемы БД.

Методические рекомендации по изучению

Рекомендуется создать домены, супертипы и подтипы сущностей в ER-модели, созданной в ходе выполнения лабораторной работы «Создание концептуальной модели данных».

Рекомендуемые источники

Для подготовки к изучению дополнительных элементов ER-модели необходимо ознакомиться с материалом, изложенным в главе 4.2.3 учебного пособия [1].

4.3.3 Получение схемы реляционной базы данных из ER-диаграммы

Перечень вопросов, подлежащих изучению

- этапы получения схемы БД.

Методические рекомендации по изучению

Рекомендуется создать скрипт на языке SQL, содержащий запросы на создание основных и дополнительных объектов БД, представленных в ER-модели. Запуск данного скрипта должен создать схему БД в выбранной СУБД.

Рекомендуемые источники

Для подготовки к изучению технологии получения схемы БД из ER-модели необходимо ознакомиться с материалом, изложенным в главах 4.2.4 и 5.1 учебного пособия [1].

4.3.4 СУБД Caché

Перечень вопросов, подлежащих изучению

- назначение СУБД Caché;
- архитектура СУБД Caché;
- основные компоненты СУБД Caché;

- сервер Caché Objects;
- объектная модель Caché;
- сервер Caché SQL;
- сервер прямого доступа (Caché Direct).

Методические рекомендации по изучению

В рамках изучения возможностей СУБД Caché рекомендуется повторить подраздел «Общие понятия объектно-ориентированного подхода к базам данных» настоящей дисциплины.

Рекомендуемые источники

Для подготовки к изучению возможностей СУБД Caché необходимо ознакомиться с материалом, изложенным в главе 7.3.4 учебного пособия [1] и с информацией, представленной на официальном сайте разработчика СУБД Caché: http://www.intersystems.com/.

4.4 Подготовка к лабораторным работам

Для подготовки к лабораторной работе «Организация хранения данных в MS Access» студенту необходимо:

- изучить разделы 1, 2.1, 4 учебного пособия [1];
- выбрать предметную область;
- провести анализ предметной области и выявить основные объекты предметной области;
- спроектировать и согласовать с преподавателем схему базы данных.

Для подготовки к лабораторной работе «Создание элементов пользовательского интерфейса в среде MS Access» студенту необходимо:

- изучить разделы 1, 2.1, 7.2 учебного пособия [1];
- спроектировать предполагаемые формы и согласовать их наполнение элементами управления с преподавателем;

Для подготовки к лабораторной работе «Построение SQL-запросов в среде MS Access» студенту необходимо:

- изучить раздел 5.1 учебного пособия [1];
- спроектировать предполагаемые SQL-запросы;
- согласовать с преподавателем содержание создаваемых SQLзапросов.

Для подготовки к лабораторной работе «Создание концептуальной модели данных в среде автоматизированного проектирования MS Visio» студенту необходимо:

- изучить разделы 4.2, 4.3 учебного пособия [1];
- спроектировать и согласовать с преподавателем ER-диаграммы выбранной предметной области.

Для подготовки к лабораторной работе «Реконструкция схемы базы данных» студенту необходимо:

- изучить разделы 4.2, 4.3, 6 учебного пособия [1];
- подготовить базу данных, которая будет подвержена реконструкции, проверив ее на соответствие 3 НФ.

Для подготовки к лабораторной работе «Сравнение информационных

систем по критерию функциональной полноты» студенту необходимо:

- изучить раздел 2.5 учебного пособия [2];
- выбрать предметную область и изучить функциональные характеристики информационных систем выбранной предметной области по данным открытых источников.

4.5 Выполнение индивидуального задания

Цель индивидуального задания

Проведение полного цикла нормализации отношения выбранной предметной области.

Порядок выполнения и содержание работы

1. Студенту необходимо выбрать предметную область и представить ненормализованное по 1-НФ отношение реляционной модели данных.

2. Провести нормализацию исходного отношения до 3 НФ, путем последовательной декомпозиции, выявив все возможные первичные ключи и функциональные зависимости, как в исходном, так и в результирующих отношениях.

3. Результат выполнения работы может быть представлен либо в печатном, либо в электронном виде. Допускается представление результата в виде презентации.

4. В качестве варианта предметной области может быть выбрана предметная область для лабораторной работы. Не допускается заимствование примера нормализации из учебного пособия.

Рекомендуемые источники

Для подготовки к выполнению индивидуального задания необходимо ознакомиться с материалом, изложенным в главе 4.1 учебного пособия [1].

4.6 Подготовка к зачету с оценкой

Для подготовки к зачету с оценкой рекомендуется повторить соответствующие тематике разделы учебно-методического пособия [1].

Вопросы представлены в рабочей программе изучаемой дисциплине, размещенной на образовательном портале ТУСУРа: https://edu.tusur.ru/.

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]. – URL: https://edu.tusur.ru/training/publications/5179

2. Сенченко П.В. Надежность, эргономика и качество АСОИУ: Учебное пособие. – Томск: Томск. гос. ун-т систем управления и радиоэлектроники, 2016. – 189 с. [Электронный ресурс]. – URL: https://edu.tusur.ru/training/publications/6066

Приложение 1 Пример оформления титульного листа

Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«**Томский государственный университет систем управления и радиоэлектроники» (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

Автоматизированная информационная система
«Наименование предметной области»
Курсовой проект по дисциплине «Организация баз данных»

Студент гр. <номер>

_____И.О. Фамилия
«_____»_____20__ г.

Руководитель

Доцент кафедры АОИ
канд. техн. наук, доцент
_____П.В. Сенченко
«___»_____20__ г.

Томск 20__

Приложение 2 Пример оформления листа задания

Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«**Томский государственный университет систем управления и радиоэлектроники» (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ
Заведующий кафедрой
АОИ канд. экон. наук,
доцент
_____А.А. Сидоров
«_____» 20__ г.

ЗАДАНИЕ

курсовой проект по дисциплине «Организация баз данных»

студенту <**Фамилия Имя Отчество**> группы <номер группы> факультета систем управления

1. Тема проекта:
   _____
2. Срок сдачи на кафедру:
   _____
3. Содержание работы:
   _____
   _____
   _____

Руководитель: доцент кафедры АОИ, канд. техн. наук, доцент
Сенченко Павел Васильевич_____«_»_____20__ г.
(подпись руководителя)

Задание принял к исполнению:
Фамилия Имя Отчество_____«___»_____20__ г.
(подпись студента)

Приложение 3. Акт приемочных испытаний

УТВЕРЖДАЮ

Должность руководителя
организации-исполнителя
_____ *И.О. Фамилия*
«___»_____ 20__ г.

АКТ
приемочных испытаний
опытного образца *наименование в родительном падеже и обозначение в соответствии с основным конструкторским/технологическим /программным документом*

«___»_____20__ г. г. *[Город]*

Комиссия по проведению приемочных испытаний в составе:

председатель	*Должность в организаци*	*Фамилия И.О. (полностью)*
членов комиссии	*Должность в организаци*	*Фамилия И.О (полностью)*
	Должность в организаци	*Фамилия И.О. (полностью)*
	Должность в организаци	*Фамилия И.О. (полностью)*
	Должность в организаци	*Фамилия И.О. (полностью)*

провела приемочные испытания опытного образца программного продукта *наименование в родительном падеже и обозначение в соответствии с основным конструкторским документом* заводские номера *цифрами* (далее – объект испытаний). Место проведения испытаний – *указать место проведения испытаний*.

1. Комиссией установлено

1.1. Программа приемочных испытаний выполнена полностью.

1.2. Состав и комплектность объекта испытаний соответствует технической документации.

1.3. Объект испытаний и его техническая документация выдержали приемочные испытания по Программе и методикам *обозначение документа.*

2 Выводы

2.1 Объект испытаний соответствует всем требованиям, заданным техническим заданием.

2.2 Техническая документация на объект испытаний в техническом пригодна для постановки на производство и последующей реализации продукции.

3. Замечания и рекомендации[2]

3.1..

3.3.[3] ...

Председатель комисси	*И.О.Фамилия*
Члены комиссии	*И.О.Фамилия*
	И.О.Фамилия
	И.О.Фамилия

[2] Текст рекомендаций приведен для случая, когда замечания и несоответствия объекта испытаний отсутствуют.

[3] Другие замечания и рекомендации по усмотрению комиссии по проведению испытаний.