МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования “Витебский государственный технологический университет”

ОРГАНИЗАЦИЯ И ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

Методические указания и задания к лабораторным работам для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем»

ВИТЕБСК
2012

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования “Витебский государственный технологический университет”

РЕКОМЕНДОВАНО:
Зам. председателя редакционно-издательского Совета УО ВГТУ
_____ В. В. Пятов
«____»_____ 20___ г.

УТВЕРЖДАЮ:
Первый проректор
УО ВГТУ
_____С. И. Малашенков
«____»_____ 20___ г.

ОРГАНИЗАЦИЯ И ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

Методические указания и задания к лабораторным работам для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем»

ВИТЕБСК
2011

УДК 004 (075.8)

ОРГАНИЗАЦИЯ И ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ: методические указания и задания к лабораторным работам для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем»

Витебск: Министерство образования Республики Беларусь, УО «ВГТУ», 2011

Составитель**:** доц. Казаков В. Е.

В методических указаниях приведены методики проектирования баз данных, реализации баз данных в средах MS Access и SQL Server, извлечения и обработки данных, разработки хранимых процедур и триггеров. В методические указания также включены индивидуальные задания по всем разделам курса «Организация и проектирование баз данных».

Методические указания предназначены для использования слушателями специальности 1-40 01 73 «Программное обеспечение информационных систем» ФПК и ПК при выполнении лабораторных работ по курсу «Организация и проектирование баз данных».

Одобрено кафедрой информатики УО “ВГТУ”
2 сентября 2011 г., протокол № 1.

Рецензент: Шарстнёв В.Л.
Редактор: Дягилев А.С.

Рекомендовано к опубликованию редакционно-издательским советом УО «ВГТУ» “___”_____ г., протокол № _____

Ответственный за выпуск: Соколов И. В.

Учреждение образования
«Витебский государственный технологический университет»

Подписано к печати _____ Формат _____ Уч.-изд. лист _____
Печать ризографическая. Тираж ____ экз. Заказ №_____

Отпечатано на ризографе учреждения образования «Витебский государственный технологический университет». Лицензия № 02330/0494384 от 16.03.2009
210035, Витебск, Московский пр-т, 72.

Содержание

Введение

Методические указания составлены в соответствии с рабочей программой курса «Организация и проектирование баз данных» и предназначены для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем» ФПК и ПК.

Основное назначение данного курса – систематическое введение в идеи и методы, используемые при проектировании реляционных баз данных, а также при реализации и использовании их средствами современных систем управления базами данных (СУБД). Это очень важная тема, без основательного знакомства с которой невозможно быть не только квалифицированным программистом, но даже и грамотным пользователем информационных технологий.

Использование данных методических указаний в контексте изучения курса «Организация и проектирование баз данных» будет способствовать решению таких задач, как: формирование навыков проектирования и реализации баз данных средствами современных СУБД; формирование навыков управления базами данных средствами языка SQL; формирование навыков прикладного использования баз данных.

Материал, излагаемый в данных методических указаниях, может в определённой степени быть использован при работе с любой современной СУБД.

Проектирование базы данных

Проектирование баз данных (БД) – процесс решения определённого круга задач, связанных с созданием баз данных:

- Обеспечение хранения в БД всей необходимой информации.
- Обеспечение возможности получения данных по всем необходимым запросам.
- Сокращение избыточности и дублирования данных.
- Обеспечение целостности данных (правильности их содержания): исключение противоречий в содержании данных, исключение их потери и т. д.

Предметная область – выделенный для реализации в виде модели данных фрагмент окружающего мира, группа объектов или явлений, между которыми существуют устойчивые во времени соотношения.

Инфологическая модель предметной области – формализованная с помощью различных способов (например, ER-диаграмм) модель. Инфологическая модель строится без ориентации на какую-либо конкретную СУБД. Основу модели составляет множество объектов (сущностей), сгруппированных в наборы сущностей, а также совокупность связей между этими наборами. Также в инфологической модели необходимо учитывать информационные потребности пользователей (описание основных запросов к будущей базе данных), алгоритмические зависимости между данными, требования к допустимым значениям данных и к связям между ними.

Даталогическая модель – отображение инфологической модели на модель данных, используемую в конкретной СУБД, например, на реляционную модель данных. Для реляционных СУБД даталогическая модель – набор таблиц, обычно с указанием ключевых полей, связей между таблицами. Даталогическое проектирование представляет собой построение таблиц по определённым формализованным правилам, а также нормализацию этих таблиц.

Основные понятия реляционной модели данных

Основой для терминологии и методологии, применяющейся в реляционной модели данных, является теория множеств. Базовым понятием является понятие **отношения** (англ. relation – отношение).

N-арным отношением, или отношением степени N, называют подмножество декартового произведения множеств, не обязательно различных. Исходные множества D1,D2,...,Dn называют в модели доменами.

Отношение в терминах баз данных может быть представлено в виде **таблицы**. **Домен** представляется множеством значений определённого типа. **Столбцы** таблицы (**поля**, **атрибуты**) соответствуют вхождениям доменов в от-

ношение. Строки таблицы (**записи, кортежи**) – наборам из N значений, взятых из исходных доменов. Число записей N называют **мощностью** отношения.

Первичный ключ. Одно или несколько полей, комбинация значений которых является уникальной для каждой записи в таблице. Первичный ключ не допускает наличия пустых значений.

Таблица обладает определёнными свойствами:

1. Отсутствие записей-дубликатов. Из этого свойства вытекает наличие у каждой таблицы первичного ключа. Для каждой таблицы, по крайней мере, полный набор её полей является первичным ключом.
2. Отсутствие упорядоченности записей.
3. Отсутствие упорядоченности полей. Для ссылки на значение поля всегда используется имя поля.
4. Атомарность значений полей, т. е. среди значений ячейки не могут содержаться данные сложных типов (таблицы, массивы).

Внешний ключ таблицы представляет собой одно или несколько её полей, значения которых должны совпадать со значениями внешнего ключа другой таблицы.

Связь между таблицами – ассоциация между записями разных таблиц, установленная между двумя полями двух таблиц. Для создания связей, обычно используют дублирование ключей.

Существуют связи **«один-к-одному»** – записи в одной таблице соответствует одна запись из другой, **«один-ко-многим»** записи в одной таблице соответствуют несколько записей из другой, **«многие-ко-многим»** нескольким записям в одной таблице соответствуют несколько записей из другой.

Нормализация

Функциональная зависимость – зависимость, имеющая место между полями X и Y некоторой таблицы, при условии, что каждому значению поля X соответствует ровно одно значение поля Y .

Отметим, что X и Y могут представлять собой не только единичные поля, но и группы, составленные из нескольких полей одной таблицы.

Избыточная функциональная зависимость – зависимость, заключающая в себе такую информацию, которая может быть получена на основе других зависимостей, имеющихся в базе данных.

Нормализация – обратимый пошаговый процесс замены данной совокупности таблиц другой схемой с устранением избыточных функциональных зависимостей.

Условие обратимости требует, чтобы декомпозиция сохраняла эквивалентность схем при замене одной схемы на другую, т. е. в результирующих таблицах:

- не должны появляться ранее отсутствовавшие поля;
- на таблицах новой схемы должно выполняться исходное множество функциональных зависимостей.

Отношение (таблица) находится в первой нормальной форме (1НФ), если каждая запись содержит точно одно значение для каждого из своих полей.

Реляционные таблицы, отвечающие свойствам, приведённым в предыдущем разделе, уже находятся в 1НФ. Строго говоря: таблица не находящаяся в 1НФ не является отношением и не может использоваться в реляционной модели данных.

Отношение (таблица) находится во 2НФ, если оно находится в 1НФ и если каждый неключевой атрибут функционально полно зависит от ключа. Или другими словами: в 2NF нет неключевых атрибутов, зависящих от части составного ключа.

Рассмотрим следующую предметную область:

- имеются сведения о поставках: номер поставщика, товар, цена;
- поставщик может поставлять различные товары, а один и тот же товар может поставляться разными поставщиками;
- пусть все поставщики поставляют товар по одной и той же цене.

Таблица, представляющая данную предметную область, представлена на рисунке 1.

Составной первичный ключ

n_поставщика	товар	цена
1	Вода	1500
2	Сок	2000
1	Сок	2000
2	Вода	1500

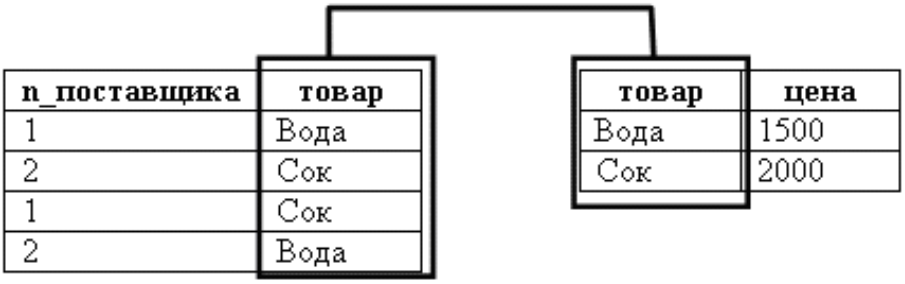
Рисунок 1 – Таблица «Поставщики»

Пусть все поставщики поставляют товар по одной и той же цене. Тогда имеются следующие функциональные зависимости:

- **N_поставщика, товар -> цена;**
- **товар -> цена.**

Неполная функциональная зависимость поля "цена" от ключа приводит к следующей аномалии: при изменении цены товара необходим полный просмотр таблицы для того, чтобы изменить все записи о его поставщиках.

Декомпозиция исходной таблицы, представленная на рисунке 2, позволяет получить две таблицы, находящиеся во 2НФ.



n_поставщика	товар
1	Вода
2	Сок
1	Сок
2	Вода

товар	цена
Вода	1500
Сок	2000

Рисунок 2 – Декомпозиция таблицы «Поставщики»

Отношение (таблица) находится в 3НФ, если оно находится во 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Рассмотрим следующую предметную область:

- имеются сведения о хранении продуктов нефтепереработки: фирма-поставщик, номер слада, объём склада;
- фирма поставляет товар только на один склад;
- на склад могут поставлять товар несколько фирм.

Таблица, представляющая данную предметную область, будет иметь вид, представленный на рисунке 3.

В данной таблице имеются следующие функциональные зависимости:

- **фирма -> склад;**
- **склад -> объем.**

Однако обратное соответствие отсутствует, т. е. «объём» не зависит функционально от «склад» и «склад» не зависит функционально от «фирма». Тогда говорят, что поле «объём» транзитивно зависит от поля «фирма».

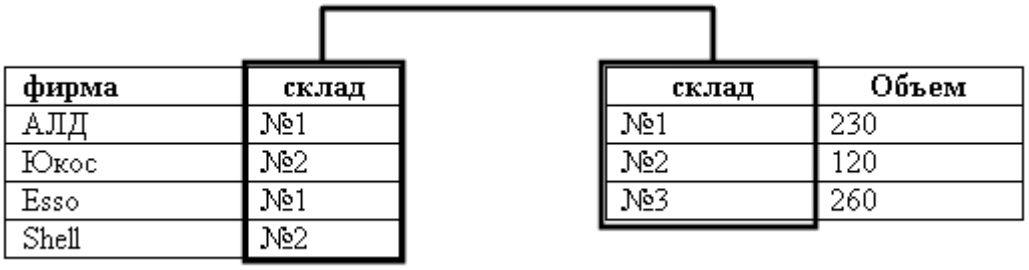
фирма	склад	объем
АЛД	№1	230
ЮКОС	№2	120
Esso	№1	230
Shell	№2	120
ТНК	№3	260

Рисунок 3 – Таблица «Хранение»

При этом возникают следующие аномалии:

- если в данный момент ни одна фирма не получает товар со склада, то в базу данных нельзя ввести данные о его объеме (т. к. не определен ключевой атрибут);
- если объем склада изменяется, необходим просмотр всего отношения и изменение записей для всех фирм, связанных с данным складом.

Для устранения этих аномалий необходимо провести декомпозицию исходного отношения, как показано на рисунке 4.



фирма	склад
АЛД	№1
Юкос	№2
Esso	№1
Shell	№2

склад	Объем
№1	230
№2	120
№3	260

Рисунок 4 – Декомпозиция таблицы «Хранение»

Отношение (таблица) находится в 4NF, если оно находится в 3NF и в нем отсутствуют многозначные зависимости, не являющиеся функциональными зависимостями.

Рассмотрим следующую предметную область:

- имеется список преподавателей, содержащий ФИО, читаемый курс, учебное пособие для изучения курса;
- преподаватель может являться автором нескольких пособий;
- учебное пособие может быть подготовлено несколькими преподавателями;
- преподаватель может вести несколько курсов;
- курс может читаться несколькими преподавателями.

Таблица, представляющая данную предметную область, будет иметь вид, представленный на рисунке 5.

В данной таблице имеются следующие функциональные зависимости:

- зависимость множества значений поля «Курс» от множества значений поля «ФИО»;
- зависимость множества значений поля «Учебное_пособие» от множества значений поля «ФИО».

Составной первичный ключ

Учебное_пособие	Курс	ФИО
Теория упругости	ТУпр	Николаев В.А.
Теория упругости	ТК	Николаев В.А.
Теория колебаний	ТУпр	Николаев В.А.
Теория колебаний	ТК	Николаев В.А.
Теория удара	ТУд	Кузнецов П.А.
Теоретическая механика	ТУд	Кузнецов П.А.

Рисунок 5 – Таблица «Преподаватели»

При этом возникает аномалия: добавление информации о том, что профессор Кузнецов П.А. будет читать лекции по курсу ТУпр, приводит к необходимости добавить две записи (по одной для каждого написанного им учебника) вместо одной.

Для устранения многозначных зависимостей необходимо провести декомпозицию исходного отношения, как показано на рисунке 6.

ФИО	Курс
Николаев В.А.	ТУпр
Николаев В.А.	ТК
Кузнецов П.А.	ТУд

ИМЯ	УЧЕБНОЕ_ПОСОБИЕ
Николаев В.А.	Теория упругости
Николаев В.А.	Теория колебаний
Кузнецов П.А.	Теория удара
Кузнецов П.А.	Теоретическая механика

Рисунок 6 – Декомпозиция таблицы «Преподаватели»

Пример разработки даталогической модели

Рассмотрим в качестве предметной области некоторую торговую организацию, выполняющую некоторые заказы. Модель предметной области опишется следующим неформальным текстом:

- Сотрудники организации обеспечивают выполнение некоторых заказов.
- Один заказ может курироваться одним или несколькими сотрудниками.
- Сотрудник может курировать один или несколько заказов.

- Организация состоит из нескольких отделов.
- В каждом отделе работает один или более сотрудников.
- Каждый сотрудник может работать только в одном отделе.
- Организация имеет список заказчиков.
- Каждый заказчик один, или более раз обращался в организацию с заказом.
- Заказ может поступить только от одного заказчика.

Из описания можно сделать вывод о наличии в предметной области четырёх наборов сущностей: «отделы», «сотрудники», «заказы» и «заказчики».

Состав атрибутов каждого набора сущностей может быть уточнён на любом этапе проектирования базы данных.

Инфологическая модель организации представлена на рисунке 7.

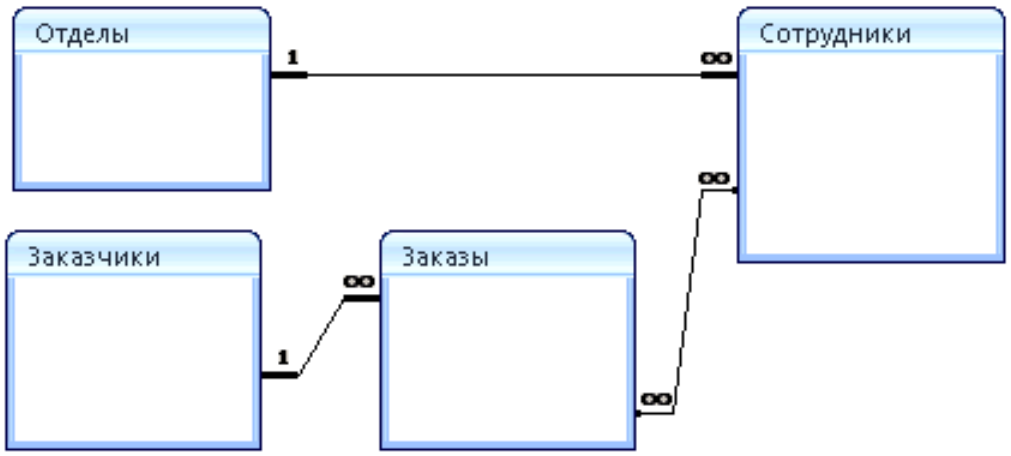


Рисунок 7 – Инфологическая модель предметной области «Организация»

Типы связей между наборами сущностей установлены исходя из описания предметной области:

Отделы – Сотрудники

- В каждом отделе работает один или более сотрудников.
- Каждый сотрудник может работать только в одном отделе.

Заказчики – Заказы

- Каждый заказчик один или более раз обращался в организацию с заказом.
- Заказ может поступить только от одного заказчика.

Заказы – Сотрудники

- Один заказ может курироваться одним или несколькими сотрудниками.

- Сотрудник может курировать один или несколько заказов.

Даталогическую модель мы связываем с реляционной моделью данных.

Каждый набор сущностей будет представлен таблицей.

Определим первичный ключ для каждой таблицы. Если среди атрибутов набора не найдётся атрибута, отвечающего требованиям первичного ключа, то в таблицу вводиться дополнительный атрибут:

Сотрудники: Табельный_номер.

Отделы: введём атрибут Номер _отдела.

Заказы: Номер_контракта.

Заказчики: введём атрибут Номер_заказчика.

Связь «один-ко-многим» (1:∞) с помощью реляционной модели реализовать с помощью копирования первичного ключа из основной таблицы достаточно просто (рисунок 8).

Проблему представляет реализация связи «многие-ко-многим» между наборами «Сотрудники» и «Заказы».

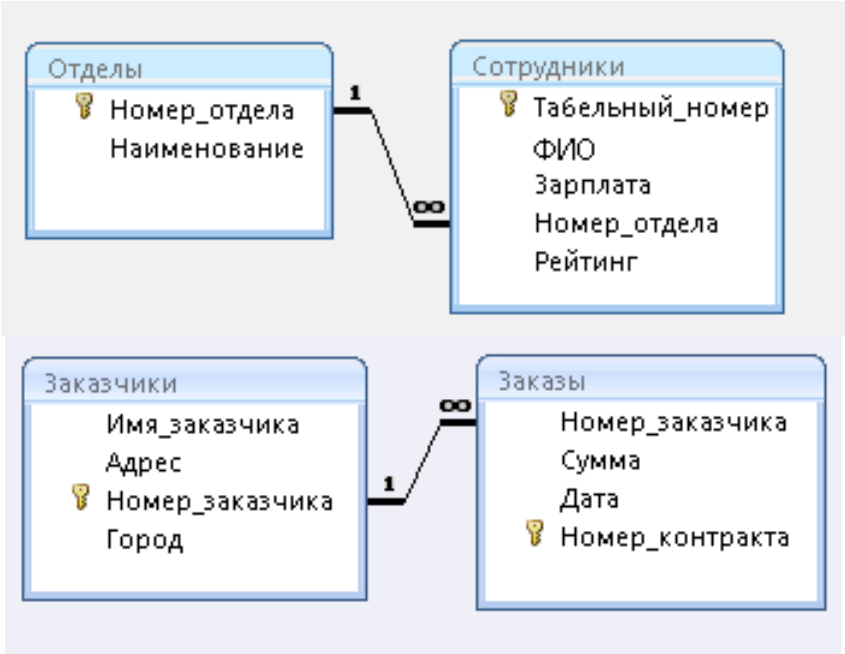


Рисунок 8 – Реализация связей «один-ко-многим» в даталогической модели предметной области «Организация»

Если установить связь между таблицами «Сотрудники» и «Заказы» напрямую, то, очевидно, будет необходимо в поле внешнего ключа для обеих таблиц занести последовательность значений (например, список всех номеров контрактов, которые курирует сотрудник, или список табельных номеров сотрудников, курирующих контракт). Такая схема, представленная на рисунке 9, не соответствует первой нормальной форме.

Сотрудники

Табельный_номер	ФИО	Номер_контракта
1	Иванов С. П.	201, 204
2	Сидоров А. В.	203
5	Петров А. П.	201, 203, 204

Заказы

Номер_контракта	Сумма	Табельный_номер
201	1000000	1, 5
203	1200000	2, 5
204	650000	1, 5

Рисунок 9 – Схема связи таблиц «Сотрудники» и «Заказы», не соответствующая первой нормальной форме

Если попытаться объединить таблицы «Сотрудники» и «Заказы» в одну таблицу, то в первичный ключ этой таблицы будет состоять из двух полей (Табельный_номер, Номер_контракта). Вследствие этого появятся поля с неполной функциональной зависимостью от первичного ключа (рисунок 10). Например, поле «Сумма» функционально зависит и от первичного ключа («Табельный_номер», «Номер_контракта»), и от части первичного ключа («Табельный_номер»).

Сотрудники_Заказы

Составной первичный ключ

Табельный_номер	Номер_контракта	ФИО	Сумма
1	201	Иванов С. П.	1000000
1	204	Иванов С. П.	650000
2	203	Сидоров А. В.	1200000
5	201	Петров А. П.	1000000
5	203	Петров А. П.	1200000
5	204	Петров А. П.	650000

Рисунок 10 – Схема декомпозиции таблицы «Сотрудники»

Очевидной является необходимость разделения таблицы «Сотрудники» на две части: сведения непосредственно о сотрудниках и сведения о выполняемых ими контрактах, как показано на рисунке 11.

Сотрудники

ФИО	Табельный_номер
Иванов С. П.	1
Сидоров А. В.	2
Петров А. П.	5

Курируемые_заказы

Номер_контракта	Табельный_номер
201	1
201	5
203	2
203	5
204	1
204	5

Рисунок 11 – Схема декомпозиции таблицы «Сотрудники»

Аналогичным образом можно провести декомпозицию таблицы «Заказы» (рисунок 12).

Заказы

Сумма	Номер_контракта
1000000	201
1200000	203
650000	204

Исполнение_заказа

Табельный_номер	Номер_контракта
1	201
5	201
2	203
5	203
1	204
5	204

Рисунок 12 – Схема декомпозиции таблицы «Заказы»

Очевидно, что таблица «Курируемые_заказы» полностью аналогична таблице «Исполнение_заказа». Первичный ключ в обеих таблицах состоит из двух полей: «Табельный_номер» и «Номер_контракта».

Окончательная даталогическая модель рассматриваемой предметной области представлена на рисунке 13.

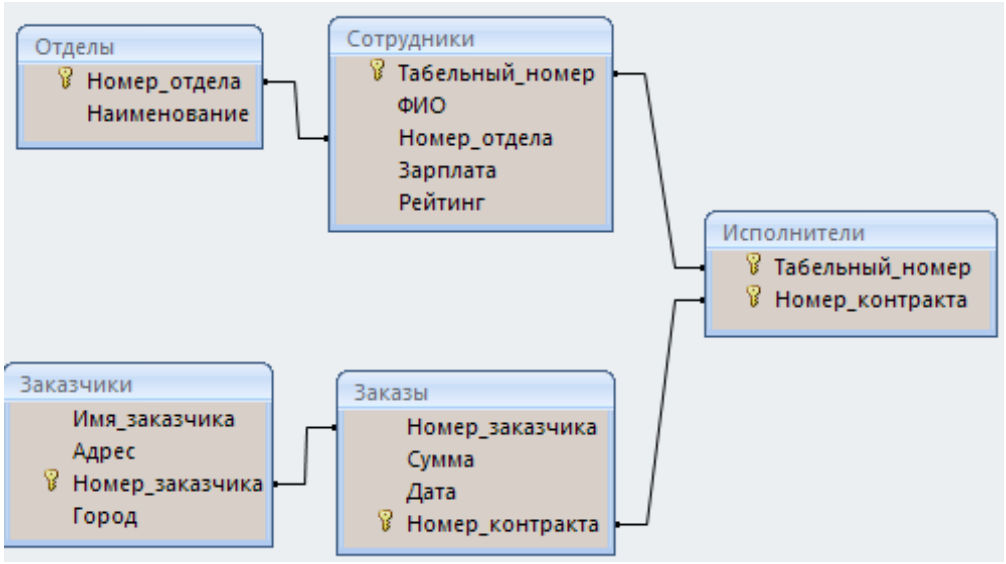


Рисунок 13 – Даталогическая модель предметной области «Организация»

Задания для самостоятельной работы

Вариант 1. База данных ПОСТАВКА ТОВАРОВ

Необходимо спроектировать базу данных, содержащую информацию о поставках предприятием различных товаров. Предприятие обслуживает определённый круг клиентов, которые периодически оформляют заказы.

При проектировании БД необходимо учитывать следующее:

- база данных должна содержать сведения: о товаре известны: наименование, цена, себестоимость производства, единица измерения количества и количество на складе; данные о заказчике: ФИО, телефон, адрес; заказ содержит сведения о дате поставки, о дате принятия заказа и количестве каждого из требуемых товаров;
- клиент может иметь несколько заказов, каждый заказ закреплён за одним клиентом;
- товар может иметь несколько заказов на поставку, заказ может состоять из нескольких товаров.

Проектируемая база данных используется для получения следующих сведений:

- информации о заказах товаров;
- информации о ежемесячных отгрузках и выручке;

- информации об активности заказчиков;
- данных о спросе на товары (и его сезонности).

Вариант 2. База данных РОЗНИЧНАЯ ТОРГОВЛЯ

Магазин “Компьютерный мир” продает персональные компьютеры, средства связи и периферийное оборудование. Товар закупается у нескольких организаций-поставщиков.

При проектировании БД необходимо учитывать следующее:

- база данных должна содержать сведения: о поставочной цене товара от каждого поставщика, о количестве товара на складе, описание технических характеристик и продажной цене каждого вида товара, реквизитах поставщиков (телефон, название организации, адрес), о дате продажи;
- товар определённого вида может поставляться разными поставщиками по различной цене, поставщик может поставлять несколько видов товаров;
- каждая продажа обязательно связана с одним видом товара.

Проектируемая база данных используется для получения следующих сведений:

- региональной информации (о показателях поставок из различных городов);
- информации о ежемесячных продажах;
- информации о наличии товаров на складе;
- данных о сезонном изменении спроса на товары.

Вариант 3. База данных БАНКОВСКИЕ ВКЛАДЫ

Необходимо создать базу данных о счетах и операциях с ними клиентов банка.

При проектировании БД необходимо учитывать следующее:

- база данных должна содержать сведения: ФИО, адрес, телефон клиента; номер счёта, дата открытия и остаток средств на нём; для 1-го типа операции: дата, сумма; для 2-го: дата, сумма и номер счета, с которым проводилась операция;
- клиент банка может иметь несколько счетов, счет открывается только на одного клиента;
- со счётом имеется возможность проводить два типа операций: 1) снять/пополнить средства на счёте, 2) перевод средств с/на другой счёт.

Проектируемая база данных используется для получения следующих сведений:

- информации о текущем состоянии счетов;
- сведений о состоянии счетов на определённый момент времени;
- истории операций с определёнными счетами.

Вариант 4. База данных ТОРГОВЛЯ

Отделы торгового дома ежедневно продают различные виды товаров и ведут учет сведений о продажах в виде чеков.

При проектировании БД необходимо учитывать следующее:

- база данных должна содержать сведения: о названии отдела, его начальнике и контактном телефоне; о закупочной и продажной цене товара, его названии и габаритах; о дате продажи и купленном количестве каждого вида товара;
- в отделе продается несколько видов товаров, каждый вид товара продаётся в одном определённом отделе;
- в чек включаются сведения о продаже нескольких товаров, каждый вид товара может быть включён в чек только один раз.

Проектируемая база данных используется для получения следующих сведений:

- информации для формирования ежемесячных отчётов о продажах в отделах;
- информации о спросе на товары различных видов и его сезонности.

Вариант 5. База данных АВИАКОМПАНИЯ

Авиакомпания осуществляет разовые чартерные рейсы. Компания имеет постоянный штат сотрудников, состоящий из пилотов, бортмехаников, стюардесс и стрелков-радистов, а также постоянный парк самолётов.

При проектировании БД необходимо учитывать следующее:

- база данных должна содержать сведения: о сотрудниках: ФИО, стаж, специальность, место, занимаемое в каждом экипаже (например: пилот может в экипаже быть и командиром, первым пилотом или стюардессой); о самолётах: тип самолёта, бортовой номер, количество мест, дата последнего техосмотра; о рейсах: дата, пункт назначения, время в полёте, количество занятых мет;
- один рейс обслуживается определённым экипажем и определённым самолётом, самолёт и сотрудники могут быть задействованы в различных рейсах;

Проектируемая база данных используется для получения следующих сведений:

- информации о количестве часов налёта у каждого из сотрудников компании и лётной техники;
- информации о текущем местонахождении самолётов и экипажей;
- данных о пассажирообороте и его сезонных колебаниях;
- информации о техническом состоянии лётной техники.

СУБД MS Access

MS Access является файл-серверной системой управления базами данных (СУБД), используемой в основном для обучения и решения небольших локальных задач. Поставляется MS Access в составе пакета MS Office.

Интерфейс СУБД MS Access

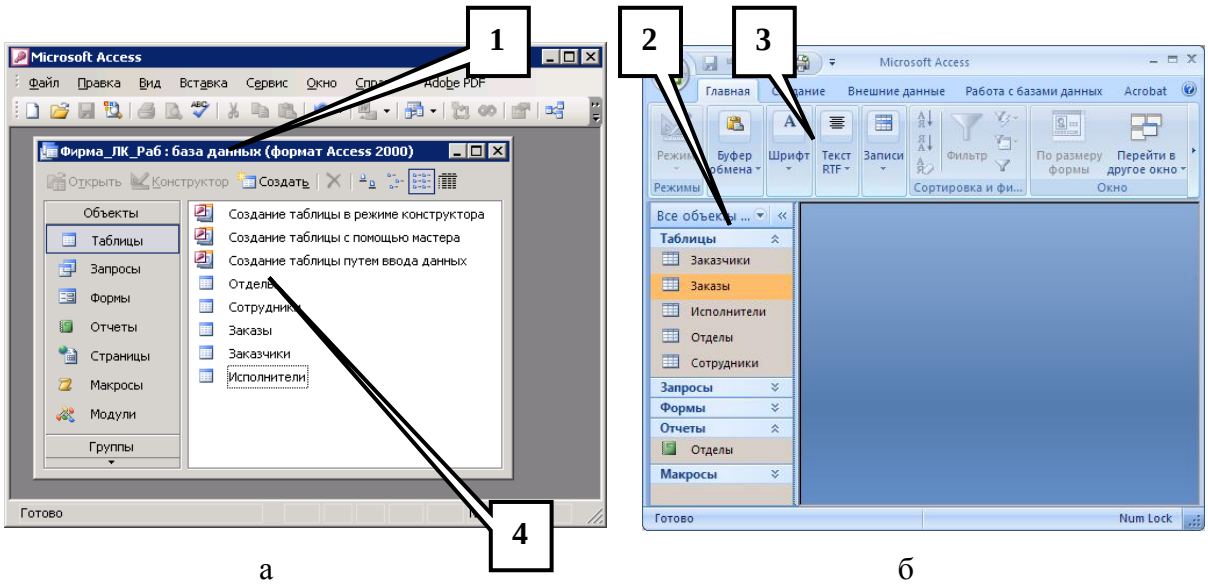
СУБД MS Access имеет несколько существенных отличий от обычного приложения, работающего с электронными документами:

- в одном экземпляре приложения можно открыть только один файл базы данных;
- база данных при создании сразу сохраняется в файловой системе.

Несмотря на то, что, начиная MS Office 2007, интерфейс приложения изменился, функционально элементы окна базы данных сохранили преемственность.

Основное окно в старой версии (1 рисунке 14) представляет собой двухуровневый список с вкладками, на которых размещаются компоненты файла базы данных (таблицы, запросы, формы и т. п.). В новой версии для этой же цели используется разворачивающийся список (2 на рисунке 14).

Для создания новых компонентов базы данных в старой версии на соответствующих вкладках имелись ярлыки, реализующие тот или иной способ создания компонента (4 на рисунке 14). В новой версии все команды создания компонентов собраны на вкладке «Создание» ленты (3 на рисунке 14).



а б

Рисунок 14 – Интерфейс MS Access:
а) MS Access 2003 (старая версия); б) MS Access 2007 (новая версия)

Каждый из компонентов файла базы данных можно открыть в нескольких режимах. Основными режимами являются:

- режим конструирования, в котором, с помощью соответствующего средства, можно изменить структуру компонента, настроить логику его работы или внешний вид, т.е. выполнить работу программиста, проектирующего приложение для работы с базой данных;

- пользовательский режим, который реализует работу конечного пользователя (ввод и редактирование данных, запуск и работа с формами и т.д.) с готовым программным продуктом.

Для запуска компонента в пользовательском режиме достаточно выбрать его в списке основного окна. Для переключения режимов отображения компонента в старой версии используется меню «Вид», в новой версии кнопка переключения режимов размещается на вкладке ленты «Главная».

В среде MS Access каждый из компонентов открывается в отдельном окне и сохраняется отдельно от других открытых для работы компонентов.

Коротко рассмотрим компоненты, из которых состоит файл базы данных MS Access.

Основными компонентами являются таблицы. Они, вместе с совокупностью связей между ними, и составляют базу данных. Конструкторы таблиц и связей будут рассмотрены в следующем разделе.

Из всех типов объектов только таблицы предназначены для хранения информации. Остальные используются для просмотра, редактирования, обработки и управления данными - иначе говоря, для обеспечения эффективного доступа к информации.

Запрос – это команда на языке Jet-SQL, которая сохраняется в файле базы данных в текстовом формате.

Формы используются для ввода и редактирования записей в таблицах. В сущности, форма представляет собой оконное приложение для прикладного использования данных, сохранённых в таблицах.

Отчеты используются для отображения информации, содержащейся в таблицах, в отформатированном виде, который легко читается как на экране компьютера, так и на бумаге.

Чтобы предоставить доступ к информации, хранящейся в базе данных, пользователям Интернета, можно создать страницы, называемые страницами доступа к данным. Работа с данными на странице доступа в сети осуществляется примерно так же, как в Access – пользователи могут просматривать таблицы, выполнять запросы и заполнять поля форм и т. д.

Макросы представляют собой небольшие программы, с помощью которых обеспечивается реакция Access на события (например: открытие формы, щелчок кнопки или обновление записи).

Модули представляют собой программы, на языке программирования высокого уровня Visual Basic for Applications (VBA).

Создание базы данных

Приступать к реализации базы данных можно только после разработки даталогической модели.

Можно предложить следующую последовательность действий:

- создание всех таблиц, составляющих базу данных;

- настройка связей между таблицами;
- заполнение таблиц данными.

Для создания таблиц используется специальный конструктор (рисунок 15). Верхняя часть представляет собой список полей таблицы. Колонка «Имя поля» (1 на рисунке 15) используется для задания, имени поля, в колонке «Тип данных» (2 на рисунке 15) можно выбрать один из стандартных типов данных. Диспетчер свойств поля (3 на рисунке 15) позволяет задать свойства выбранного в списке полей поля. В области подсказки (4 на рисунке 15) отображается короткая справка по выделенному в диспетчере свойству.

После того, как созданы все таблицы, можно переходить к установлению связей между ними. Для этого используется специальный конструктор, который в старой версии можно вызвать из меню «Сервис», команда «Схема данных». В новой версии кнопка вызова этого конструктора находится на вкладке ленты «Работа с данными».

На рабочей области конструктора размещаются отображения таблиц из базы данных и связи, соединяющие внешние ключи связанных таблиц (рисунок 16).

Для выполнения операций со строками в списке полей (вставка, удаление, установка ключевого поля) можно использовать контекстное меню.

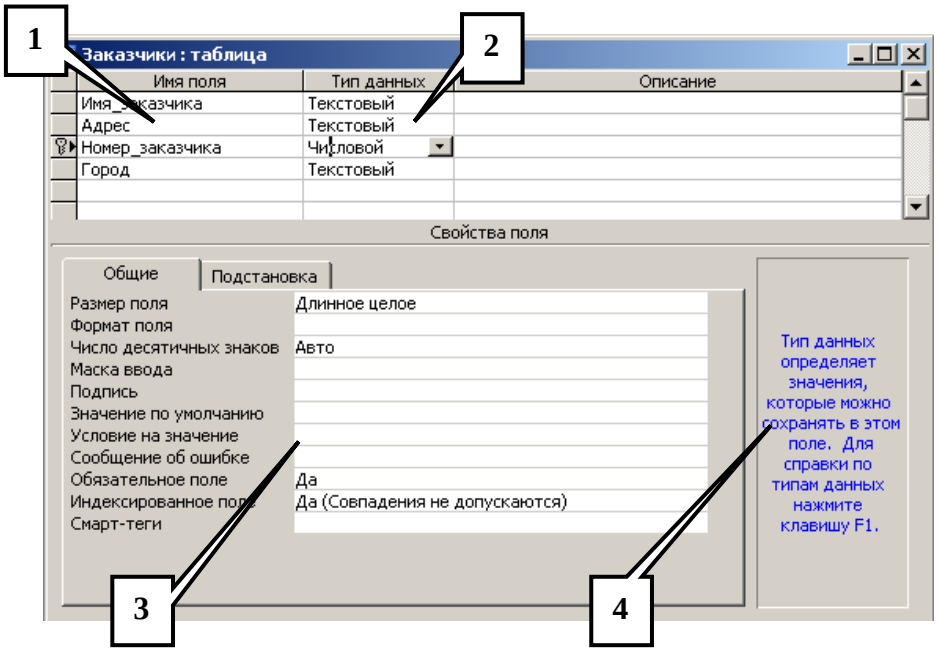


Рисунок 15 – Конструктор таблиц MS Access

Для добавления отображения таблицы можно воспользоваться контекстным меню рабочей области конструктора, для установления новой связи нужно перетащить курсор с внешнего ключа одной таблицы на внешний ключ другой таблицы. При этом появляется диалоговое окно «Изменение связей», позволяющее установить обеспечение целостности, автоматически поддержи-

вающее соответствие в значениях во внешних ключах связанных таблиц, или с помощью списка внешних ключей связываемых таблиц.

Для удаления и редактирования связей и отображений можно воспользоваться их контекстным меню. Также из контекстного меню представления таблицы можно вызвать конструктор самой таблицы.

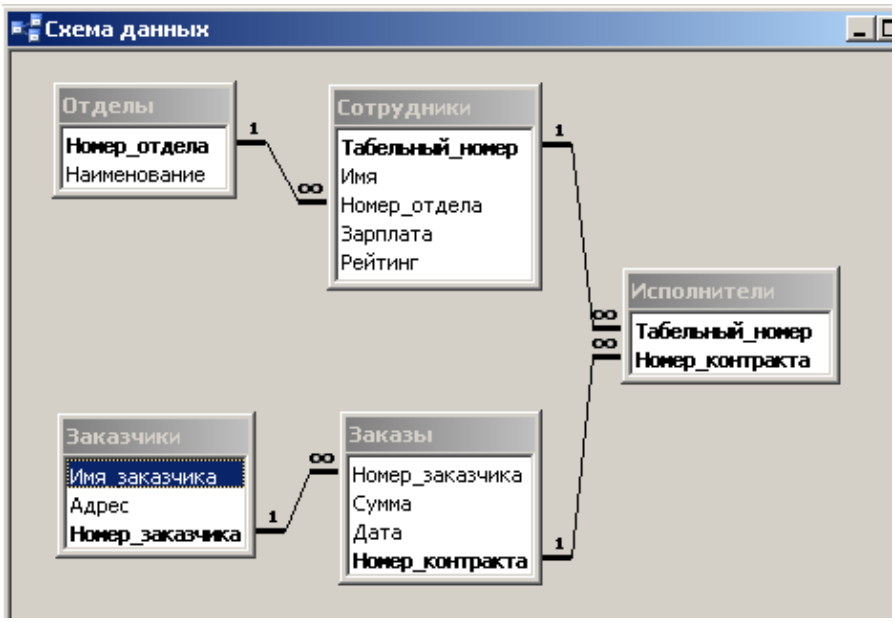


Рисунок 16 – Конструктор «Схема данных» MS Access

Нужно обратить внимание на различие между отображением таблицы и непосредственно таблицей. При выполнении любых операций с отображениями таблиц в конструкторе «Схема данных» (удаление, добавление, создание копий) состав таблиц базы данных не изменяется.

Изменения, внесённые в схему данных, также требуют отдельного сохранения.

После того, как полностью создана структура базы данных, можно приступать к заполнению её данными. Для заполнения используется редактор, который запускается при открытии таблицы в пользовательском режиме. Редактор таблиц MS Access позволяет редактировать таблицу, как и редактор табличного процессора.

Однако следует учитывать некоторые особенности, отличающие его от обычных табличных редакторов:

- При удалении столбца происходит удаление поля из таблицы.
- Новая запись в таблицу добавляется при начале редактирования пустой строки, расположенной внизу таблицы.
- При попытке перейти к редактированию другой записи осуществляется контроль дублирующихся значений в первичном ключе, уникальных полях, несоответствующих значений во внешнем ключе. Пока ре-

дактируемая запись содержит некорректные данные, переход к редактированию другой строки будет заблокирован.

- Проверка соответствия введённого значения типу поля осуществляется при попытке перейти к редактированию другой ячейки. Пока редактируемая ячейка содержит значение, не соответствующее типу данных поля, переход к редактированию другой ячейки будет заблокирован.

При заполнении базы данных с законченной структурой следует обратить внимание на последовательность заполнения таблиц. Например, если начать заполнение базы данных с таблицы «Сотрудники», необходимо будет заполнить поле «Номер_отдела», являющееся внешним ключом, а так как таблица «Отделы» не содержит ни одной записи, с которой можно было бы установить связь, то вставить запись в таблицу «Сотрудники» будет невозможно.

Для создания базы данных «Организация» по даталогической модели, представленной на рисунке 13, можно предложить следующую последовательность заполнения таблиц:

1. Заполнение основных таблиц: «Отделы», «Заказчики».
2. Заполнение подчинённых таблиц: «Заказы», «Сотрудники».
3. Заполнение связующей таблицы «Исполнители».

Такой метод заполнения позволяет использовать одну дополнительную возможность пользовательского режима работы с таблицей. MS Access, начиная с 2007 версии, позволяет раскрывать и редактировать связанные записи подчинённой прямо из редактора основной.

Обратите внимание на рисунок 17, на нём представлена открытая в редакторе таблица «Отделы».

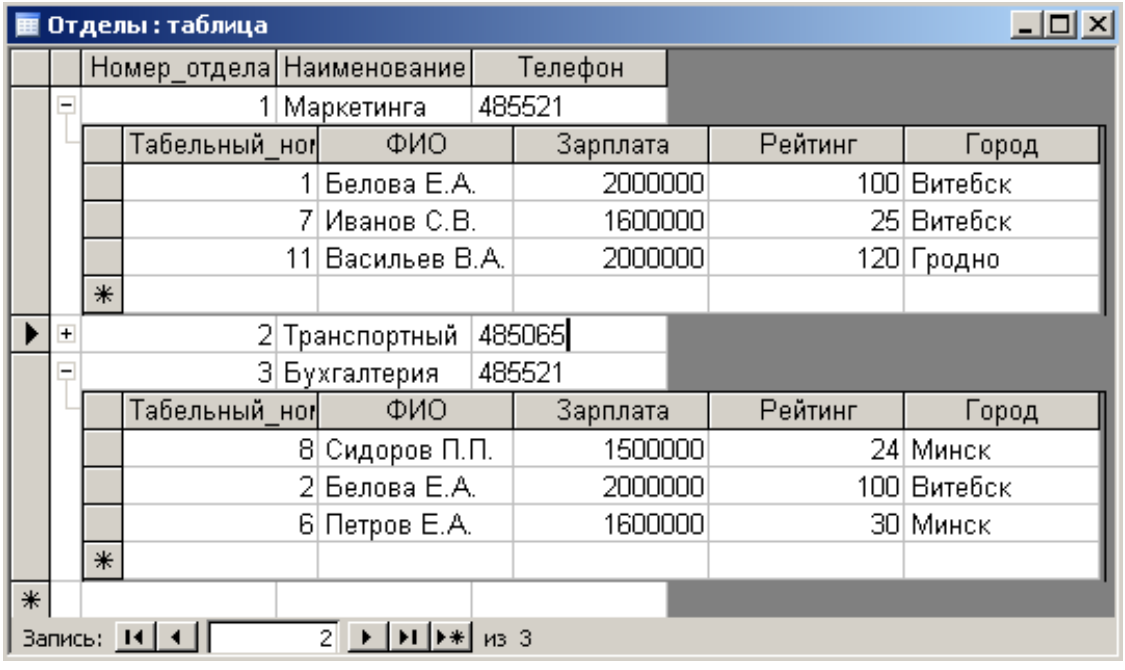


Рисунок 17 – Режим редактирования данных таблицы MS Access

Каждая её запись связана с несколькими записями в таблице «Сотрудники». Редактор позволяет развернуть и отредактировать эти записи прямо в своей рабочей области.

Таким образом: редактируются сразу две связанные таблицы, а также отпадает необходимость контролировать значения во внешних ключах (обратите внимание, что в развёрнутых записях таблицы «Сотрудники» отсутствует поле «Номер отдела», так как оно заполняется автоматически).

Когда база данных заполнена начальным набором данных, можно переходить к составлению запросов для извлечения необходимой информации. Начальный набор данных необходим для отладки запроса, точнее, проверки результатов его работы.

Для создания запросов можно воспользоваться мастером, конструктором или текстовым редактором, написав запрос на языке Jet-SQL.

Конструктор запросов состоит из двух частей: источника выборки (1 на рисунке 18) и списка полей результирующей таблицы (2 на рисунке 18) с настраиваемым фильтром (3 на рисунке 18).

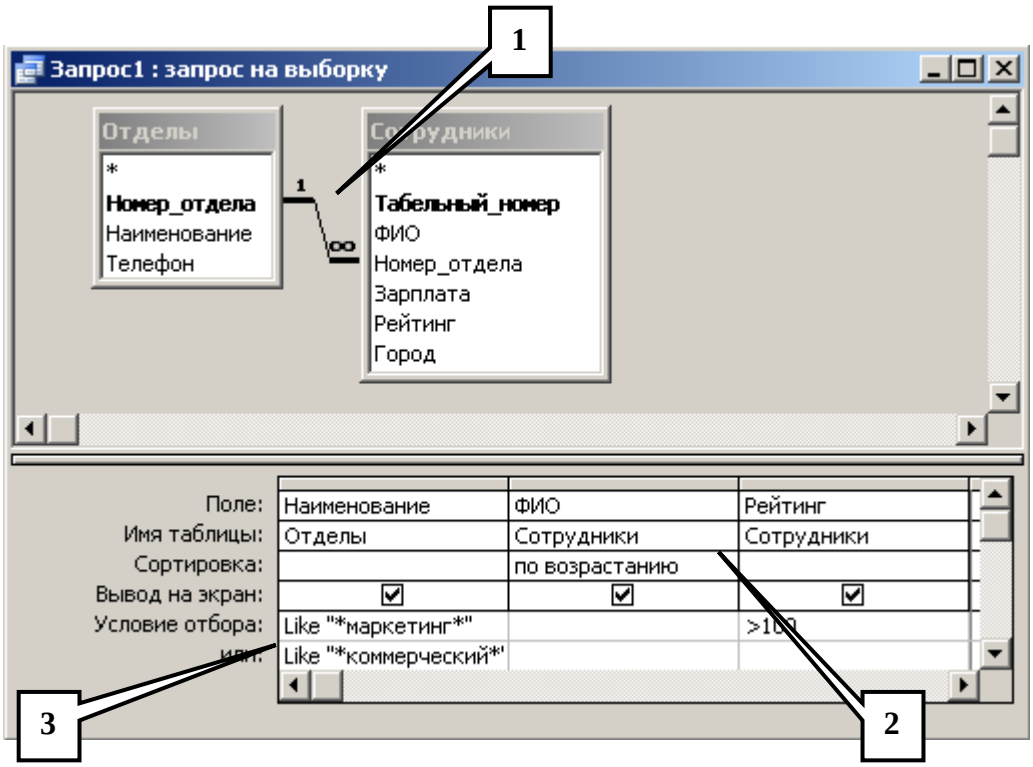


Рисунок 18 – Конструктор запросов MS Access

Источник выборки редактируется так же, как и конструктор схемы данных, однако здесь, помимо таблиц можно использовать и запросы на выборку, уже сохранённые в файле базы данных. Поля фильтра можно заполнять вручную логическими выражениями на языке Jet-SQL.

Конструктор запросов имеет ещё ряд инструментов для создания боле сложных запросов, однако более подробно работу с ними мы рассматривать не

будем. В основном он используется для создания костяка запроса, который затем дорабатывается вручную на языке SQL.

Перейти в текстовый редактор обработки запроса можно с помощью команд переключения режима отображения объекта (меню «Вид» команда «Режим SQL»). Описывать работу с текстовым редактором, надеюсь, не требуется.

Готовый запрос можно запустить с помощью команды «Запуск» из меню «Запрос», запрос на выборку будет запущен также, если переключить его в режим таблицы из меню «Вид».

Рассмотрим некоторые наиболее распространенные ошибки.

Синтаксические ошибки

Выдаётся предупреждающее сообщение «Обнаружены символы за пределами инструкции SQL»: возникает, когда в тексте запроса имеются символы после символа «;». Символ «;» должен завершать текст запроса.

Выдаётся предупреждающее сообщение «Ошибка синтаксиса (пропущен оператор) в выражении …». Указанное в сообщении выражение содержит синтаксическую ошибку.

Выдаётся предупреждающее сообщение «Ошибка синтаксиса в операции JOIN». Кроме наличия в операции JOIN нарушений синтаксиса, ошибка может быть вызвана опечаткой в названии одного из участников объединения.

Выдаётся предупреждающее сообщение «Ссылка на поле … может относиться к полям нескольких таблиц, перечисленных в предложении FROM инструкции SQL». Для устранения ошибки указанное название поля необходимо дополнить названием таблицы (Номер_отдела -> Отделы.Номер_отдела).

Ошибки ядра базы данных

При попытке запустить запрос появляется окно «Введите значение параметра». Данная ситуация возникает при появлении в тексте запроса неизвестного идентификатора. Ядро базы данных воспринимает такие идентификаторы, как параметры запросов, которые должны быть заданы при его запуске. Если при решении задания вы не используете параметрический запрос (данные методические указания не содержат описания таких типов запросов и заданий, требующих их применения), то появление окна «Введите значение параметра» сигнализирует о наличие опечатки в названии поля.

При попытке запустить запрос появляется окно «Попытка выполнить запрос, который не включает указанное выражение … как часть статистической функции или группы». Такая ошибка возникает при нарушении правила применения группировки в команде SELECT: если в запросе используется группировка, или одно из полей результирую-

щей таблицы вычисляется с помощью агрегатной функции, то остальные поля результирующей таблицы должны либо присутствовать в списке GROUP BY, либо вычисляться с помощью агрегатной функции.

Каждый запрос сохраняется отдельным объектом в файле базы данных.

Нужно обратить внимание на то, что в MS Access можно сохранить только синтаксически правильный запрос.

Выборка из базы данных

В MS Access для написания запросов используется диалект языка SQL – Jet-SQL.

Результатом запроса на выборку данных из базы является таблица, а также перечисление и скалярное значение как частные случаи таблицы с одним полем и несколькими записями и одним полем и одной записью соответственно. Запрос на выборку – это описание того, как должна быть создана результирующая таблица.

Выборка информации из базы данных осуществляется с помощью команды SELECT языка SQL.

Синтаксис команды SELECT:

```
SELECT <список_вывода>
     FROM <источник_выборки>
     [ WHERE <условие_WHERE> ]
     [ GROUP BY <список_группировки>]
     [ HAVING <условие_HAVING> ]
     [ ORDER BY <список_сортировки>];
```

<**список_вывода**> состоит из выражений, описывающих содержимое столбцов результирующей таблицы.

<источник_выборки> является конструкцией, описывающей таблицу, из которой будет производиться выборка.

<условие_WHERE> – это логическое выражение, определяющее записи источника выборки, которые попадут в результирующую таблицу.

<список_группировки> – это список выражений, на основе которых производиться группировка записей результирующей таблицы.

<условие_HAVING> – это логическое выражение, определяющее записи сгруппированной таблицы, которые попадут в результирующую таблицу.

<список_сортировки> – это список выражений, на основе которых производиться сортировка записей результирующей таблицы.

Разработку запроса можно осуществлять поэтапно.

1. Определение источника выборки. При изучении задания нужно выяснить, какие таблицы необходимы для его выполнения. На этом же этапе можно прийти к заключению о необходимости сконструировать источник выборки в виде подзапроса или объединения таблиц с помощью отличных от уже существующих в базе данных связей.
2. Составление условия фильтрации полученной на предыдущем этапе таблицы.
3. Группировка записей и формирование выражений для наполнения полей результирующей таблицы.
4. Составление условия фильтрации полученной после группировки таблицы.
5. Задание способа сортировки результирующей таблицы.

На некоторых этапах, возможно, потребуется составление подзапроса, для получения определённого значения, перечисления или таблицы.

Рассмотрим основные приёмы, используемые при составлении запросов.

Фильтрация

Вывести сведения о заказчиках с офисом, расположенном на Московском проспекте в Витебске.

Для выполнения задания необходимо использовать таблицу «Заказчики», выбрав из неё записи, которые в поле «Город» содержат значение “Витебск” и в текстовой строке поля «Адрес» содержится подстрока «пр. Московский».

```
SELECT  Заказчики.Имя_заказчика
FROM Заказчики
WHERE (Заказчики.Адрес Like "*пр. Московский*") AND (Заказчики.Город="Витебск");
```

Результат работы запроса:

Имя_заказчика
Родионов А.П.

Вычисляемое поле результирующей таблицы

Вывести сведения о зарплатах сотрудников 1-го отдела в долларах.

Запрос для вывода требуемых сведений будет выглядеть следующим образом:

```
SELECT ФИО, round(Зарплата/2800,2) as [Зарплата в $]
FROM Сотрудники
WHERE Номер_отдела =1;
```

Обратите внимание на описание второго поля в разделе SELECT:

round(Зарплата/2800,2) as [Зарплата в $]

Значения данного поля формируются при помощи вычисления выражения (**round –** скалярная функция округления вещественного значения до определённого количества знаков после запятой). В данном выражении 2800 – оптимистичный курс доллара, который можно получить с помощью подзапроса из таблицы, содержащей реальный текущий курс доллара.

Следует также обратить внимание на использование квадратных кавычек (**[Зарплата в $]**). В языке SQL в такие кавычки заключаются идентификаторы содержащие недопустимые в обычных идентификаторах символы (например, пробел). Все символы внутри квадратных скобок становятся частью идентификатора.

Результат работы запроса:

ФИО	Зарплата в $
Белова Е.А.	714,29
Иванов С.В.	571,43

Объединение нескольких таблиц в источнике выборки

Вывести сведения о сотрудниках 1-го и 2-го отделов, которые обслуживают заказы клиентов из своего родного города.

Для обеспечения заданной выборки потребуются данные четырёх таблиц: «Заказчики», «Заказы», «Исполнители» и «Сотрудники». Таблицы соединяются с помощью внутреннего объединения с использованием своих внешних ключей. Объединения подобного рода создаются в большинстве СУБД автоматически.

Рассмотрим конструкцию такого объединения (рис. 19). Блок 1 на рисунке 19 – это объединение двух таблиц. Результат объединения также является таблицей, содержащей все поля из исходных объединяемых таблиц. Блок 2 на рисунке 19 – это конструкция объединения таблицы «Заказчики» с объединением, полученным в блоке 2. Аналогичным образом к полученному объединению присоединяется таблица «Сотрудники».

Рисунок 19 – Внутреннее объединение четырёх таблиц

Запрос для вывода требуемых сведений будет выглядеть следующим образом:

SELECT Сотрудники.ФИО, [Сотрудники].[Город]
FROM Сотрудники INNER JOIN (Заказчики INNER JOIN (Заказы INNER JOIN Исполнители ON Заказы.Номер_контракта = Исполнители.Номер_контракта) ON Заказчики.Номер_заказчика = Заказы.Номер_заказчика) ON Сотрудники.Табельный_номер = Исполнители.Табельный_номер
WHERE (Заказчики.Город=[Сотрудники].[Город]) AND (Сотрудники.Номер_отдела=2 Or Сотрудники.Номер_отдела=1);

Результат работы запроса:

ФИО	Город
Белова Е.А.	Витебск
Белова Е.А.	Витебск
Иванов С.В.	Витебск

Поскольку объединение в источнике выборки возвращает информацию о каждом контракте каждого сотрудника, то в результирующей таблице появляются дублирующиеся записи. Для исключения этих записей в начале раздела SELECT используем ключ DISTINCT:

SELECT DISTINCT Сотрудники.ФИО, [Сотрудники].[Город]

Результат работы окончательного варианта запроса:

ФИО	Город
Белова Е.А.	Витебск
Иванов С.В.	Витебск

Группировка

Для каждого отдела выяснить средний рейтинг и среднюю заработную плату сотрудников, при этом в каждом отделе выделить группы сотрудников родом из одного города.

```
SELECT Отделы.Наименование AS Отдел, Отделы.Телефон,
        Avg(Сотрудники.Рейтинг) AS [Средний рейтинг],
        Avg(Сотрудники.Зарплата) AS [Средняя зарплата], Сотрудни-
        ки.Город
FROM
        Отделы INNER JOIN Сотрудники
        ON Отделы.Номер_отдела = Сотрудники.Номер_отдела
GROUP BY
        Отделы.Наименование,
        Отделы.Телефон, Сотрудники.Город;
```

Результат работы запроса:

Отдел	Город	Телефон	Средний рейтинг	Средняя зарплата
№1	Витебск	485521	62,5	1800000
№2	Минск	485065	30	1600000
№3	Брест	485055	50	1500000
№3	Гродно	485055	120	2000000
№3	Минск	485055	24	1500000

Фильтрация сгруппированной таблицы

Вывести сведения об отделах, в которых имеются сотрудники с одинаковым рейтингом

```
SELECT DISTINCT Отделы.Наименование
FROM Отделы INNER JOIN Сотрудники ON Отделы.Номер_отдела =
        Сотрудники.Номер_отдела
GROUP BY Отделы.Наименование, Сотрудники.Рейтинг
HAVING (((Count(Сотрудники.Рейтинг))>1));
```

Результат запроса – пустая таблица, так как в нашей базе данных нет сотрудников с одинаковым рейтингом.

Использование подзапроса, возвращающего единственное значение

Выбрать данные о сотрудниках, сумма контрактов которых превышает среднюю сумму контрактов по организации.

Для начала нужен подзапрос, вычисляющий среднюю сумму заказов по организации. Основной запрос должен: объединить три таблицы «Сотрудники», «Исполнители» и «Заказы», сгруппировать данные по фамилиям сотрудников, вычислить агрегатную функцию **sum(Заказы.Сумма)** для каждой группы и из полученной таблицы извлечь только те записи, в которых значение **sum(Заказы.Сумма)** будет больше, чем значение возвращаемое подзапросом.

```
SELECT Сотрудники.ФИО, sum(Заказы.Сумма) as Сумма
FROM Заказы INNER JOIN (Сотрудники INNER JOIN Исполнители
        ON Сотрудники.Табельный_номер = Исполните-
        ли.Табельный_номер) ON Заказы.Номер_контракта = Испол-
        нители.Номер_контракта
GROUP BY Сотрудники.ФИО
HAVING   sum(Заказы.Сумма)> (SELECT Avg(Заказы.Сумма) FROM
        Заказы );
```

Итоговая результирующая таблица выглядит так:

ФИО	Сумма
Белова Е.А.	4150000
Васильев В.А.	4750000
Иванов С.В.	7050000
Павлов П.А.	2900000
Сидоров П.П.	7600000

Использование подзапроса в качестве источника выборки

Выяснить, сколько заработали сотрудники каждого отдела за июнь с учётом того, что каждому сотруднику, заключившему контракт, положен бонус в размере 5 % от суммы заказа.

Вначале составим подзапрос для определения суммы бонусов для каждого сотрудника за июнь:

```
SELECT Сотрудники.ФИО, Sum(Заказы.Сумма*0.05) AS Бонусы
FROM Сотрудники INNER JOIN (Заказы INNER JOIN Исполнители
    ON Заказы.Номер_контракта=Исполнители.Номер_контракта)
    ON Сотрудники.Табельный_номер=Исполнители.Табельный_номер
WHERE (Заказы.Дата>=#6/1/2010#) And (Заказы.Дата<#7/1/2010#)
GROUP BY Сотрудники.ФИО, Сотрудники.Зарплата;
```

Подзапрос возвращает следующую таблицу:

ФИО	Бонусы
Белова Е.А.	120000
Иванов С.В.	80000
Сидоров П.П.	75000

Сохраним подзапрос в базе данных с именем SubQ, это упростит текст основного запроса.

Для выполнения основного задания необходимо к полученной от запроса «SubQ» таблице добавить сведения о зарплатах, причём нужно сделать так, чтобы сотрудники, не получившие бонус в июне, также попали в эту таблицу:

ФИО	Номер_отдела	Зарплата	Бонусы
Белова Е.А.	1	2000000	2120000
Васильев В.А.	4	2000000	
Иванов С.В.	1	1600000	1680000
Павлов П.А.	4	1500000	
Петров Е.А.	2	1600000	
Сидоров П.П.	4	1500000	1575000

Для создания такой таблицы нужно использовать запрос, который объединяет левым объединением таблицу «Сотрудники» и подзапрос «SubQ» по равенству значений в полях ФИО.

```
SELECT Сотрудники.ФИО, Сотрудники.Номер_отдела, Сотрудни-
        ки.Зарплата, SubQ.Бонусы
FROM Сотрудники LEFT JOIN SubQ ON
        SubQ.ФИО=Сотрудники.ФИО;
```

Теперь необходимо сложить поля «Зарплата» и «Бонус» в полученной таблице и суммировать эти значения в группах для каждого номера отдела. Однако здесь существует проблема: при сложении числа и пустого значения (NULL) будет получено пустое значение.

Для того чтобы вместо NULL получить число 0, можно использовать функцию IIf и функцию проверки IsNull:

```
IIf(IsNull(Sum(SubQ.Бонусы)),0,Sum(SubQ.Бонусы))
```

Если значение Sum(SubQ.Бонусы) пустое, то вернуть 0, иначе вернуть значение Sum(SubQ.Бонусы). Полный текст основного запроса выглядит так:

```
SELECT Сотрудники.Номер_отдела,
        IIf(IsNull(Sum(SubQ.Бонусы)),0,Sum(SubQ.Бонусы))+Sum(Сотру
        дники.Зарплата) AS Итого
FROM Сотрудники LEFT JOIN SubQ ON
        SubQ.ФИО=Сотрудники.ФИО
GROUP BY Сотрудники.Номер_отдела;
```

Результат работы запроса:

Номер отдела	Общий заработок
1	4160000
2	1632500
4	5762500

Использование соотнесённых подзапросов

Вывести для каждого отдела фамилию сотрудника с максимальным для данного отдела рейтингом.

Если бы в условии отсутствовало требование вывести фамилию сотрудника, то запрос содержал бы только группировку по полю «Номер_отдела» и вычисление агрегатной функции **max(Рейтинг)**.

Одним из вариантов решения этой задачи является использование запроса с соотнесённым подзапросом:

```
SELECT Номер_отдела, ФИО, Рейтинг
FROM Сотрудники  a
WHERE a.Рейтинг =(
                        SELECT max(Рейтинг)  FROM  Сотрудники  b
                    WHERE a.Номер_отдела = b.Номер_отдела
) ;
```

Если обычный подзапрос выполняется только один раз, то соотнесённый подзапрос выполняется один раз для каждой строки источника выборки основного запроса.

В основном запросе происходит фильтрация записей источника выборки в соответствии с условием:

```
a.Рейтинг =(
                SELECT max(Рейтинг)  FROM  Сотрудники  b
              WHERE a.Номер_отдела = b.Номер_отдела
              ) ;
```

При этом происходит последовательный анализ каждой записи на соответствие критерию, начиная с первой (рис. 20).

Обратите внимание на использование псевдонимов: для того, чтобы создать копию обрабатываемой таблицы «Сотрудники» внутри подзапроса для неё вводится псевдоним «a», в основном же запросе эта таблица имеет псевдоним «b».

Через выражение **a.Номер_отдела** в подзапрос попадает значение из поля «Номер_отдела» текущей строки таблицы «Сотрудники» основного запроса. Подзапрос выбирает из всей таблицы «Сотрудники», которая также и для него является источником данных, все записи соответствующие условию **a.Номер_отдела = b.Номер_отдела**. После этого выбирается максимальный рейтинг для определённого отдела, и полученное значение передаётся в основной запрос.

Обработка первой строки в основном запросе

ФИО	Рейтинг	Номер_отдела
Белова Е.А.	100	1
Сидоров П.П.	24	4
Иванов С.В.	25	1
Петров Е.А.	30	2
Васильев В.А.	120	4
Павлов П.А.	50	4

Подзапрос

SELECT max(Рейтинг) FROM Сотрудники b
WHERE 1 = b.Номер_отдела

ФИО	Рейтинг	Номер_отдела
Белова Е.А.	100	1
Иванов С.В.	25	1

SELECT Номер_отдела, [ФИО], Рейтинг
FROM Сотрудники a
WHERE a.Рейтинг = 100

max(Рейтинг) = 100

ФИО	Рейтинг	Номер_отдела
Белова Е.А.	100	1
Сидоров П.П.	24	4
Иванов С.В.	25	1
Петров Е.А.	30	2
Васильев В.А.	120	4
Павлов П.А.	50	4

Первая строка включается в результирующую таблицу.

Рисунок 20 – Обработка одной строки в запросе с соотнесённым подзапросом

Полученное значение сравнивается со значением рейтинга в текущей записи и, если оно совпадает, то запись включается в результирующую таблицу. Затем подобным образом обрабатывается вторая запись в источнике выборки основного запроса.

Итоговая результирующая таблица выглядит так:

Номер_отдела	ФИО	Рейтинг
1	Белова Е.А.	100
2	Петров Е.А.	30
4	Васильев В.А.	120

Использование объединения UNION

Результирующая таблица будет представлять собой вертикально объединение (т.е. объединение записей) двух запросов, первый из которых возвращает список лидеров, второй – список аутсайдеров.

Напомним основные требования к запросам, входящим в объединение UNIT:

- результирующие таблицы запросов должны иметь одинаковое количество полей;
- соответствующие поля результирующих таблиц должны быть совместимы для объединения, т. е. иметь совместимый тип.

Вывести сведения о лидерах и аутсайдерах среди сотрудников. Лидером считается сотрудник с рейтингом, превышающим 90 пунктов, аутсайдером – сотрудник с рейтингом ниже 30.

```
SELECT ФИО, Зарплата,  "лидер" as [Статус]
FROM Сотрудники
WHERE Рейтинг>90
UNION
SELECT ФИО, Зарплата, "аутсайдер" as [Статус]
FROM Сотрудники
WHERE Рейтинг<30
ORDER BY ФИО
```

Обратите внимание на поле «Статус»: каждый из запросов содержит в нём соответствующую константу.

Раздел **ORDER BY** в данном случае используется для сортировки сразу всех записей результирующей таблицы.

Итоговая результирующая таблица выглядит так:

ФИО	Зарплата	Статус
Белова Е.А.	2000000	лидер
Васильев В.А.	2000000	лидер
Иванов С.В.	1600000	аутсайдер
Сидоров П.П.	1500000	аутсайдер

Задания для самостоятельной работы

Вариант 1. База данных ПОСТАВКА ТОВАРОВ

1. Вывести сведения о товарах, для которых разница между ценой и себестоимостью не превышает 20 %.
2. Вывести номера телефонов и фамилии заказчиков, заказывавших товары более чем двух наименований.
3. Вывести данные о заказах объёмом более 1000 шт., сделанных заказчиками из городов Витебск или Минск.
4. Вывести ФИО и телефон заказчика, который оформил последний заказ в 2006 году.
5. Вывести сведения о заказчиках, заказывавших только те товары, цена на которые ниже средней цены всех товаров.
6. Вывести цену, общее количество заказов и наименование товара, имеющего наибольшую разницу между ценой и себестоимостью.
7. Определить общее количество товаров каждого наименования, отгруженных к 10.02.2005.
8. Определить общую сумму, полученную от реализации товаров в январе 2007 года, за вычетом себестоимости.
9. Вывести цену, общее количество заказов и наименование товаров заказывавшихся в ноябре 2007 г.
10. Определить заказчиков (вывести ФИО и телефон), заказавших товаров, на общую сумму больше 20000 руб.
11. Вывести сведения о товарах, цена на которые ниже средней цены всех товаров, предлагаемых организацией.
12. Вывести сведения о количестве заказчиков из городов Минск и Витебск (отдельно).
13. Определить заказчиков (вывести ФИО и телефон), заказавших товаров на общую сумму большую, чем сумма, на которую сделали заказ заказчики из Смоленска.
14. Составить таблицу изменения объёма продаж в стоимостном выражении по трём месяцам в виде:

Месяц	**Объём**
июнь	2000
июль	25020
август	35000

15. Для каждого вида товара определить дату отгрузки самой большой партии.

Вариант 2. База данных РОЗНИЧНАЯ ТОРГОВЛЯ

1. Вывести названия и номера телефонов поставщиков, поставлявших принтеры летом 2006 года.
2. Вывести сведения о московских или минских поставщиках, поставляющих мониторы.
3. Определить поставщиков, поставляющих более двух видов товаров.
4. Вывести сведения о количестве поставщиков, размещающихся в каждом городе.
5. Определить общую стоимость находящихся на складе товаров.
6. Вывести общее количество на складе и наименование видов товаров, поставлявшихся в ноябре 2007 г.
7. Вывести сведения о товарах, цена на которые ниже средней цены всех товаров, предлагаемых фирмой.
8. Определить поставщиков (вывести ФИО и адрес), поставивших товаров на общую сумму больше 70000 руб.
9. Вывести сведения о количестве поставщиков из каждого города, за исключением Минска и Витебска.
10. Для каждого вида товара определить поставщика, поставляющего его по самой низкой цене.
11. Для каждого поставщика определить общее количество наименований поставляемых товаров.
12. Выбрать сведения о поставщике (название, телефон), который поставил в январе 2007 товар по самой большой цене.
13. В каком месяце произошла самая крупная покупка (вывести наименование товара).
14. Вывести статистику о количестве заказов в марте, апреле, мае 2006 года в виде:

Месяц	**Объём**
март	2
апрель	1
май	3

15. Вывести сведения о самом дорогом товаре, поставляемом каждым из поставщиков.

Вариант 3. База данных БАНКОВСКИЕ ВКЛАДЫ

1. Для каждого клиента определить общую сумму средств на всех счетах.
2. Вывести сведения о пополнениях счета наличными на сумму более 2000000 руб., сделанных в сентябре 2006 г.
3. Вывести историю операций определённого клиента со счётом №230034.

4. Для каждого клиента определить счет с максимальной суммой.
5. Вывести список клиентов, получавших средства со счёта №450501.
6. Определить общее количество счетов у клиентов, проживающих в Минске, и общее количество счетов у клиентов, проживающих в Гродно.
7. Вывести информацию о клиенте, имеющем наибольшее количество счетов.
8. Вывести информацию об обладателе самого большого, по количеству денежных средств, счёта.
9. Вывести фамилии и адреса вкладчиков, имеющих более двух счетов.
10. Для каждого города определить вкладчика, имеющего с максимальную сумму на счёте.
11. Вывести сведения о вкладчике, который пополнил счёт на самую большую сумму в январе 2007 г.
12. Определить, какого числа было открыто больше всего счетов.
13. Вывести информацию о вкладчике, который суммарно снял со счёта самую большую сумму в декабре 2008 г.
14. Вывести статистику по количеству открытых вкладов по трём месяцам в виде:

Месяц	Количество
Сентябрь	27
Октябрь	21
Ноябрь	33

15. Вывести сведения о самом большом пополнении наличностью каждого из счетов.

Вариант 4. База данных ТОРГОВЛЯ

1. Вывести сведения о товарах, продаваемых в отделе № 2, количество купленных единиц за одну продажу которых превысило 3.
2. Вывести сведения о товарах с ценой до 2000 руб., продаваемых в отделах № 1 и № 2.
3. Вывести суммарный объём (габарит), занимаемый покупкой № 456.
4. Вывести количество запросов (без учёта количества покупаемых единиц товара) на покупку товаров в отделе № 2.
5. Вывести сведения (название и телефон) об отделах, получивших за октябрь 2007 г выручку более 2000000 руб.
6. Вывести сведения о товарах, имеющих наценку не более 10 %;
7. Получить общую сумму выручки для каждого отдела за период с 15 ноября 2008 года по 2 января 2009 года.
8. Вывести информацию о наиболее продаваемом товаре (по количеству).

9. Вывести информацию о товаре, "сделавшем" наибольшую выручку.
10. Для каждого отдела вывести информацию о самом дешевом товаре.
11. Вывести фамилию заведующего отделом, в котором была получена наибольшая дневная выручка за декабрь 2008.
12. Вывести список отделов, выручка определённый период которых меньше средней выручки отделов за этот же период.
13. Вывести информацию о количестве наименований товаров, продаваемых в отделах № 1 и № 2.
14. Вывести статистику по объемам реализации товаров определённого вида в магазине по месяцам в виде:

Месяц	Количество
Сентябрь	20
Октябрь	15
Ноябрь	30

15. Вывести сведения о самом дорогом товаре для каждой из покупок, сделанных в отделе № 2.

Вариант 5. База данных АВИАКОМПАНИЯ

1. Определить общее количество часов налёта каждого самолёта.
2. Вывести список самолётов типа Ту, выполнявших рейсы на Москву и Варшаву.
3. Определить общее количество рейсов, совершённых самолётами типа Як-40.
4. Для каждого пилота вывести статистику о количестве совершённых рейсов в качестве командира экипажа.
5. Вывести список командиров экипажей, выполнявших рейсы на Москву.
6. Определить суммарное количество часов у каждой стюардессы и пилота.
7. Определить пункт назначения самого часто выполняемого рейса.
8. Вывести список пилотов, выполнявших рейсы только в роли командира экипажа.
9. Определить количество часов налёта пилотом Сидоровым в качестве командира экипажа.
10. Определить, где находился Ту-154 №123332 12 января 2006 г.
11. Определить общее количество часов налёта самолета, который чаще других пилотов пилотировал пилот Иванов в качестве командира экипажа.
12. Определить, сколько раз работали в паре проводницы Петрова В. В. и Петрова А. В. в первой половине 2006 года.
13. Определить, какого числа было совершено самое большое количество рейсов.

14. Составить график капитального ремонта самолётов: если с момента последнего техосмотра прошло более пяти лет, то в графе дата следующего техосмотра необходимо поместить: “срочно”.

Самолёт	№ борта	Техосмотр
Ту-144	544	не срочно
Ту-144	566	не срочно
Як-42	501	срочно

15. Для каждого экипажа вывести сведения о проводнице с самым большим стажем работы.

СУБД MS SQL Server

MS SQL Server является клиент-серверной СУБД. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия, а также для web-приложений.

Microsoft SQL Server Express является бесплатно распространяемой версией SQL Server. Данная версия имеет некоторые технические ограничения. Такие ограничения делают её непригодной для развертывания больших баз данных, но она вполне годится для ведения программных комплексов в небольших масштабах. Поэтому MS SQL Server используется в приложениях, при проектировании или для самостоятельного изучения.

SQL Server поставляется в виде пакета программ, среди которых можно выделить два типа.

Службы – программы или процессы, выполняющие специфические функции поддержки других программ. Службы обычно запускаются в фоновом режиме. Большая часть служб MS SQL Server запускается автоматически при загрузке операционной системы. Среди служб можно назвать такие, как: **Microsoft Distributed Transaction Coordinator** (менеджер распределённых транзакций), **SQL Server Agent** (планировщик заданий), **Microsoft Search** (служба полнотекстного индексирования и поиска).

Утилиты – программы, выполняющие разного рода функции, связанные с администрированием, обслуживанием и работой.

Когда вы запускаете SQL Server, в операционной системе Windows NT или Windows 2000 запускается служба **sqlservr – ядро СУБД**.

SQL Server Agent осуществляет планирование и исполнение заданий, оповещений, извещений и планов обслуживания базы данных.

Microsoft Distributed Transaction Coordinator – это администратор транзакций (transaction manager), при помощи которого в транзакции ваших приложений можно включать данные из различных источников, в том числе данные

из баз данных с удаленных компьютеров. Это значит, что при помощи одной транзакции можно обновлять данные на многих серверах, доступных через сеть.

Microsoft Search – поддержка полнотекстного индексирования и поиска. Благодаря полнотекстному индексированию возможно выполнение более сложного поиска среди данных, содержащих текстовые строки. Например, вы можете искать слова, близкие к заданному слову, или можете искать определенную фразу.

Интерфейс утилиты Enterprise Manager

Утилита **Enterprise Manager** предназначена для администрирования СУБД SQL Server. Утилита выполняет следующие основные функции:

- конфигурирование локальных и удаленных серверов,
- конфигурирование многосерверных инсталляций и управление ими,
- управление учётными записями и политикой безопасности, а также разграничением доступа к объектам СУБД,
- создание и планирование заданий,
- конфигурирование SQL Server для оповещения системных администраторов через электронную почту,
- создание объектов базы данных (таблиц, индексов, триггеров и т. п.)
- управление резервным копированием, журналами ошибок,
- управление клиентскими сеансами работы с СУБД.

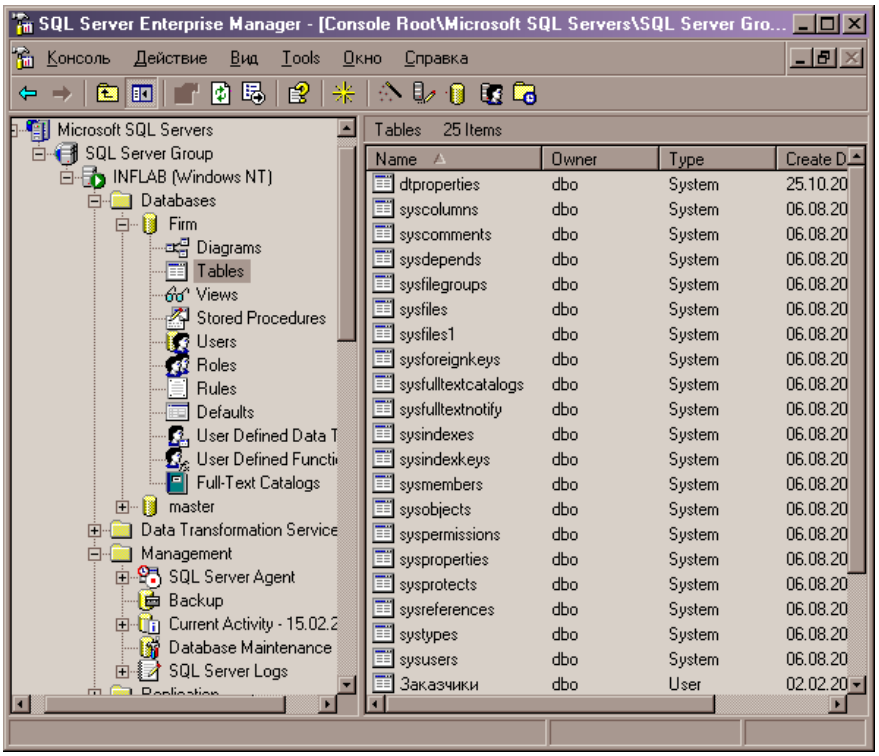


Рисунок 21 – Окно утилиты Enterprise Manager

Окно утилиты состоит из двух частей (рисунок 21):

- слева – иерархическая структура, содержащая все объекты СУБД SQL Server, начиная со списка доступных по сети серверов СУБД SQL Server, заканчивая отдельными полями в таблицах,
- справа – список объектов, которые содержаться в объекте, выделенном на правой панели.

Такой подход к построению интерфейса характерен для всех надстроек Microsoft Management Console (MMC) операционной системы Windows, чем, по сути, и является утилита Enterprise Manager.

Для создания и редактирования объектов утилита Enterprise Manager имеет большое количество диалоговых окон, конструкторов и мастеров, которые вызываются с помощью контекстного меню. Интерфейс основных конструкторов, таких как конструкторы таблиц, запросов, схем данных, сходен с интерфейсом соответствующих конструкторов в MS Access.

Интерфейс утилиты Query Analyzer

Утилита **Query Analyzer** предназначена для отладки запросов на языке T-SQL,

Query Analyzer позволяет:

- разрабатывать и исполнять сценарии на языке T-SQL,
- получать и сохранять результаты работы сценариев в форматированном виде,
- анализировать индексы запросов.

Утилита представляет многодокументный тестовый редактор с анализатором кода, который позволяет форматировать цветом различные типы конструкций языка SQL.

У левой границы окна можно расположить браузер объектов (1 на рисунке 22). Окно браузера имеет две вкладки. На вкладке Objects (объекты) в виде иерархии представлены все объекты сервера СУБД, к которому подключена утилита (утилита позволяет подключиться только к одному серверу, диалоговое окно подключения появляется сразу после запуска утилиты). Также на данной вкладке представлен Common Objects (общие объекты) в который включены разбитые по категориям функции и стандартные типы данных. Эту ветвь можно использовать для вставки шаблонов вызовов и в качестве справочника.

На вкладке Templates (шаблоны), которая помечена меткой 2 на рисунке 22, можно найти шаблоны создания различных объектов (таблиц, триггеров, хранимых процедур и т. п.), а также заготовки некоторых операций (использование курсоров, управление правами доступа, подключение серверов и т. п.).

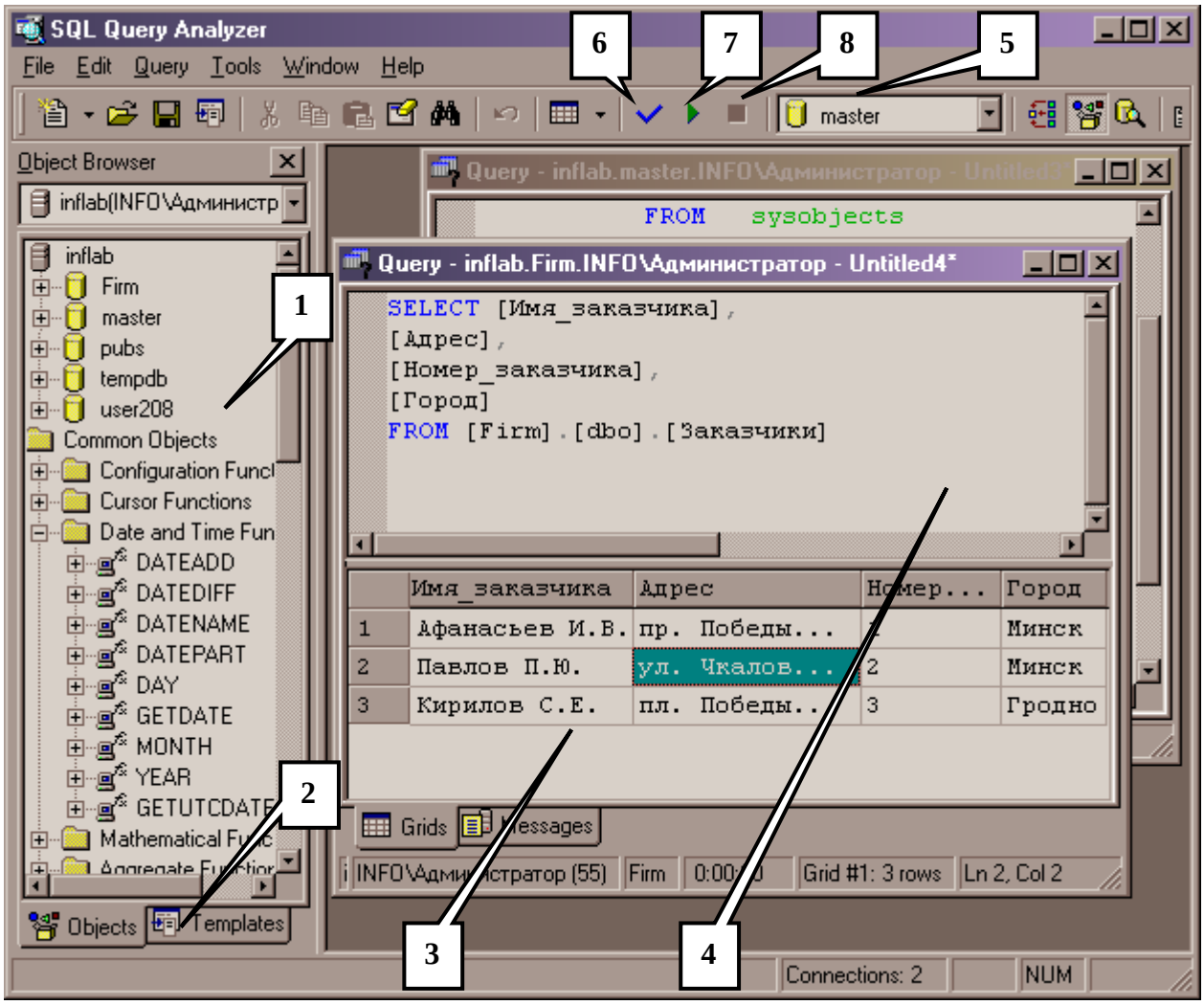


Рисунок 22 – Окно утилиты Query Analyzer

При использовании любого из шаблонов утилита создаёт новое окно и помещает в него требуемый пакет команд на языке T-SQL. В тексте шаблона имеются вставки, заключённые в треугольные скобки (<>). При использовании шаблона эти вставки необходимо заменить собственными конструкциями (идентификаторами, выражениями и т. д.).

Рассмотрим назначение основных элементов на панели инструментов (рисунок 22):

5 – контекстная база данных, устанавливается индивидуально для каждого окна редактора, фактически заменяет собой команду:

USE Идентификатор_базы_данных;

6 – проверка синтаксиса пакета, запуск пакета на исполнение при этом не происходит;

7 – запуск пакета на исполнение;

8 – остановка выполнения пакета.

Окно редактора кода может быть разделено на две части: верхняя часть (4 на рисунке 22) используется для редактирования кода, нижняя часть (3 на рисунке 22), которая может быть отключена, используется для визуализации результата запроса (результирующей таблицы (вкладка «Grids»), или для размещения сообщений сервера базы данных (вкладка «Messages»).

Таблица 1 – Некоторые сообщения, размещаемые на вкладке «Messages»

Сообщение	**Описание**
The command(s) completed successfully	Команды выполнены успешно
(3 row(s) affected)	Команда повлияла на 3 строки.
Server: Msg 170, Level 15, State 1, Line 1 Line 1: Incorrect syntax near '<'.	Ошибка синтаксиса с указанием номера (Msg 170), строки (Line 1) и лексемы ('<'), рядом с которой она была обнаружена
Server: Msg 207, Level 16, State 3, Line 1 Invalid column name 'Номер_аказчика'.	Неправильное название поля

Разработка скрипта создания базы данных

В качестве стандартного языка управления базами данных в MS SQL Server используется язык Transact SQL (T-SQL). Основное его отличие от диалекта Jet-SQL состоит в возможности пакетного исполнения команд. Такой пакет команд называют скриптом. В языке T-SQL также имеются управляющие конструкции (циклы, ветвления), с помощью которых можно организовать команды в определённый алгоритм.

Таким образом, используя возможности пакетного режима, можно разработать скрипт, который при запуске создаст базу данных заданной структуры. С помощью таких скриптов можно перемещать базы данных между серверами, хотя надо отметить, что в различных СУБД существуют и другие инструменты импорта-экспорта баз данных.

Непосредственно для создания базы данных используется команда:

```
CREATE DATABASE <имя_базы_данных>
```

Однако у слушателей, использующих SQL Server INFLAB на лабораторных работах, отсутствуют права добавления базы данных, поэтому будет рас-

смотрен пример скрипта добавляющего группу связанных таблиц в уже существующую базу данных.

Напомним, что для создания таблиц используется команда CREATE TABLE.

Синтаксис команды создания таблицы:

```
  CREATE TABLE <имя_таблицы>
     (
         <имя_поля1> <тип_поля1> [NOT NULL][UNIQUE | PRI-
MARY KEY]
              [REFERENCES <имя_мастер_таблицы>
[(<имя_связанного_поля>)]],
          ...
        [CONSTRAINT <имя_ограничения>
                  PRIMARY KEY (<имя_поля1>[, ...]) |
               FOREIGN KEY (<имя_ поля1>[, ...])
                        REFERENCES <имя_связанной_таблицы>

(<имя_связанного_поля1>[, ...]),
          ... ]
     )
```

Синтаксис команды удаления таблицы:

```
  DROP TABLE <имя_таблицы>
```

Таблицу можно рассматривать как компонент, состоящий из полей, ограничений и триггеров. Ограничения – это специальные объекты базы данных, связанные с таблицами, предназначенные для создания первичных и внешних ключей, а также для задания выражений, контролирующих вводимые в таблицу данные (ограничения CHEK в данных указаниях не рассматриваются).

С помощью команды CREATE TABLE создаются поля (часть команды **до** ключевого слова CONSTRAINT) и ограничения таблицы (часть команды **после** ключевого слова CONSTRAINT).

Следует обратить внимание на то, что внешние ключи могут быть созданы непосредственно при описании поля (конструкция REFERENCES). В этом случае также создаётся ограничение, но такой способ его создания называют неявным. Несмотря на различные способы описания, каждое из создаваемых ограничений является отдельным, связанным с таблицей, объектом. Имена для неявно описанных ограничений задаются самим ядром СУБД.

Для удобства отладки необходимо включать в скрипт команды очистки базы данных от ранее созданных объектов, которые могли остаться от предыду-

щих запусков скрипта. Такая процедура обычно содержит проверку наличия объекта и команду его удаления, в случае, если он находится в базе данных.

Для проверки наличия объекта используется системная таблица «sysobjects», в которой имеется по одной записи для каждого объекта базы данных (таблицы, хранимой процедуры, ограничения, триггера и т. д.).

В таблице 2 приведено описание некоторых наиболее полезных полей системной таблицы «sysobjects».

Таким образом, для запуска скрипта не потребуется вручную подготавливать базу данных.

Таблица 2 – Поля таблицы «sysobjects»

Поле	Тип данных	Описание
name	**nvarchar(128)**	Имя объекта
id	**int**	Идентификатор объекта
type	**char(2)**	Тип объекта. Может принимать одно из значений, идентифицирующих тип объекта: C = ограничение CHECK, D = ограничение DEFAULT, F = ограничение FOREIGN KEY, K = ограничение PRIMARY KEY, P = хранимая процедура, S = системная таблица, TR = триггер, U = пользовательская таблица, V = представление.
parent_obj	**int**	Идентификатор хозяина объекта. Например: хозяином триггера является таблица, для которой он был создан.
crdate	**datetime**	Дата и время создания объекта.

Рассмотрим следующий пример:

Разработать скрипт для создания части структуры базы данных, приведённой на рисунке 13 . Для примера выберем три связанные таблицы: «Сотрудники», «Заказы», «Исполнители».

```
USE Firm
GO
if exists ( SELECT name FROM sysobjects
        WHERE name = 'Исполнители' AND type = 'U' )
drop table Исполнители
GO
```

```
if exists (  SELECT name FROM sysobjects
        WHERE  name = 'Сотрудники' AND type = 'U' )
drop table Сотрудники
GO
if exists (  SELECT name FROM sysobjects
        WHERE  name = 'Заказы' AND type = 'U' )
drop table Заказы
GO
CREATE TABLE Сотрудники (
  Табельный_номер int PRIMARY KEY,
  ФИО nvarchar (20),
  Номер_отдела int,
  Зарплата float,
  Рейтинг int,
  Город nvarchar (50)
)
GO
CREATE TABLE Заказы (
  Номер_заказчика int,
  Сумма real,
  Дата smalldatetime,
  Номер_контракта int  PRIMARY KEY
)
GO
CREATE TABLE Исполнители (
  Табельный_номер int REFERENCES Сотрудники (Табельный_номер),
  Номер_контракта int REFERENCES Заказы (Номер_контракта),
  CONSTRAINT Ключ PRIMARY KEY (Табельный_номер, Номер_контракта )
)
GO
```

В первую очередь используем команду **USE Firm** для подключения к базе данных. Это избавляет от необходимости следить за тем, какая база данных в данный момент является активной (5 на рисунке 22). Нужно обратить внимание на последовательность размещения команд удаления и создания таблиц.

При выполнении команды создания таблицы «Исполнители» создаётся три ограничения: явно описанное ограничение «Ключ» и два неявно описанных ограничения внешнего ключа.

Для успешного выполнения скрипта необходимо сначала удалить таблицу «Исполнители», так как с этой таблицей связаны ограничения, ссылающиеся на две другие таблицы.

Однако имеются и другие варианты. Если вначале скрипта удалить все связанные с таблицами объекты (в том числе и ограничения), то последовательность удаления таблиц уже не будет играть никакой роли.

Задания для самостоятельной работы

Для всех вариантов:

- Разработать скрипт создания базы данных, разработанной при выполнении задания из раздела «Проектирование баз данных».
- Включить в скрипт по одному запросу на вставку записи в каждую из таблиц базы данных.

Разработка хранимых процедур

Хранимая процедура (stored procedure) – это именованный набор команд, хранящийся непосредственно на сервере и представляющий собой самостоятельный объект базы данных.

Для создания хранимой процедуры на языке используется команда CREATE PROCEDURE:

```
CREATE PROC[EDURE] <Имя_процедуры>
[@<параметр1>  <тип_параметра1>]
[= Значение_по_умолчанию ] [ OUTPUT ] ] ,
...
AS
пакет sql команд
```

Вызов хранимой процедуры выполняется с помощью следующей команды:

```
EXEC <Имя_процедуры>
[ [@<параметр1> = ] выражение | @<Имя_переменной> [
OUTPUT ] | [ DEFAULT ] ] ,
…
```

Создание и использование хранимых процедур не отличается от создания и использования подпрограмм в обычном алгоритмическом языке программирования.

Рассмотрим несколько особенностей.

Обратите внимание, что при вызове процедуры имеется возможность указать название параметра. Такая конструкция вызова позволяет не принимать во внимание последовательность указания параметров при вызове.

Например, имеется следующий заголовок процедуры:

```
CREATE PROCEDURE Пр1
  @A varchar(50),   @B real,  @C int
```

Вызовы процедуры **Пр1** могут быть составлены такими образом:

```
EXEC Пр1 @C = 1, @A = 'Str', @B = 1.001
EXEC Пр1 'Str', 1.001, 1
```

Значение по умолчанию при вызове процедуры принимается либо в случае отсутствия параметра, с заданным по умолчанию значением, в вызове, либо при использовании ключа **DEFAULT**.

Например, имеется следующий заголовок процедуры:

```
CREATE PROCEDURE Пр2
  @A varchar(50),   @B real = 12.5 ,  @C int
```

Вызовы процедуры **Пр2** могут быть составлены такими образом:

```
EXEC Пр2 @C = 1, @A = 'Str'
EXEC Пр2 'Str', DEFAULT, 1
```

Параметр, описанный при создании процедуры с ключом **OUTPUT** (например, **@Prm int OUTPUT),** может быть использован при вызове несколькими способами:

1. **@Парам = 3** – в этом случае параметр инициализируется значением 3 и, естественно, никакое значение из процедуры не будет возвращено;
2. **@Парам = @Пр** – в этом случае параметр инициализируется значением, взятым из переменной **@Пр**, но после окончания работы процедуры значение переменной **@Пр** не изменится, т. е. этот случай аналогичен первому варианту;
3. **@Парам = @Пр OUTPUT** – только при таком способе инициализации значение переменной **@Пр** может быть изменено в процедуре и, таким образом, будет осуществлён вывод результата из процедуры.

Рассмотрим следующий пример:

Разработать хранимую процедуру добавления нового заказа в базу данных «Организация». При добавлении заказа указывается имя заказчика, сумма заказа, ФИО исполнителя. Если заказчик с указанным именем отсутствует в базе данных, то добавление заказа следует отменить. Если исполнитель с указанным ФИО отсутствует в базе данных, то исполнителем заказа необходимо назначить сотрудника с минимальным рейтингом, при этом необходимо возвратить ФИО.

```
-- =======очистка базы данных====================
USE Firm
GO
IF EXISTS (SELECT name
    FROM   sysobjects
    WHERE  name = 'AddZk'
    AND         type = 'P')
   DROP PROCEDURE AddZk
GO
-- ==== Создание процедуры ======================

CREATE PROCEDURE AddZk
  @Name varchar(50),
  @Sum real,
  @Isp varchar(50) = '' OUTPUT
AS
      DECLARE @NZ int
      DECLARE @Id int;
      DECLARE @TNomIsp int;

-- ======== Блок №1============================
  if NOT EXISTS (SELECT Имя_заказчика
    FROM   Заказчики

    WHERE  Имя_заказчика = @Name)
      begin
         PRINT 'Заказчик не найден!'
         RETURN(1)
      end
-- ======== Блок №2============================

      if @Isp <> '' AND  NOT EXISTS (SELECT ФИО
    FROM   Сотрудники
    WHERE  ФИО = @Isp)
      begin
         PRINT 'Исполнитель не найден!'
```

```
            RETURN(2)
        end
    -- ======== Блок №3============================

        if @Isp = ''
        begin
             SELECT TOP 1 @Isp = ФИО
            FROM   Сотрудники
            WHERE  Рейтинг = (   SELECT MIN( Рейтинг)
                                 FROM   Сотрудники
                              )
        end
    -- ======== Блок №4============================

        SELECT @NZ = [Номер_заказчика]
           FROM   Заказчики
      WHERE  Имя_заказчика = @Name

        SELECT @TNomIsp = [Табельный_номер]
      FROM   Сотрудники
      WHERE  ФИО = @Isp

        SELECT @Id = MAX(Номер_контракта)+1
      FROM   Заказы;

    -- ======== Блок №5============================

       INSERT INTO [Заказы] ([Номер_контракта], [Номер_заказчика],
[Сумма], Дата)
       VALUES(@Id, @NZ, @Sum, getdate());

       INSERT INTO [Исполнители]([Табельный_номер], [Но-
мер_контракта])
       VALUES(@TNomIsp, @Id)
    GO

    -- ========= Запуск процедуры ======================

    EXECUTE AddZk @Name = 'Афанасьев И.В.', @Sum = 20, @Isp =
'Волкова А.П.'
    GO
```

В начале скрипта очищаем базу данных, как было описано в пункте «Разработка скрипта создания базы данных». Затем следует заголовок процедуры. Обратите внимание на описание параметра:

@Isp varchar(50) = '' OUTPUT

Этот параметр принимает имя исполнителя, имеет в качестве значения по умолчанию пустую строку и может использоваться для вывода данных из процедуры.

Далее следует тело процедуры:

Блок №1 – проверка наличия заказчика с наименованием, переданным в параметре **@Name.**

Блок №2 – проверка наличия исполнителя с фамилией, переданной в параметре **@Isp**. Если параметр **@Isp** содержит пустую строку, то данная проверка не выполняется.

Блок №3 – если **@Isp** содержит пустую строку, то выполняется выборка из базы данных ФИО сотрудника с наименьшим рейтингом. Конструкция **TOP 1** позволяет извлекать только одну, первую, запись, в случае если минимальное значение рейтинга окажется у нескольких сотрудников.

Блок №4 – используется для извлечения из базы данных табельного номера исполнителя и номера заказчика для добавления записи в таблицу «Исполнители». Последняя команда получает уникальное значение номера контракта для вносимого в базу данных заказа.

Блок №4 – используется непосредственно для внесения записей в таблицы «Заказы» (причём для заполнения поля «Дата» используется функция **getdate()**, возвращающая значение текущей даты) и «Исполнители».

Последняя часть скрипта предназначена для пробного запуска процедуры.

Сразу после запуска скрипта нужно проверить правильность его работы, открыв таблицы «Заказы» и «Исполнители».

Для того чтобы проверить правильность работы процедуры, необходимо запустить этот скрипт несколько раз, внося изменения в последнюю его часть.

Задания для самостоятельной работы

Вариант 1. База данных ПОСТАВКА ТОВАРОВ

Разработать процедуру добавления нового пункта в заказ. Входные параметры: имя клиента, номер товара, количество.

Новый пункт добавляется в последний оформленный на данного клиента заказ.

Если в заказе уже значился добавляемый товар, то необходимо увеличить количество данного товара в соответствующем пункте заказа, в противном случае – добавить новый пункт.

Вариант 2. База данных РОЗНИЧНАЯ ТОРГОВЛЯ

Разработать процедуру добавления сведений о поставке товара. Входные параметры: имя поставщика, номер товара, количество, цена товара, дата поставки.

Если происходит добавление в уже существующую поставку (дата поставки и номер поставщика совпадают с датой поставки и номером поставщика одной из имеющихся в базе поставок), то необходимо увеличить количество товара и обновить цену в записи о поставке, в противном случае – добавить новую запись о поставке.

Вариант 3. База данных БАНКОВСКИЕ ВКЛАДЫ

Разработать процедуру добавления новой операции со счётом. Входные параметры: номер счёта, тип операции (наличный/безналичный; приход/расход), номер счёта (для операций с безналичным расчётом), сумма.

Изменить баланс счёта (или балансы обоих счетов, если осуществляется операция с безналичным расчётом).

Если добавляется операция снятия со счёта суммы, превышающей текущий баланс, то оформить снятие всей имеющейся на счёте суммы и закрыть счёт (удалить запись о данном счёте).

Вариант 4. База данных ТОРГОВЛЯ

Разработать процедуру добавления нового пункта покупки товара в чек. Входные параметры: название товара, количество, номер чека.

Если происходит добавление новой покупки (номер чека не совпадает ни с одним номером чека из имеющихся в таблице покупок), то необходимо добавить новый чек.

Вариант 5. База данных АВИАКОМПАНИЯ

Разработать процедуру добавления нового члена экипажа. Входные параметры: ФИО сотрудника, код рейса, место в экипаже.

Проверить: не занят ли сотрудник в этот день в другом рейсе, не занято ли его место в экипаже (экипаж должен состоять из: командира, пилота, стрелка-радиста и трёх стюардесс). Если один из пунктов проверки не соответствует требованию, то добавление нужно отменить.

Разработка триггеров

Триггер – это специальный тип хранимых процедур, запускаемых сервером автоматически при выполнении тех или иных действий с данными таблицы. Каждый триггер привязывается к конкретной таблице. При попытке,

например, изменить данные в таблице, сервер автоматически запускает триггер и, только если он завершается успешно, разрешается выполнение изменений.

Все производимые триггером модификации данных рассматриваются как одна транзакция. В случае обнаружения ошибки или нарушении целостности данных происходит откат этой транзакции. Тем самым внесение изменений будет запрещено и будут отменены также все изменения, уже сделанные триггером.

Синтаксис команды создания триггера:

```
CREATE TRIGGER <имя_триггера> ON
                        <имя_таблицы> | <имя_представления>
FOR | AFTER | INSTEAD
OF [DELETE] [,] [INSERT] [,] [UPDATE]
AS
пакет sql команд
```

У триггера не может быть ни параметров, ни возвращаемого значения. Для получения данных в триггере используются таблицы **DELETED** и **INSERTED** – временные таблицы, имеющие структуру, аналогичную структуре таблицы с которой связан триггер. В таблице **DELETED** содержится удаляемая запись, в таблице **INSERTED** – вставляемая. При обновлении записи в таблице **DELETED** будет храниться исходное состояние обновляемой записи, в таблице **INSERTED** – обновлённое.

Рассмотрим следующий пример:

Разработать триггер, который позволил бы контролировать назначение курирующих заказы сотрудников. Курировать заказы могут только сотрудники маркетинговых и коммерческих отделов. При добавлении нового курируемого заказа сотруднику необходимо увеличивать рейтинг на 10 пунктов.

```
-- =======очистка базы данных====================
Use firm
GO

IF EXISTS (SELECT name
    FROM   sysobjects
    WHERE  name = 'ChRt'
    AND         type = 'TR')
   DROP TRIGGER ChRt
GO

-- ==== Создание триггера ======================
CREATE TRIGGER ChRt
```

```
ON Исполнители
FOR INSERT
AS
BEGIN

-- ======== Блок №1============================
      if NOT EXISTS (
            select Сотрудники.Номер_отдела
            from (Сотрудники inner join INSERTED ON
            Сотрудники.Табельный_номер =
INSERTED.Табельный_номер)
            inner join Отделы ON
            Сотрудники.Номер_отдела = Отделы.Номер_отдела
            where Отделы.Наименование like '*маркетинг*' or
                Отделы.Наименование like '*коммерческий*'
        )
      begin

-- ======== Блок №2============================
        print 'Сотрудник не может курировать контракты!'
        rollback
      end
       else
      begin

-- ======== Блок №3============================
        update Сотрудники Set Рейтинг = Рейтинг + 10
        where Сотрудники.Табельный_номер = (
                                  select Табельный_номер
                                    from INSERTED
                                  )
        commit
      end
END
GO
```

В начале скрипта очищаем базу данных. Затем следует заголовок триггера. Триггер прикрепляется к таблице «Исполнители» перед вставкой в неё записи, именно так конкретный сотрудник назначается куратором конкретного заказа.

Далее следует тело триггера:

Блок №1 – проверка принадлежности сотрудника к коммерческому или маркетинговому отделу. В данном запросе объединяются таблицы: «**INSERTED**» (содержит вставляемую в таблицу «Исполнители» запись), «Со-

трудники» и «Отделы». Это делается для получения названия отдела, в котором работает сотрудник, назначаемый куратором контракта. Фильтр в разделе **where** позволит включить в результирующую таблицу только записи о сотрудниках, работающих либо в коммерческом, либо в маркетинговом отделе.

Блок №2 – возвращает в выходной поток сообщение о недопустимости операции и откатывает транзакцию вставки записи.

Блок №3 – изменяет запись о сотруднике, получившем новый контракт, увеличивая значение в поле «Рейтинг» на 10 и фиксирует транзакцию вставки записи.

Для отладки триггера потребуется вставлять в таблицу «Исполнители» записи, связывающие существующий заказ с сотрудниками различных отделов. Вставку лучше осуществлять с помощью запроса SQL. В этом случае все сообщения, возвращаемые триггером, будут отображаться в консоли сообщений запроса на вставку.

Задания для самостоятельной работы

Вариант 1. База данных ПОСТАВКА ТОВАРОВ

При добавлении нового заказа необходимо проверить, достаточно ли каждого из заказываемых товаров на складе для обеспечения заказа. В случае, если хотя бы одного из заказываемых товаров недостаточно, добавление заказа отменяется.

Вариант 2. База данных РОЗНИЧНАЯ ТОРГОВЛЯ

При оформлении покупки необходимо определить, есть ли на складе требуемый товар. При отсутствии требуемого товара необходимо отменить покупку, в противном случае уменьшить количество соответствующего товара на складе.

Вариант 3. База данных БАНКОВСКИЕ ВКЛАДЫ

При закрытии счёта клиентом необходимо перевести остаток средств на счёте на любой другой счёт этого клиента. Если у клиента счёт был единственным, то оформить снятие остатка денег со счёта.

Вариант 4. База данных ТОРГОВЛЯ

При расформировании отдела товары, которые в нём продавались должны быть переданы в другой отдел. Выбор отдела для передачи осуществить случайным образом.

Вариант 5. База данных АВИАКОМПАНИЯ

При составлении нового рейса необходимо проверить дату последнего техосмотра выбранного самолёта. Если техосмотр был пройден более чем

полгода назад, то необходимо подобрать для выполнения рейса другой свободный самолёт, имеющий количество пассажирских мест большее, чем количество занятых мест в рейсе. Если таких самолётов не будет найдено, то регистрацию нового рейса необходимо отменить.

Список рекомендуемых литературных источников и веб-ресурсов

1. Дейт, К. Введение в системы баз данных / К. Дейт. – 6-издание. – Киев : Диалектика, 1998. – 784 с.
2. Грабер, М. Справочное руководство по SQL / М. Грабер. – Москва : Лори, 1997. – 291 с.
3. Клайн К. SQL. Справочник. 2-е издание. – Москва : КУДИЦ-ОБРАЗ, 2006 – 832 с.
4. Материалы сайта http://office.microsoft.com/ru-ru/access/
5. Материалы сайта www.microsoft.com/sqlserver/
6. Артёмов, Д. В. Microsoft SQL Server 7.0: установка, управление, оптимизация / Д. В. Артёмов, Г. В. Погульский. – Москва : Издательский отдел «Русская редакция», 1998. – 488 с.
7. Бойко, В.В. Проектирование баз данных информационных систем. / В. В. Бойко, В. М. Савинков. – Москва : Финансы и статистика, 1989. – 351 с.
8. Кириллов, В. В. Структурированный язык запросов (SQL) / В. В. Кириллов. – Санкт-Петербург : ИТМО, 1994. – 80 с.
9. Мейер, М. Теория реляционных баз данных / М. Мейер. – Москва : Мир, 1987 – 608 с.
10. Microsoft Access 2002. Русская версия. Шаг за шагом : практ. пособ. / пер. с англ. – Москва : Издательство ЭКОМ, 2002. – 352 с.