**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное учреждения высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«_____»_____2017 г.

# ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ. РАБОТА С ТАБЛИЦАМИ. СОЗДАНИЕ ДИАГРАММЫ.

Методические указания по выполнению лабораторной работы №1
для студентов направления подготовки бакалавриата
10.03.01«Информационная безопасность»

Курск 2017

УДК 621.(076.1)

Составитель: А.Г. Спеваков

Рецензент

Кандидат технических наук, доцент кафедры
«Информационная безопасность» И.В. Калуцкий

**Проектирование базы данных** [Текст] : методические указания по выполнению лабораторной работы / Юго-Зап. Гос. ун-т; сост.: А.Г. Спеваков. – Курск, 2017. – 46с.: ил. 32, табл. 6. – Библиогр.: с. 46.

Содержат сведения по вопросам проектирования баз данных. Указывается порядок выполнения лабораторной работы, правила содержание отчета.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по специальности.

Предназначены для студентов направления подготовки бакалавриата 10.03.01 «Информационная безопасность».

Текст печатается в авторской редакции

Подписано в печать . Формат 60х84 1/16.
Усл.печ. л. 2,67. Уч.-изд. л. 2,42. Тираж 100 экз. Заказ. Бесплатно.
Юго-Западный государственный университет.
305040, г.Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Лабораторный практикум по предмету "Базы данных" состоит из четырех лабораторных работ, итогом которых является разработка полноценного приложения баз данных. Цель практического курса – освоить на простом примере методы и приемы, которые в будущем составят фундамент при разработке более сложных проектов.

Программа ErWin 3.5.2. позволяет сгенерировать скрипт для создания таблиц базы данных для различных SQL серверов. Таким образом, не обязательно знать особенности работы различных серверов, достаточно лишь иметь навыки для работы с ErWin.

Подавляющую массу задач администрирования SQL Server можно выполнить в графической утилите SQL Server Management Studio. В ней можно создавать базы данных и все ассоциированные с ними объекты (таблицы, представления, хранимые процедуры и др.). Здесь вы можете выполнить последовательности инструкций Transact-SQL (запросы). В этой утилите можно выполнить типовые задачи обслуживания баз данных, такие как резервирование и восстановление. Здесь можно настраивать систему безопасности базы данных и сервера, просматривать журнал ошибок и многое другое.

В данной работе мы рассмотрим, как из представления таблиц базы данных в ErWin получить представление в SQLServer 2005, а также, как спроектировать базу данных в графической утилите SQL Server Management Studio.

# ЦЕЛЬ РАБОТЫ

Научиться проектировать базу данных по индивидуальному заданию. Построить уточненную концептуальную модель в виде ER-диаграммы. Разработать текст SQL-запросов для создания базы данных, доменов, таблиц, ограничений целостности и при необходимости других объектов БД.

Получить представление базы данных в среде Microsoft SQL Server 2008 R2. Создать таблицы, диаграмму и связи между таблицами.

## ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.
2. Изучить теоретическую часть.
3. Построить систему функциональных зависимостей.
4. Разработать модель "сущность-связь" (ER-диаграмму).
5. На основании разработанной ER-диаграммы и полученных функциональных зависимостей разработать реляционную модель.
6. Разработать текст SQL-запросов для создания базы данных, доменов, таблиц, ограничений целостности.
7. Получить логическую модель в среде ErWin.
8. Получить физическую модель в среде ErWin.
9. Получить диаграмму БД в среде MS SQL Server 2005.
10. Спроектировать базу данных с помощью SQL Server Management Studio.
11. Создать таблицы с помощью среды SQL Server Management Studio или Transact-SQL.
12. Построить диаграмму полученной БД с помощью SQL Server Management Studio.
13. Создать связи между таблицами полученной БД с помощью SQL Server Management Studio.

## СОДЕРЖАНИЕ ОТЧЕТА

1. Содержание.
2. Индивидуальное задание.
3. Система функциональных зависимостей.
4. Диаграмма "сущность-связь".
5. Структура и поля нормализованных таблиц с указанием всех ключей.
6. Текст SQL-запросов на создание объектов базы данных.
7. Уточненная логическая и физическая модели в среде ErWin 3.5.2.
8. SQL-скрипт с подробными комментариями.
9. Развернутая диаграмма БД в среде MS SQL Server 2005.
10. Спроектированная база данных с помощью SQL Server Management Studio.
11. Созданные таблицы с помощью SQL Server Management Studio.
12. Развернутая диаграмма БД со связями.
13. SQL-скрипт с подробными комментариями (при использовании Transact-SQL).
14. Контрольные вопросы.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Терминология, используемая в области баз данных очень широка и неоднозначна, будем последовательно вводить термины и определения, описывать особенности существующих технологий для всех этапов работы с базами данных.

Предметная область (ПО) - часть реального мира (например: банк, институт, спортклуб, деканат, студенческая группа, предприятие, бухгалтерия, цех и т.д.). Конкретное состояние ПО фиксируется в данных. ПО описывается с какой-то степенью точности, следовательно, она является моделью, т.е. создается модель предметной области (МПО). ПО моделируется при помощи информационных объектов и связей между ними, следовательно, МПО - информационная модель.

Объект - это нечто существующее и различное. Иногда вместо слова "объект" употребляют слово "сущность". Каждый объект имеет уникальное имя.

При идентификации (выделении) ПО теория рекомендует составить описание объекта, т.е. краткое информативное утверждение, которое позволяет установить: является ли реальный предмет экземпляром объекта.

Каждый объект может быть задан (выделен среди других) при помощи наборов свойств. Свойства, которые справедливы для всех возможных экземпляров объекта, т.е. являются общими, называются атрибутами объекта. Т.к. объект может иметь несколько свойств, то он задается набором своих атрибутов.

Элемент данных (ЭД) - то, при помощи чего определяется или задается свойство данных объекта. ЭД имеет уникальное имя и совокупность значений, связанных с этим именем. Это множество называется доменом.

Типичные домены - это множество целых или рациональных чисел, множество символьных строк.

Иногда те значения, которые принимают элементы данных, называются данными, которые хранятся в предметной области.

Пример:

| Объект | Элементы данных | Домен |
|---|---|---|
| Личность | паспорт<br>Ф.И.О.<br>адрес | 38 04 124674<br>Газаев С.Е. |
| Счет | тип<br>номер<br>баланс | расходный<br>367<br>27.8 |

Если речь идет о конкретном объекте, то среди множества его атрибутов существует некоторое его подмножество, значения атрибутов которого однозначно определяют каждый экземпляр данного объекта. Это подмножество называется ключом.

Ключ (ключевой элемент данных) - минимальное множество элементов данных, значение которых однозначно определяет запись об объекте.

Суперключ - любое множество атрибутов, включающее ключ.

Элементы данных, не являющиеся ключами, называются атрибутами. Ключей может быть несколько для одного объекта. Ключ, который выбирают для идентификации записи, называются первичными (основными). Если ключ содержит несколько элементов данных, он называется составным, сцепленным, сложным.

Запись - совокупность значений элементов данных, которые описывают конкретный экземпляр объекта.

Пример:
ЛИЧНОСТЬ=Номер паспорта+Фамилия+Имя+Отчество+Адрес

Если два элемента данных двух конкретных информационных объектов могут использоваться вместе, тогда говорят, что между такими объектами есть связь.

Связь – отношение между элементами данных или объектами, которые могут использоваться совместно в информационной модели. Связь можно выявить в предметной области на основе анализа взаимодействий элементов данных или объектов. Также

она возникает, когда есть соотношение "часть - целое", "класс - подкласс", "вид - подвид".

Свойства баз данных (требования, предъявляемые к файлам в данной предметной области).

1. Неизбыточность в базах данных (минимально необходимая избыточность). Если данные об одном и том же объекте или связи объекта, без остаточных на то оснований, хранятся в нескольких файлах данных, то такая ситуация называется избыточностью. Избыточность приводит к противоречивости данных (например, в одном файле сведения об объекте поменяли, а в другом - забыли). Избыточность в наборах файлов минимальная, если при удалении какого-нибудь файла теряются сведения об объекте данных или о его связях. Файлы данных интегрируют в одну систему. Выделяют в этих файлах одинаковые части.

2. Независимость данных и программ. Все данные хранятся в файлах. Для файла различают логическую и физическую организацию. Программисту необходимо знать структуру записи и порядок их следования. Программа физически не зависит от данных, если она не зависит от того, как реально на диске расположены данные. Под логической независимостью данных и программ подразумеваются два свойства:

- Возможность создания и ведения (модификация, удаление, добавление) файлов данных с помощью универсальной системы управления данных независимо от прикладных программ.
- Изменение в файлах данных (например, добавление еще одного поля данных) не должно приводить к изменению прикладных программ.

3. Взаимосвязь данных и их структурируемость (файлы должны быть формализованы).

4. Способность набора данных и системы данных отвечать на непредвиденные запросы, наращивать (расширять) предметную область без переделки существующих файлов.

Для добавления новых или модификации существующих данных, а также для осуществления поиска данных, применяется общий управляющий способ. Программно-технические средства, реализующие общий управляющий способ и поддерживающие все

свойства (т.е. которая позволяет реализовать) жизненный цикл) БД, называют системой управления БД (СУБД). Конкретная СУБД с конкретной заполненной БД называется банком данных. Иногда сюда включаются прикладные программы. Жизненный цикл БД - это проектирование, заполнение и обслуживание.

Модель Сущность-Связь (ER-модель) (англ. entity-relationship model или entity-relationship diagram) — это модель данных, позволяющая описывать концептуальные схемы. Она предоставляет графическую нотацию, основанную на блоках и соединяющих их линиях, с помощью которых можно описывать объекты и отношения между ними какой-либо другой модели данных. В этом смысле ER-модель является мета-моделью данных, то есть средством описания моделей данных.

ER-модель удобна при прототипировании (проектировании) информационных систем, баз данных, архитектур компьютерных приложений, и других систем (далее, моделей). С её помощью можно выделить ключевые сущности, присутствующие в модели, и обозначить отношения, которые могут устанавливаться между этими сущностями. Важно отметить что сами отношения также являются сущностями (выделяются в отдельные графические блоки), что позволяет устанавливать отношения на множестве самих отношений.

ER-модель является одной из самых простых визуальных моделей данных (графических нотаций). Она позволяет обозначить структуру «крупными мазками», в общих чертах. Это общее описание структуры называется ER-диаграммой или онтологией выбранной предметной области (area of interest).

На этапе перехода к реализации данной ER-диаграммы в виде реальной информационной системы или программы, происходит отображение ER-модели в более детальную модель данных реляционной (объектной, сетевой, логической, или др.) базы данных, которая называется физической моделью данных по отношению к исходной ER-диаграмме.

Существует несколько графических нотаций описания ER-диаграмм (несколько похожих ER-моделей данных). Классический вариант описывается во второй части данной статьи. Есть

несколько типичных примеров использования ER-модели данных: IDEF1x (ICAM DEFinition Language) и dimensional modelling.

Тип данных определяет формат, в котором хранится значение столбца. Стандарт SQL92 предусматривает набор базовых типов данных. Реальные СУБД, как правило, дополняют этот набор своими, специфическими типами данных с целью повышения функциональности системы. В таблице 1 приведены типы данных, поддерживаемые СУБД Microsoft SQL Server 2000, MySQL 3.23, InterBase 6.0.

Рассмотрим типы данных. При выборе типа данных для столбца следует отдавать предпочтение типу, который позволит хранить любые возможные для этого столбца значения и занимать при этом минимальное место на диске. Типы данных в MS SQL Server можно разделить на восемь категорий:

Целочисленные данные

bit (1 байт). Может хранить только значения 0, 1 или null (пустое значение, сообщающее об отсутствии данных). Его удобно использовать в качестве индикатора состояния – включено/выключено, да/нет, истина/ложь.

tinyint (1 байт). Целые значения от 0 до 255.

smallint (2 байта). Диапазон значений от -215 (-32768) до 215 (3767).

int (4 байта). Может содержать целочисленные данные от -231 (-2147483648) до 231 (21474833647).

bigint (8 байт). Включает в себя данные от -263 (9223372036854775808) до 263 (9223372036854775807). Удобен для хранения очень больших чисел, не помещающихся в типе данных int.

Текстовые данные

char. Содержит символьные не Unicode - данные фиксированной длины до 8000 знаков.

varchar. Содержит символьные не Unicode - данные переменной длины до 8000 знаков.

nchar. Содержит данные Unicode фиксированной длины до 4000 символов. Подобно всем типам данных Unicode его удобно

использовать для хранения небольших фрагментов текста, которые будут считываться разноязычными клиентами.

nvarchar. Содержит данные Unicode переменной длины до 4000 символов.

### Десятичные данные

decimal. Содержит числа с фиксированной точностью от -1038-1 до 1038-1. Он использует два параметра: точность и степень. Точностью называется общее количество знаков, хранящееся в поле, а степень – это количество знаков справа от десятичной запятой.

numeric. Это синоним типа данных decimal – они идентичны.

### Денежные типы данных

money (8 байт). Содержит денежные значения от -263 до 263 с десятичной точностью от денежной единицы.

smallmoney (4 байта). Содержит значения от -214748,3648 до 214748,3647 с десятичной точностью.

### Данные с плавающей точкой

float. Содержит числа с плавающей запятой от -1,79E+38 до 1,79E+38.

real. Содержит числа с плавающей запятой от -3,40E+38 до 3,40E+38.

### Типы данных даты и времени

datetime (8 байт). Содержит дату и время в диапазоне от 1 января 1753 года до 31 декабря 9999 года с точностью 3,33 мс.

smalldatetime (4 байта). Содержит дату и время, начиная от 1 января 1900 года и заканчивая 6 июнем 2079, с точностью до 1 минуты.

### Двоичные типы данных

binary. Содержит двоичные данные фиксированной длины до 8000 байт.

varbinary. Содержит двоичные данные переменной длины до 8000 байт.

### Специализированные типы данных

sql_variant. Используется для хранения значения с различными типами данных.

timestamp. Используется для установки временных меток записей при вставке, которые соответствующим образом обновляются.

uniqueidentifier. Глобальный уникальный идентификатор.

xml. Используется для хранения целых документов или фрагментов XML.

Окно Management Studio имеет следующую структуру:

Оконное меню – содержит полный набор команд для управления сервером и выполнения различных операций.

Панель инструментов – содержит кнопки для выполнения наиболее часто производимых операций. Внешний вид данной панели зависит от выполняемой операции.

Панель «Обозреватель объектов». Это панель с древовидной структурой, отображающая все объекты сервера, а также позволяющая производить различные операции, как с самим сервером, так и с его базами данных и их объектами. Обозреватель объектов является основным инструментом для разработки.

Рабочая область. В рабочей области производятся все действия с базой данных, а также отображается её содержимое.

Прежде чем перейти к созданию своих собственных рабочих баз данных рассмотрим служебные базы данных SQL Server, которые создаются автоматически в процессе его установки. Если мы раскроем узел «Базы данных – Системные базы данных» в обозревателе объектов, то увидим следующий набор служебных баз данных:

1. master. Главная служебная база данных всего сервера. В ней хранится общая служебная информация сервера: настройки его работы, список баз данных на сервере с информацией о настройках каждой базы данных и ее файлах, информация об учетных записях пользователей, серверных ролях и т.п.
2. msdb. Эта база данных в основном используется для хранения информации службы SQL Server Agent (пакетных заданий, предупреждений и т.п.), но в нее записывается и

другая служебная информация (например, история резервного копирования).

3. model. Эта база данных является шаблоном для создания новых баз данных в SQL Server. Если внести в нее изменения, например, создать набор таблиц, то эти таблицы будут присутствовать во всех создаваемых базах данных.
4. tempdb. Эта база данных предназначена для временных таблиц и хранимых процедур, создаваемых пользователями и самим SQL Server. Эта база данных создается заново при каждом запуске SQL Server.

Таблицы представляют собой объекты базы данных, используемые непосредственно для хранения всех данных. Одним из самых главных правил организации баз данных является то, что в одной таблице должны храниться данные лишь об одном конкретном типе сущности (например, книги, читатели, движение книги и т. п.).

Данные в таблицах организованы по полям и записям. Поля (или столбцы таблицы) содержат определенный тип информации, например, фамилию, адрес, место работы читателя. Запись (или строка таблицы) - группа связанных полей, содержащих информацию об отдельном экземпляре сущности.

# ВЫПОЛНЕНИЕ РАБОТЫ

## ЗАДАНИЕ НА РАЗРАБОТКУ БАЗЫ ДАННЫХ:

Разработать базу данных предметов инвентаризации, в которой будет храниться информация о кафедрах, ответственных, компьютерах, компьютерных комплектующих. О кафедрах в базе хранится: название кафедры, декан, телефон кафедры. О компьютерах хранится: наименование, описание, прочее. Об ответственных хранится: ФИО, телефон, e-mail, прочее. О комплектующих в базе хранится: тип, имя, серийный номер, описание.

База используется:

-Для инвентаризации материальных ценностей ЮЗГУ.

-Для поиска ответственных лиц за материальные ценности.

-Для формирования отчетов.

Область включает в себя:

1. Кафедры – основное учебно-научное подразделение, осуществляющее учебную, методическую и научно-исследовательскую работу по одной или нескольким родственным дисциплинам, воспитательную работу среди студентов, а также подготовку научно-педагогических кадров, повышение квалификации специалистов.
2. Аудитории – помещения для проведения лекционных, семинарских, практических и иных занятий.
3. Компьютеры – ЭВМ (электронная вычислительная машина) — машина для проведения вычислений, а также приёма, переработки, хранения и выдачи информации по заранее определённому алгоритму (компьютерной программе).
4. Комплектующие - составляющие части компьютеров.
5. Ответственные - люди, отвечающие за состояние и сохранность компьютеров и комплектующих в данной аудитории.

Таблица1 - Объект “Comps”

| Meaning | Designation | Example | Type |
|---|---|---|---|
| По уникальному идентификатору имеем возможность найти объект. | id_comps | 1 | INT, NOT NULL, PRIMARY KEY |
| У каждой аудитории имеется свой номер. | id_audit | 100 | INT, NOT NULL |
| Качественные признаки системного блока. | comps_name | P4/100Gb/1024Mb/LAN | VARCHAR(50), NOT NULL |
| Отличительные признаки. | comps_desc | Notebook | VARCHAR(50), NULL |
| Дополнительная информация. | comps_other | | VARCHAR(255) NULL |

Таблица2 – Объект “Kaf”

| Meaning | Designation | Example | Type |
|---|---|---|---|
| По уникальному идентификатору имеем возможность найти объект. | id_kaf | 1 | INT, NOT NULL, PRIMARY KEY |
| У каждой кафедры имеется свое наименование. | kaf_name | КЗИС | VARCHAR(50), NOT NULL |
| Имя декана кафедры. | kaf_dekan | Лопин В.Н. | VARCHAR(50), NOT NULL |
| Телефон кафедры. | kaf_tel | 44-44-33 | VARCHAR(50), NULL |

Таблица3 - Объект “Otvetst”

| Meaning | Designation | Example | Type |
|---|---|---|---|
| У каждой аудитории имеется свой номер. | id_audit | 1 | INT, NOT NULL, PRIMARY KEY |
| По уникальному | | | |

| | | | |
|---|---|---|---|
| идентификатору имеем возможность найти объект. | id_kaf | 100 | INT, NOT NULL |
| Наименование корпуса. | otvetst_korpus | Главный | VARCHAR(50), NULL |
| Номер аудитории, закрепленной за ответственным лицом. | otvetst_audit | A400 | VARCHAR(50), NOT NULL |
| ФИО ответственного. | otvetst_fio | Петров Петр | VARCHAR(150) NOT NULL |
| Телефонный номер ответственного. | otvetst_tel | 29938 | VARCHAR(50) NULL |
| Email ответственного. | otvetst_email | 1@1.ru | VARCHAR(50), NULL |
| Прочее. | otvetst_other | После 9 не звонить. | VARCHAR(255) NULL |

Таблица4 - Объект “Otvetst”

| Meaning | Designation | Example | Type |
|---|---|---|---|
| У каждой аудитории имеется свой номер. | id_audit | 1 | INT, NOT NULL, PRIMARY KEY |
| По уникальному идентификатору имеем возможность найти объект. | id_kaf | 100 | INT, NOT NULL |
| Наименование корпуса. | otvetst_korpus | Главный | VARCHAR(50), NULL |
| Номер аудитории, закрепленной за | otvetst_audit | A400 | VARCHAR(50), NOT NULL |

| | | | |
|---|---|---|---|
| ответственным лицом. | | | |
| ФИО ответственного. | otvetst_fio | Петров Петр | VARCHAR(150), NOT NULL |
| Телефонный номер ответственного. | otvetst_tel | 29938 | VARCHAR(50) NULL |
| Email ответственного. | otvetst_email | 1@1.ru | VARCHAR(50), NULL |
| Прочее. | otvetst_other | После 9 не звонить. | VARCHAR(255), NULL |

Таблица5 - Объект “Hardware”

| Meaning | Designation | Example | Type |
|---|---|---|---|
| По уникальному идентификатору имеем возможность найти объект. | id_hardware | 1 | INT, NOT NULL, PRIMARY KEY |
| Уникальный идентификатор компьютера. | id_comps | 100 | INT, NOT NULL |
| Инвентаризационный номер | id_invent | NZ324 | VARCHAR(25), NOT NULL |
| Отличительные признаки. | comps_desc | Notebook | VARCHAR(50), NULL |
| Дополнительная информация. | hardware_type | 3 | SMALLINT, NOT NULL |
| Наименование комплектующих. | hardware_name | BENQ 17" | VARCHAR(100) NOT NULL |
| Серийный номер. | hardware_name | Ahtgur6t875 | VARCHAR(50), NULL |
| Описание | hardware_desc | Без | VARCHAR(200) |

| | | | |
|---|---|---|---|
| | | Сидирома. | NULL |

Таблица6 - Объект “Otvetst”

| Meaning | Designation | Example | Type |
|---|---|---|---|
| У каждой аудитории имеется свой номер. | id_audit | 1 | INT, NOT NULL, PRIMARY KEY |
| По уникальному идентификатору имеем возможность найти объект. | id_kaf | 100 | INT, NOT NULL |
| Наименование корпуса. | otvetst_korpus | Главный | VARCHAR(50), NULL |
| Номер аудитории, закрепленной за ответственным лицом. | otvetst_audit | А400 | VARCHAR(50), NOT NULL |
| ФИО ответственного. | otvetst_fio | Петров Петр | VARCHAR(150), NOT NULL |
| Телефонный номер ответственного. | otvetst_tel | 29938 | VARCHAR(50) NULL |
| Email ответственного. | otvetst_email | 1@1.ru | VARCHAR(50), NULL |
| Прочее. | otvetst_other | После 9 не звонить. | VARCHAR(255), NULL |

## СИСТЕМА ФУНКЦИОНАЛЬНЫХ ЗАВИСИМОСТЕЙ

Функциональная зависимость (functional dependency) – такая логическая связь между атрибутами отношения, при которой по известному значению одного атрибута можно найти (или вычислить) значение другого атрибута.

## Нормальные формы

Здесь изложены несколько правил, относящихся к нормализации. Все эти правила представляют собой частные случаи только что описанного процесса нормализации.

В 70-х годах XX века теоретики реляционных баз данных обнаруживали различные типы аномалий модификации, вызванные структурой отношений. Классы отношений, лишенные аномалий определенного типа, называются нормальными формами (normal forms). Известно семь нормальных форм: первая, вторая, третья, четвертая, пятая нормальные формы (1НФ, 2НФ, 3НФ, 4НФ, 5НФ), нормальная форма Бойса-Кодда (НФБК) и доменно-ключевая нормальная форма (ДКНФ). Нормальные формы являются вложенными друг в друга (рис.11). То есть отношение во второй нормальной форме является отношением в первой нормальной форме, а отношение в 5НФ одновременно находится в 4НФ, НФБК, 3НФ, 2НФ, 1НФ.

### Первая нормальная форма (first normal form) – 1НФ

Таблица находится в 1НФ, если она удовлетворяет определению отношения. Таблица, находящаяся в 1НФ может быть подвержена аномалиям. Более старшие нормальные формы позволяют избежать определенных типов аномалий.

### Вторая нормальная форма (second normal form) – 2НФ

Известно, что по значению первичного ключа можно однозначно определить значения остальных ячеек этой строки. Следовательно, все неключевые атрибуты функционально зависят от первичного ключа.

Определение: таблица находится во второй нормальной форме (2НФ), если она удовлетворяет определению 1НФ и все ее атрибуты, не входящие в первичный ключ, функционально зависят от первичного ключа и не зависят от части первичного ключа. Отсюда следует вывод: все таблицы с простым первичным ключом находятся во 2НФ.

### Третья нормальная форма (third normal form) – 3НФ

Определение: таблица находится в третьей нормальной форме (3НФ), если она удовлетворяет определению 2НФ и не существует функциональных зависимостей между не ключевыми атрибутами.

Нормальная форма Бойса-Кодда (Boyce-Codd normal form, BC/NF) – НФБК

Определение: таблица находится в нормальной форме Бойса-Кодда (НФБК), если не ключевые атрибуты функционально зависят только от возможных ключей, и не зависят от частей этих потенциальных ключей.

Четвертая нормальная форма (fourth normal form) – 4НФ

В отношениях возможны другие виды аномалий, связанные с наличием многозначных зависимостей (multivalue dependency) между атрибутами. По определению, атрибут А многозначно определяет атрибут В той же таблицы, если для каждого значения атрибута А существует хорошо определенное множество соответствующих значений В.

Пятая нормальная форма (fifth normal form) – 5НФ

Пятая нормальная форма затрагивает отношения, которые имеют несколько многозначных атрибутов, и эти атрибуты зависимы между собой.

Доменно-ключевая нормальная форма (domain/key normal form) – ДКНФ.

Определение: отношение находится в доменно-ключевой нормальной форме, если каждое ограничение целостности, накладываемое на это отношение, является логическим следствием определения доменов и ключей.

Доказано, что таблицы, находящиеся в ДКНФ, лишены каких бы то ни было аномалий модификации. К сожалению, общего подхода, позволяющего привести таблицу к ДКНФ, пока не существует.

При составлении модели "сущность-связь", а затем реляционной модели данных, следует планировать данные так, чтобы каждая таблица содержала ровно одну тему. Это поможет избежать аномалий в таблицах.

Далее каждую таблицу необходимо проверить на соответствие нормальным формам в следующем порядке (и при необходимости разбить на более мелкие таблицы):

1НФ -> 2НФ -> НФБК -> 4НФ -> 5НФ -> ДКНФ

Нормализация таблиц имеет свои плюсы и минусы. Существенным плюсом является то, что пропадают аномалии модификации, избыточность, хранение противоречивой информации. Минус нормализации проявляется в замедленной выборке данных. После нормализации количество таблиц возрастает; информация, которая раньше лежала в одной таблице, теперь разбросана по нескольким. Чтобы составить комплексный отчет, приходится просматривать несколько таблиц, что занимает больше времени, чем поиск в одной ненормализованной таблице. В некоторых базах данных это замедление поиска оказывается столь существенным, что выгоднее пренебречь нормализацией, зато выиграть в производительности. Тогда выполняют обратный процесс – денормализацию – намеренное соединение нормализованных таблиц в не нормализованные. Логику работы прикладных программ, обрабатывающих базу данных, дополняют процедурами дополнительной поддержки целостности и непротиворечивости ненормализованных данных.

Пример СИСТЕМЫ ФУНКЦИОНАЛЬНЫХ ЗАВИСИМОСТЕЙ:

(1) Объяснить почему был выбран первичный ключ.

(2) F1- Описание таблицы, первым выделяются первичные ключевые атрибуты таблицы, после знака -> идёт описание остальных атрибутов.

(3) K1- Только первичные ключи.

«otvetst»:

(1)Не может быть двух ответственных за одну аудиторию.

(2)F1: id_audit-> id_kaf, otvetst_korpus, otvetst_audit, otvetst_fio, otvetst_tel, otvetst_email, otvetst_other

(3)K1 = id_audit.

«kaf»:

Не может быть двух кафедр с одинаковыми идентификаторами.

F2: id_kaf -> kaf_name, kaf_dekan, kaf_tel

K2 = id_kaf

«hardware»:

Не может существовать несколько единиц комплектующих с одинаковым уникальным идентификатором.

F3: id_hardware -> id_comps, id_invent, hardware_type, hardware_name, hardware_sn, hardware_desc

K3 = id_hardware

«comps»:

Не может существовать несколько компьютеров с одинаковыми уникальными идентификаторами.

F4: id_comps -> id_audit, comps_name, comps_description, comps_other

K4 = id_comps

Рассмотрим концептуальную модель (ER-диаграмма). С помощью программы PLATINUM ERwin ERX 3.5.2 составляем ER-диаграмму ("сущность-связь" Entity-Relationship) нашей базы данных.

1.Открываем PLATINUM ERwin ERX 3.5.2

2.Настраиваем программу-> Вкладка Server-> Выбираем Target Server-> устанавливаем SQL Server и версию 7.x.

3.Для того чтобы создать таблицу выбираем вкладку Windows-> нажимаем Erwin Toolbox или ctrl+t, появится инструментарий Erwin-> выбираем Independent Table и щёлкаем на рабочем столе программы, создаётся таблица и переходим к шагу заполнения значениями.

4.Даём название таблице Sportsmens -> далее начинаем заполнять таблицу значениями-> щёлкнув на ней два раза появится зона в которой можно редактировать атрибуты нашей таблицы-> в вкладке General выбираем тип атрибута (Default, Blob, Datetime, Number, String) переходим к вкладке SQL Server, выбираем нужный нам тип атрибута и задаём его длину.

5.Далее начинаем расставлять связи в нашей диаграмме-> в инструментарии выбираем соответствующую связь и связываем таблицы.

Текст SQL-запросов на создание объектов базы данных.

```
CREATE DATABASE inv
USE inv
CREATE TABLE [dbo].[kaf](
[id_kaf] [int] IDENTITY(1,1) NOT NULL,
```

```
[kaf_name] [varchar](50) COLLATE Cyrillic_General_CI_AS
NOT NULL,
[kaf_dekan] [varchar](50) COLLATE Cyrillic_General_CI_AS
NULL,
[kaf_tel] [varchar](50) COLLATE Cyrillic_General_CI_AS
NULL,
CONSTRAINT [PK_kaf] PRIMARY KEY CLUSTERED    )
GO
CREATE TABLE [dbo].[otvetst](
[id_audit] [int] IDENTITY(1,1) NOT NULL,
[id_kaf] [int] NOT NULL,
[otvetst_korpus] [varchar](50) COLLATE
Cyrillic_General_CI_AS NULL,
[otvetst_audit] [varchar](50) COLLATE Cyrillic_General_CI_AS
NOT NULL,
[otvetst_fio] [varchar](150) COLLATE Cyrillic_General_CI_AS
NOT NULL,
[otvetst_tel] [varchar](50) COLLATE Cyrillic_General_CI_AS
NULL,
[otvetst_email] [varchar](50) COLLATE Cyrillic_General_CI_AS
NULL,
[otvetst_other] [varchar](255) COLLATE
Cyrillic_General_CI_AS NULL,
CONSTRAINT [PK_otvetst] PRIMARY KEY CLUSTERED )
GO
CREATE TABLE [dbo].[hardware](
[id_hardware] [int] IDENTITY(1,1) NOT NULL,
[id_comps] [int] NOT NULL,
[id_invent] [varchar](25) COLLATE Cyrillic_General_CI_AS
NOT NULL,
[hardware_type] [smallint] NOT NULL,
[hardware_name] [varchar](100) COLLATE
Cyrillic_General_CI_AS NOT NULL,
[hardware_sn] [varchar](50) COLLATE Cyrillic_General_CI_AS
NULL,
[hardware_desc] [varchar](200) COLLATE
Cyrillic_General_CI_AS NULL)
```

```
GO
CREATE TABLE [dbo]. [comps](
[id_comps] [int] IDENTITY (1,1) NOT NULL,
[id_audit] [int] NOT NULL,
[comps_name] [varchar](50) COLLATE Cyrillic_General_CI_AS
NOT NULL,
[comps_description] [varchar](50) COLLATE
Cyrillic_General_CI_AS NULL,
[comps_other] [varchar](255) COLLATE
Cyrillic_General_CI_AS NULL,
CONSTRAINT [PK_comps] PRIMARY KEY CLUSTERED )
```

Откройте вашу Erwin-диаграмму. В появившемся окне после выбора пункта главного меню File -> New задать раздел Blank Diagram.

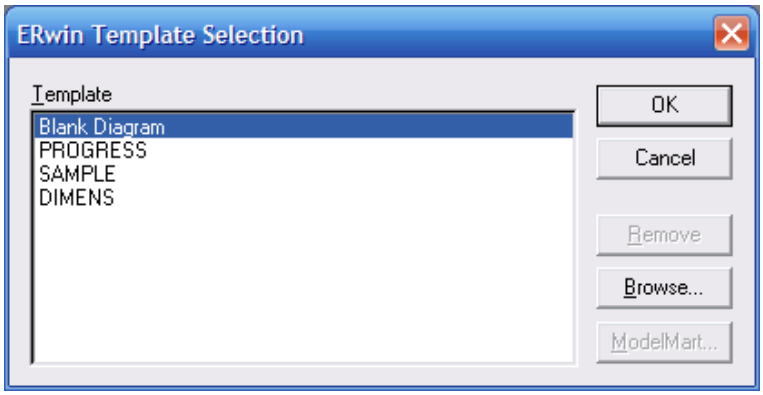



Рис. 1 - Окно создания новой диаграммы

Для отображения символов русского языка, в появившемся диалоговом окне задайте по умолчанию шрифт, поддерживающий символы кириллицы (@Arial Unicode MS) как показано на Рис. 2. Применить данный шрифт для всех объектов (Apply settings to All objects).

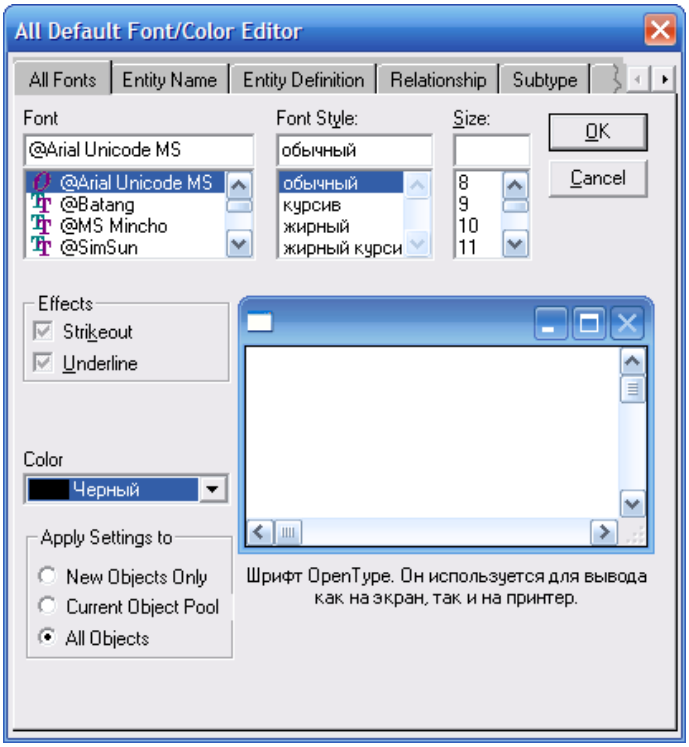



Рис. 2 - Окно настройки шрифта

Задать отображение физической модели и выбрать сервер-назначение MS SQL Server 7

В итоге получаем примерно такую физическую модель базы данных.

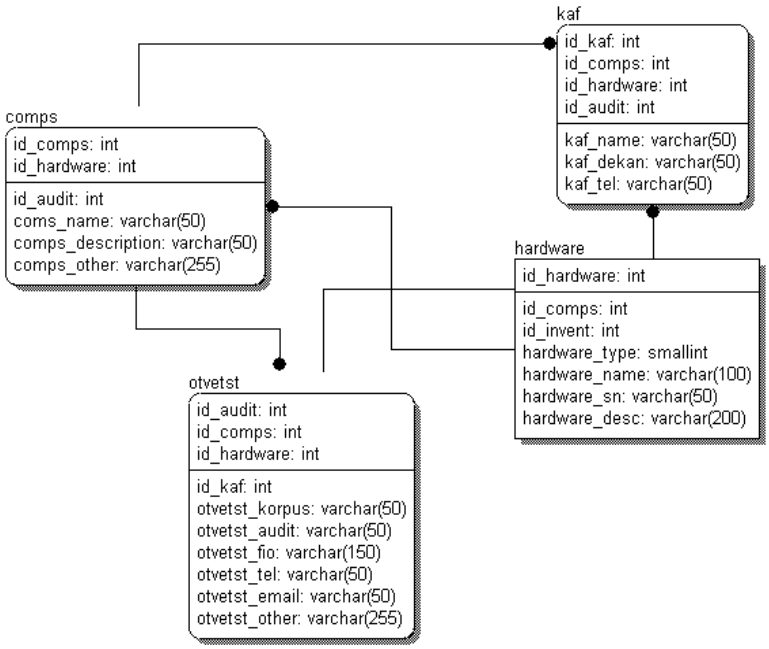



Рис. 3 - Физическая модель БД

Выбираем меню типа сервера (SQL Server 7x) (Рис. 4)

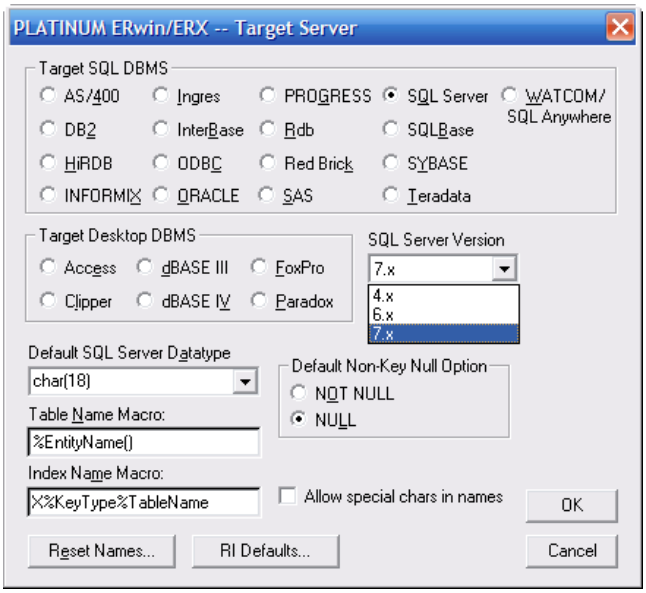
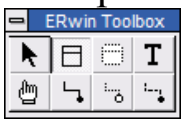



Рис. 4 - Окно выбора типа SQL Server

Задаем сущности. Для этого на панели инструментов выбираем второй инструмент.




После двойного нажатия на добавленной сущности появится окно, показанное на Рис. 5. Для добавления атрибута нажмите кнопку New.

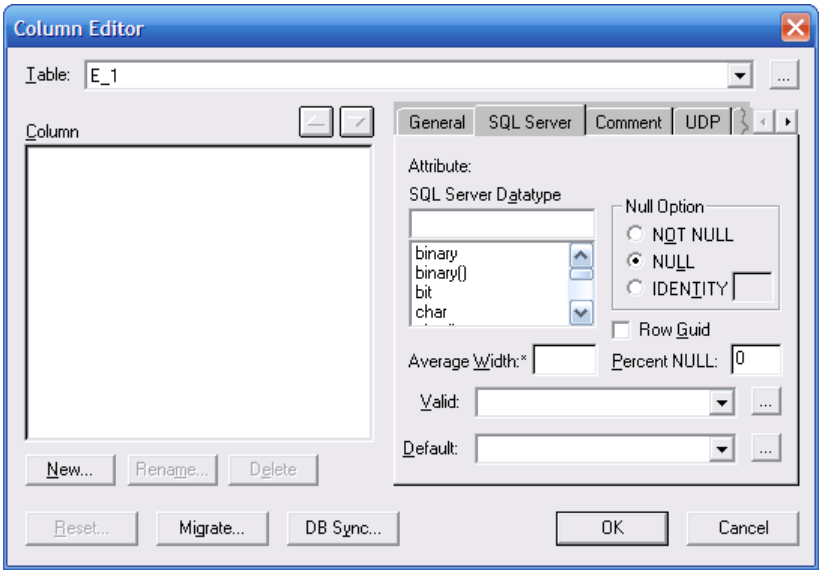



Рис. 5 - Окно добавления атрибута

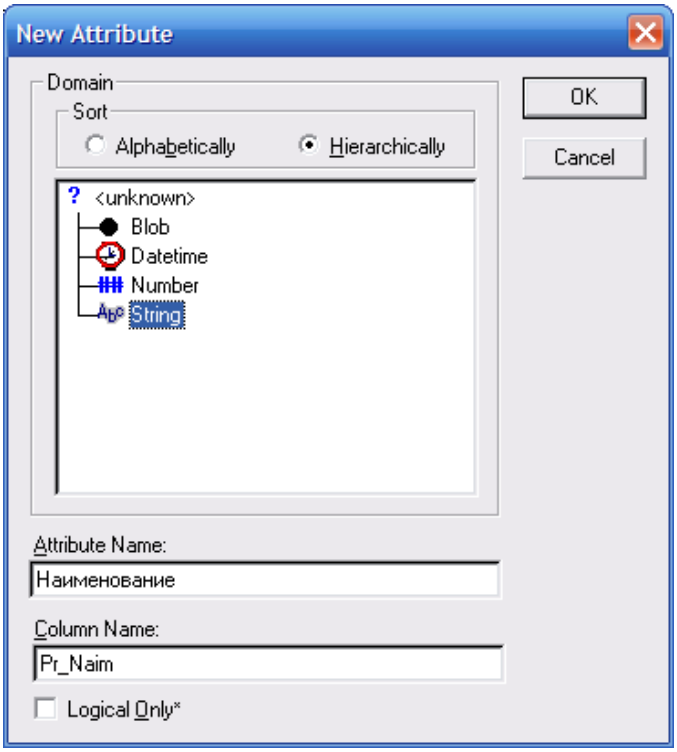



Рис. 6 - Окно свойств атрибута

В появившемся окне необходимо задать тип атрибута, его логическое и физическое имена. Логическое имя может содержать произвольные символы. Физическое представление должно начинаться с латинского символа и содержать латинские буквы, цифры и знаки подчеркивания. Желательно не использовать длинные физическое имена (более 12 символов). (Рис. 6)


В результате добавления атрибутов логическое и физическое представления сущности:

В верхней части таблицы располагаются ключевые поля. Для того, чтобы атрибут стал первичным ключом, необходимо в редакторе атрибутов (Attribute Editor) на закладке General установить свойство Primary Key.

Следующим этапом является переопределение типов (в случае если типы не указаны точно при изначальном задании

атрибутов) в соответствии с набором типов сервера-назначения. Также здесь указываются значения по умолчанию, множество допустимых значений атрибута и признак обязательного заполнения. (Рис. 7)

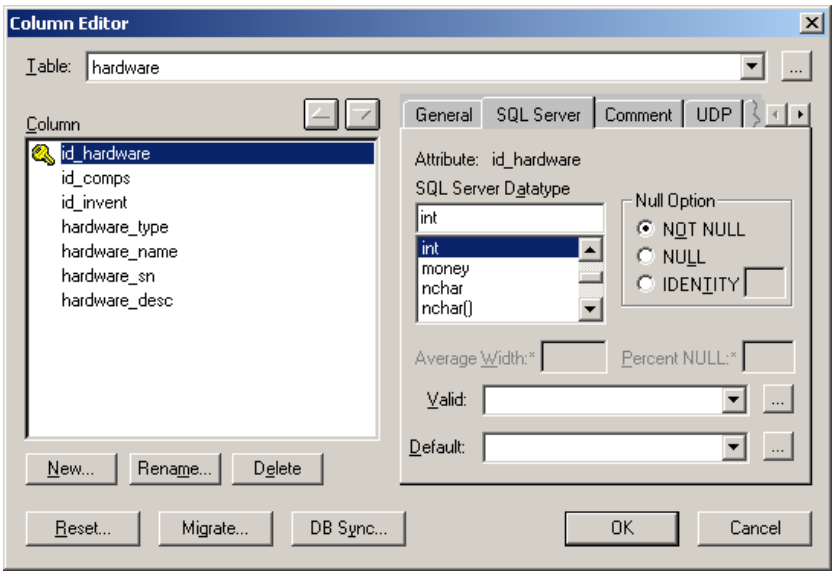



Рис. 7 - Окно переопределения типов

Valid – Правило, определяющее множество значений атрибута (используется в случае задания домена)

Default – Значение по умолчанию.

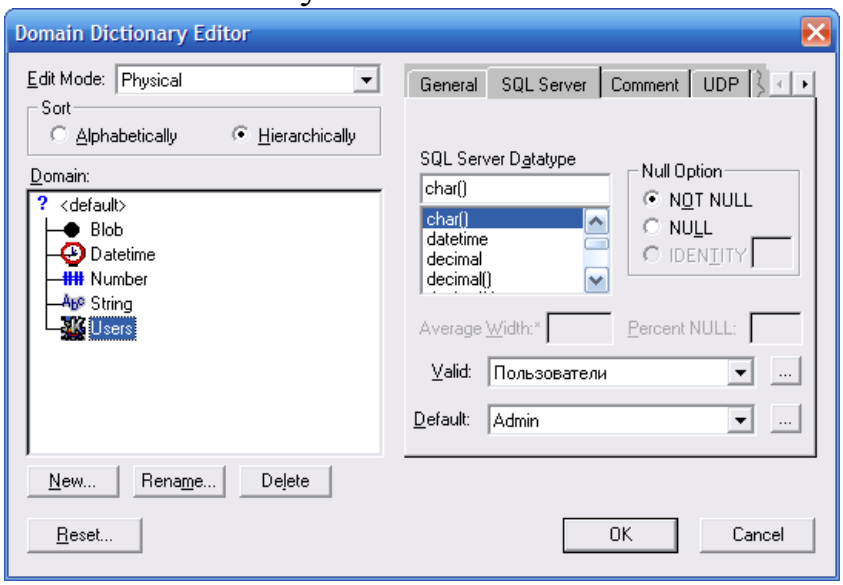



Рис. 8 - Выбор типа данных

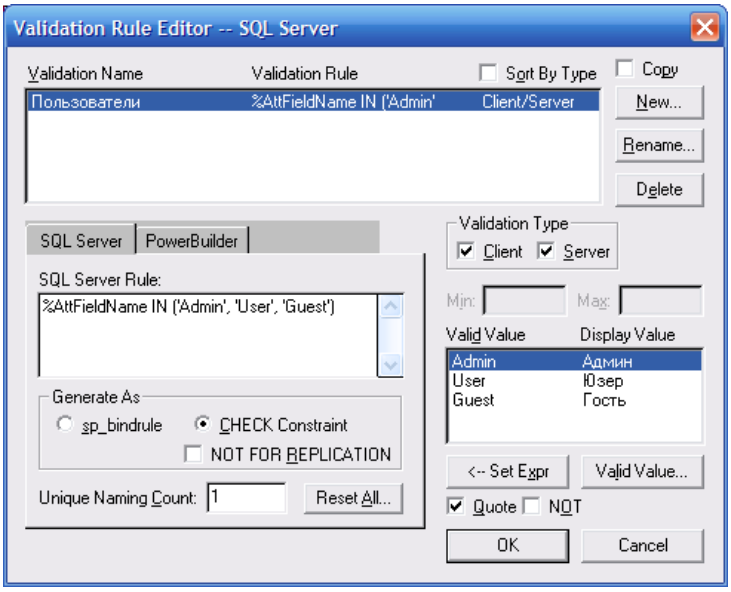



Рис. 9 - Назначение пользователей

После того, как все сущности заданы, установите связи между ними.

Для представления физической модели на языке SQL воспользуйтесь функцией Forward Engineer.

В появившемся окне выставите нужные опции (вид, таблицы, индексные поля, ограничения целостности данных, триггеры и др.) и проверьте схему на закладке Summary. (Рис. 10)

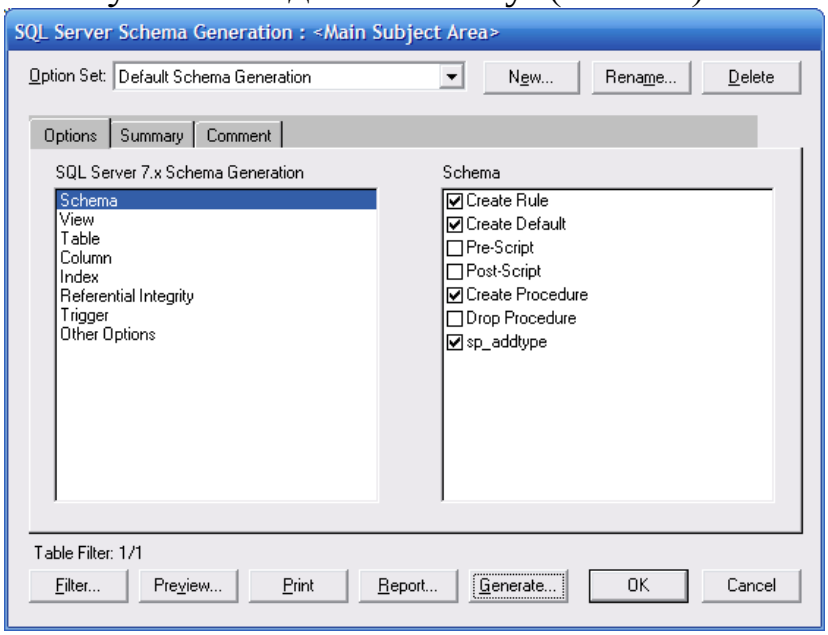



Рис. 10 - Окно настройки схемы.

Нажмите кнопку Preview и скопируйте сформированный SQL-скрипт:

CREATE TABLE kaf(

```
    id_kaf int IDENTITY(1,1) NOT NULL,
    kaf_name varchar(50) NOT NULL,
    kaf_dekan varchar(50) NULL,
    kaf_tel varchar(50) NULL,
CONSTRAINT PK_kaf PRIMARY KEY CLUSTERED)
GO
CREATE TABLE otvetst(
    id_audit int IDENTITY(1,1) NOT NULL,
    id_kaf int NOT NULL,
    otvetst_korpus varchar(50) NULL,
    otvetst_audit varchar(50) NOT NULL,
    otvetst_fio varchar(150) NOT NULL,
    otvetst_tel varchar(50) NULL,
    otvetst_email varchar(50) NULL,
    otvetst_other varchar(255) NULL,
CONSTRAINT PK_otvetst PRIMARY KEY CLUSTERED)
GO
CREATE TABLE hardware(
    id_hardware int IDENTITY(1,1) NOT NULL,
    id_comps int NOT NULL,
    id_invent varchar(25) NOT NULL,
    hardware_type smallint NOT NULL,
    hardware_name varchar(100) NOT NULL,
    hardware_sn varchar(50) NULL,
    hardware_desc varchar(200) NULL)
GO
CREATE TABLE comps(
    id_comps int IDENTITY(1,1) NOT NULL,
    id_audit int NOT NULL,
    comps_name varchar(50) NOT NULL,
    comps_description varchar(50) NULL,
    comps_other varchar(255) NULL,
CONSTRAINT PK_comps PRIMARY KEY CLUSTERED)
```

Запустите приложение SQL Server Management Studio Express. В меню дерева сервера выберите пункт “New Database” (Рис. 11)

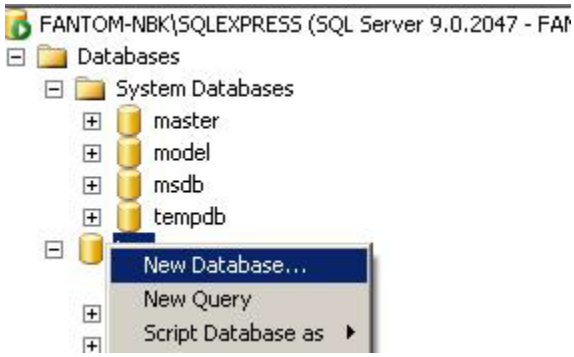


Рис. 11 - Обозреватель объектов sql

В появившемся окне введите имя новой БД.

После сообщения об успешном создании БД выполните SQL-скрипт, полученный в результате выгрузки модели из среды ErWin.

Примечание: Первой командой, предшествующей полученному запросу необходимо использовать USE (имя базы данных).

После успешного выполнения запроса (при возникновении ошибок и неточностей внести необходимые коррективы) переходим к созданию диаграммы БД в SQL Server Enterprise Manager.

Связи должны отобразиться автоматически (в противном случае допущена ошибка на более раннем этапе – вернитесь на более ранний этап выполнения работы!).

Переходим к созданию диаграммы БД, для этого в SQL Server Management Studio Express выбираем базу данных `inv` -> Database Diagram. (Рис. 12)

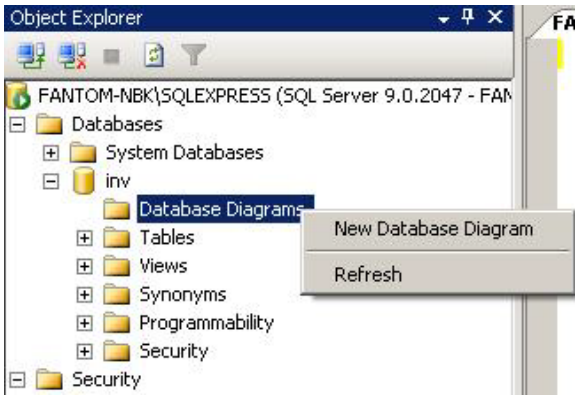


Рис. 12 - Создание диаграммы БД

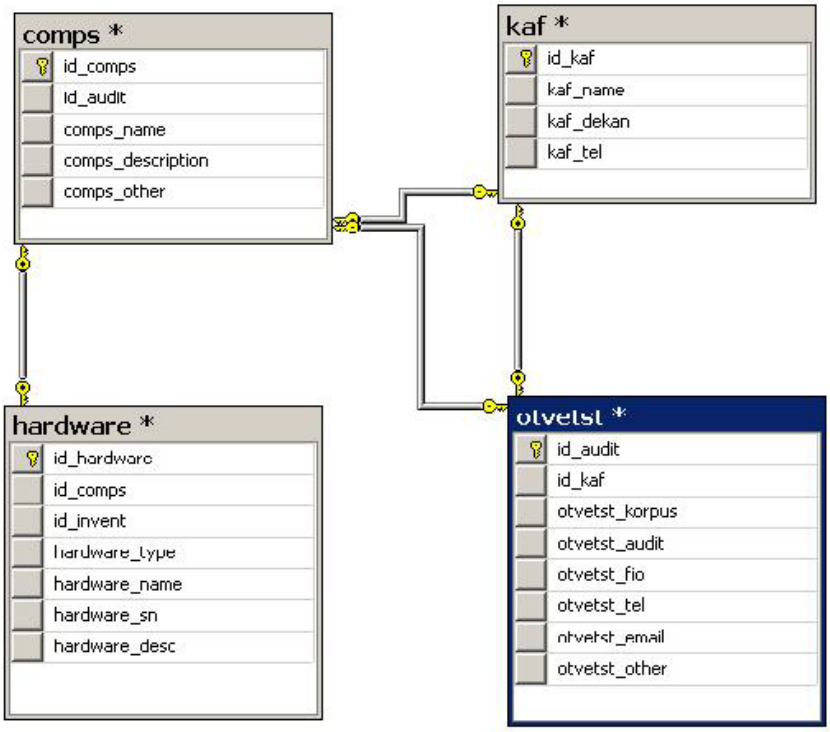



Рис. 13 - Диаграмма полученной БД

Рассмотрим второй способ проектирования базы данных с помощью SQL Server Management Studio.

ЗАДАНИЕ НА РАЗРАБОТКУ БАЗЫ ДАННЫХ:

Разработать электронный каталог библиотеки с поисковой системой. Существует алфавитный и систематический каталог, при этом одна книга может соответствовать нескольким разделам систематического каталога. Для каждой книги хранятся выходные данные: шифр, авторы, название, место и год издания, число страниц, количество экземпляров и их инвентарные номера, местонахождение каждого экземпляра (читальный зал или абонемент). Поиск книг может проводиться по разделу систематического каталога, шифру, автору и названию (можно по маске). Также в базе данных учитываются все факты выдачи книг читателям. В результатах поиска должно отражаться, сколько книг в настоящий момент находится в фондах библиотеки, и сколько выдано на руки. Для библиотекарей предусмотреть вывод списка книг, к которым не было обращений в течение последних пяти лет.

Запускаем «SQL Server Management Studio» («Пуск» — «Microsoft SQL Server 2008 R2» — «Среда SQL Server Management Studio»)

В открывшемся окне выбираем:

1. Тип сервера. Здесь следует выбрать, к какой именно службе необходимо подключится. Оставьте вариант «Компонент Database Engine».
2. Имя сервера. Позволяет указать, к какому серверу будет осуществляться подключение. По умолчанию имя SQL Server совпадает с именем компьютера. Выберите ваш локальный компьютер.
3. Проверка подлинности. Способ аутентификации, можно выбрать «Проверка подлинности Windows» или «Проверка подлинности SQL Server». Первый способ использует учетную запись, под которой текущий пользователь осуществил вход в Windows. Вариант SQL Server использует свою собственную систему безопасности. Оставьте вариант проверки подлинности Windows.
4. После чего нажимаем «Соединить» (Рис. 14)

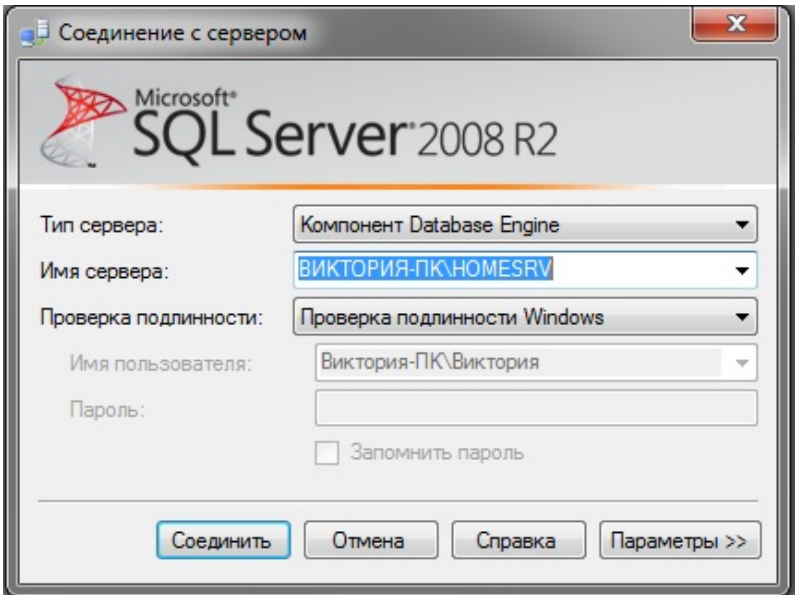



Рис. 14 - Окно подключения к «Среда SQL Server Management Studio»

Если все введено верно, в окне «Обозреватель объектов» мы увидим вкладку с именем нашего SQL-сервера.

Для добавления новой БД, в «Среде Microsoft SQL Server Management Studio» кликаем правой кнопкой мышки на вкладке «Базы данных» и выбираем «Создать базу данных» (Рис. 15)

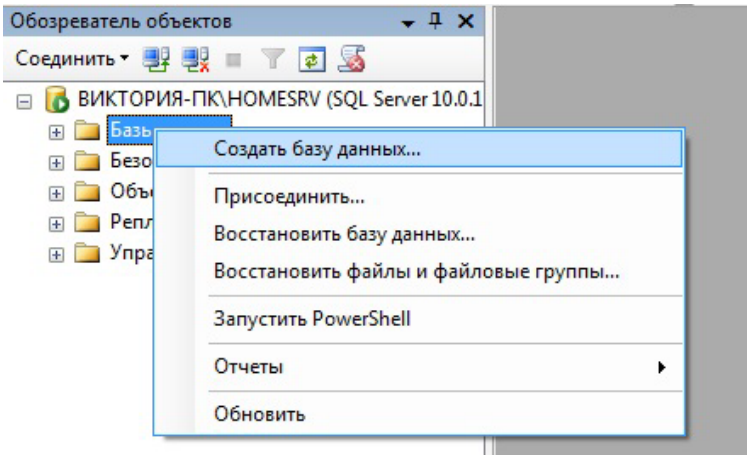



Рис. 15 - Создание новой БД

В открывшемся окне «Создание базы данных» на вкладке «Общие» заполняем:

1. Задаем имя базы данных. Имя базы данных не должно начинаться с цифры или иметь пробелы в названии, иначе получим ошибку: «неправильный синтаксис около конструкции %имя базы данных%»
2. Владельца оставляем по умолчанию. (Рис. 16)

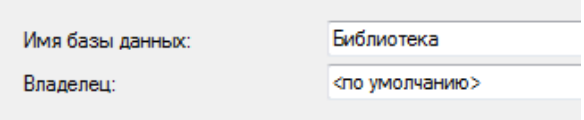



Рис. 16 - Окно свойств БД

После чего в списке мы должны увидеть только что созданную базу данных. (Рис. 17)

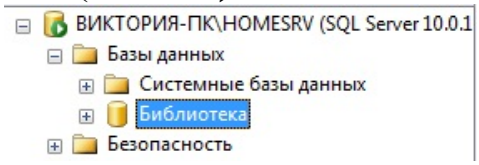



Рис. 17 - БД в обозревателе компонентов

Теперь переходим непосредственно к созданию самих таблиц. Предусмотрена возможность создавать новые таблицы, присваивать им имена и добавлять к существующим базам данных

в SQL Server 2008, используя «Среда SQL Server Management Studio» и Transact-SQL.

Использование среды SQL Server Management Studio

1. В обозревателе объектов щелкните правой кнопкой мыши узел Таблицы базы данных и выберите Создать таблицу. (Рис. 18)

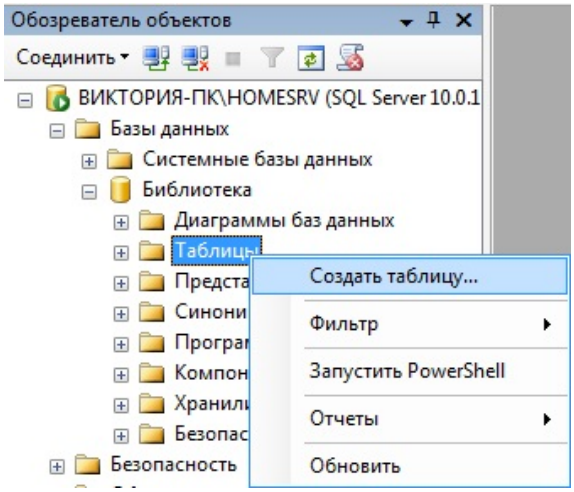



Рис. 18 - Порядок создание таблиц

2. Введите имена столбцов, выберите типы данных и определите для каждого столбца, могут ли в нем присутствовать значения NULL. (Рис. 19)

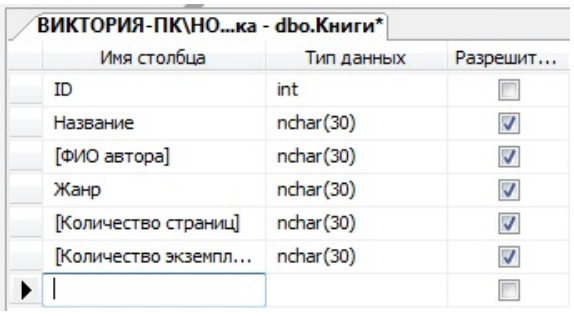



Рис. 19. Создание столбцов и их свойства

3. Чтобы указать, что столбец является столбцом первичного ключа, щелкните его правой кнопкой мыши и выберите Задать первичный ключ. В нашем случае ID - первичный ключ. (Рис. 20)

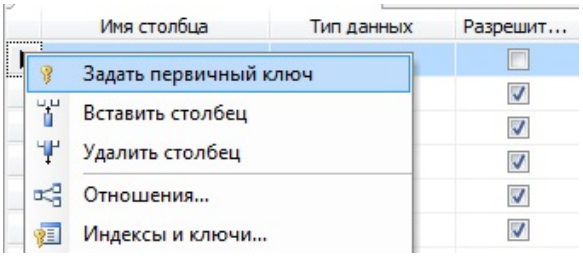



Рис. 20 - Задание первичного ключа

4. Щелкните правой кнопкой мыши по вашей таблице и выберите Сохранить table_1.

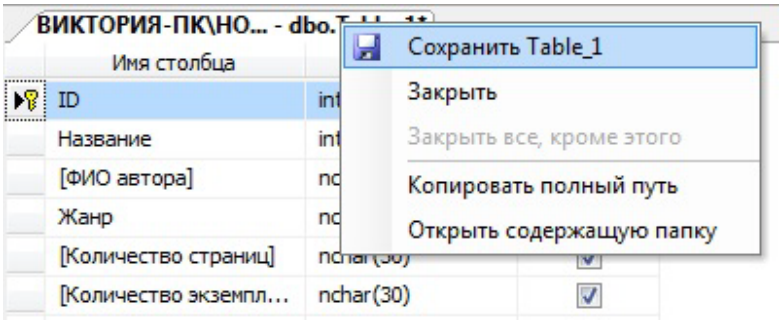



Рис. 21 - Сохранение таблицы

5. В диалоговом окне Выбор имени введите имя таблицы - Книги и нажмите кнопку OK. (Рис. 22)

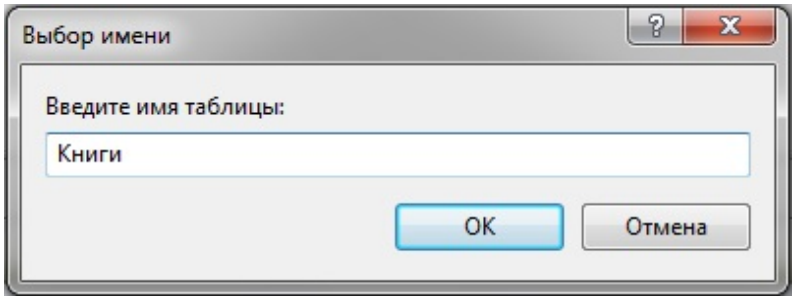



Рис. 22 - Окно Выбор имени таблицы

6. Чтобы просмотреть новую таблицу, в обозревателе объектов разверните узел Таблицы, а затем нажмите клавишу F5, чтобы обновить список объектов. Новая таблица будет отображена в списке таблиц. (Рис. 23)

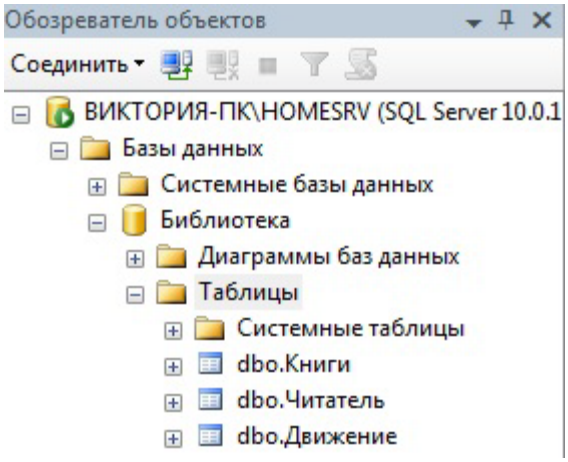



Рис. 23 - Отображение таблиц в обозревателе компонентов

Создадим еще несколько таблиц. Таким образом, наша база данных Библиотека имеет три таблицы: Книги, Читатель, Движение.

Теперь построим диаграмму базы данных и создадим связи между нашими таблицами.

1. В обозревателе объектов щелкните правой кнопкой мыши узел Диаграммы базы данных и выберите Создать диаграмму (Рис.24)

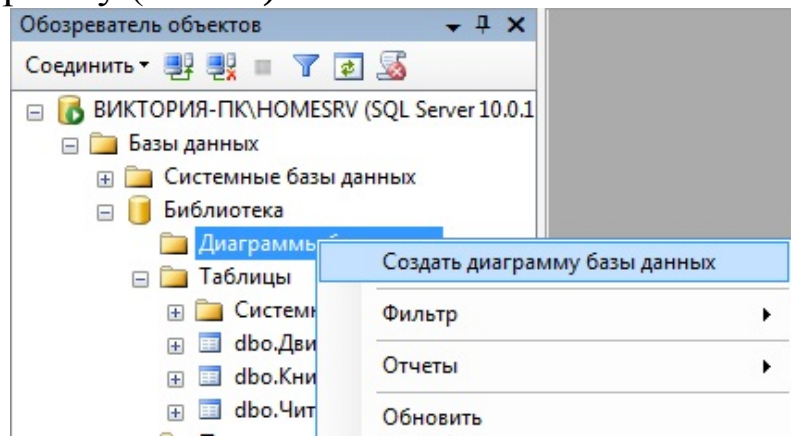



Рис. 24 - Порядок создания диаграммы

2. В диалоговом окне выберите нужные таблицы и нажмите ОК.
3. Проводим нужные нам связи:
    1. Протягиваем первичный ключ ID таблицы Книги к [Название книги] таблицы Движение.
    2. В диалоговом окне проверяем правильность соединения столбцов и нажимаем ОК (Рис. 25)

Важно, чтобы тип данных связанных столбцов совпадал.

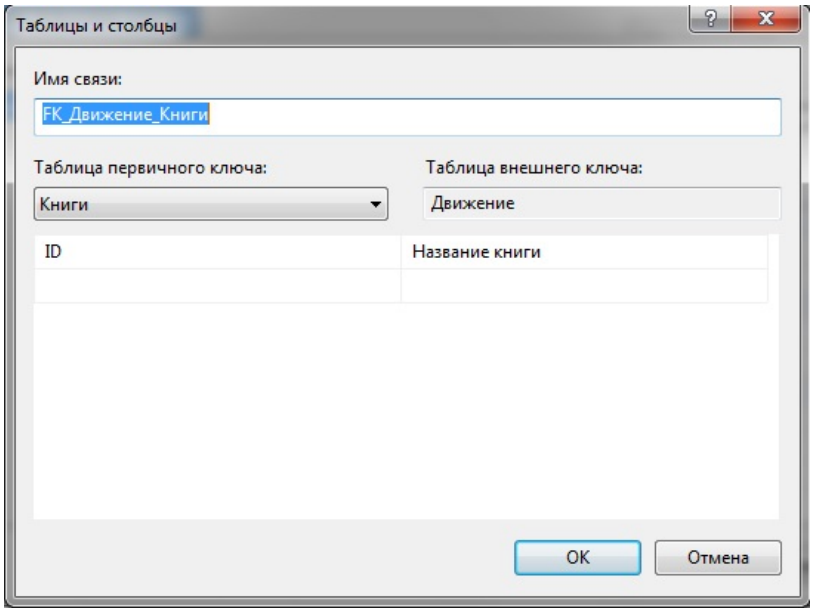



Рис. 25 - Связь между таблицами

Проделываем то же с таблицами Движение и Читатель (Рис. 26)

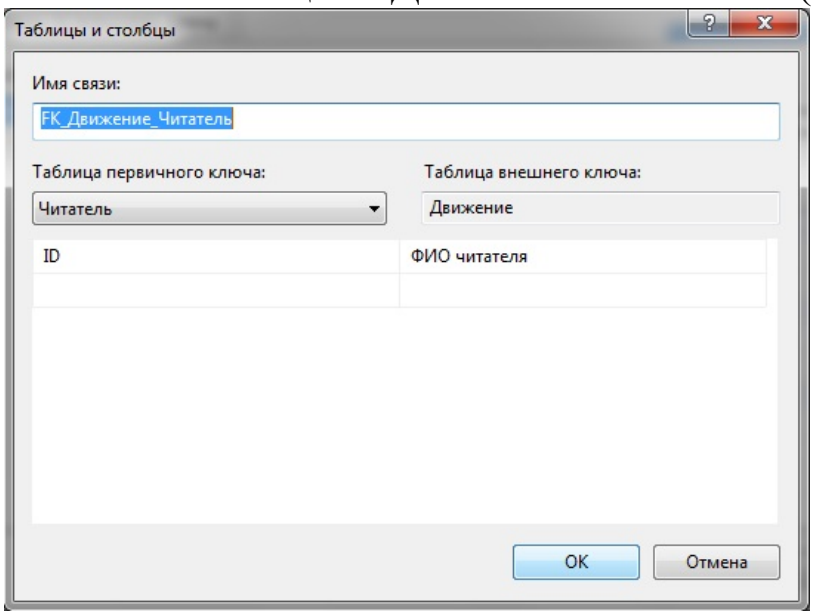



Рис. 26 - Связь между таблицами

Таким образом диаграмма примет следующий вид (Рис. 27)

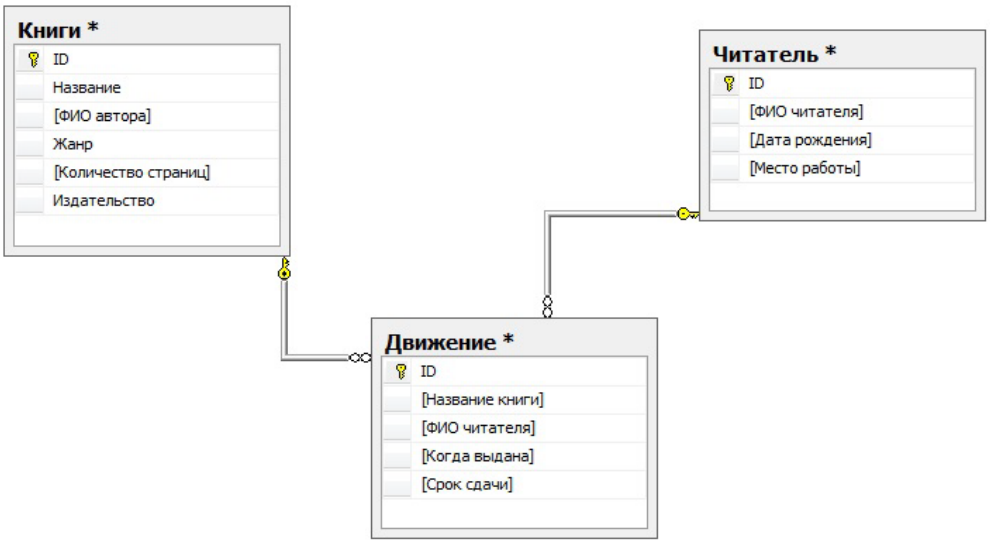



Рис. 27 - Диаграмма базы данных

Чтобы сохранить диаграмму нажмите правой кнопкой по названию и выберите Сохранить диаграмму. (Рис. 28)

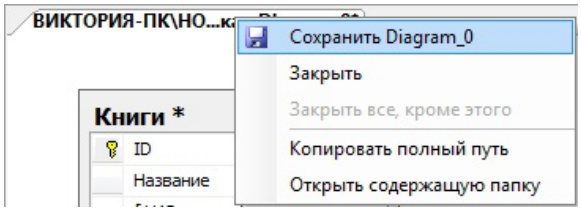



Рис. 28 - Окно сохранения диаграммы БД

В диалоговом окне введите название диаграммы и нажмите ОК (Рис. 29)

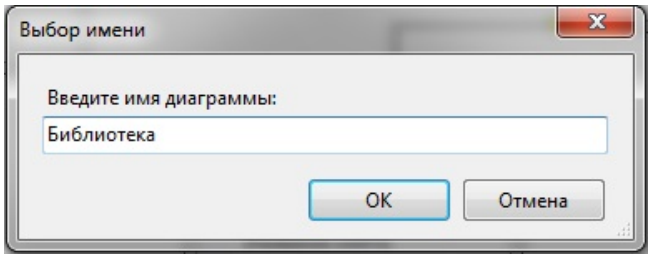



Рис. 29 - Окно «Выбор имени диаграммы»

Теперь ваша диаграмма отображается в обозревателе объекта. (Рис. 30)

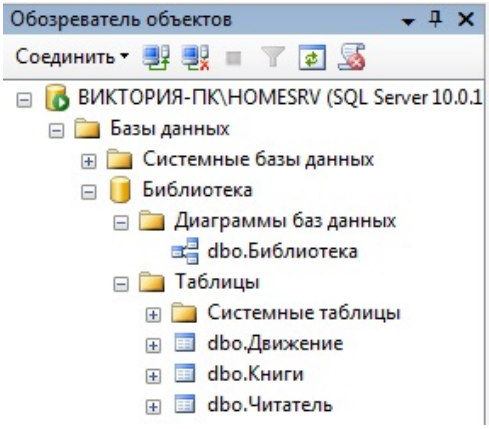



Рис. 30. Отображение диаграммы в обозревателе компонентов

Переходим к созданию базы данных с помощью Transact-SQL.

1. В обозревателе объектов щелкните правой кнопкой мыши Библиотека и выберите Создать запрос. (Рис. 31)

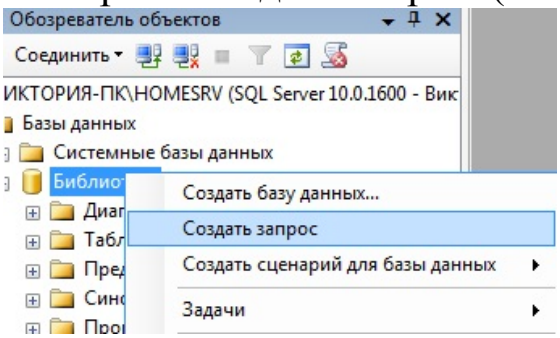



Рис. 31 - Порядок создания запроса

2. Введите следующий запрос и нажмите кнопку Выполнить.

```
CREATE TABLE KNIGI/*Создание таблицы KNIGI*/(
ID                    INT IDENTITY(1,1) CONSTRAINT
PK_KNIGI_ID PRIMARY KEY,  /* IDENTITY(1,1)-счетчик,
CONSTRAINT PK_KNIGI_ID PRIMARY KEY-задание первичного
ключа*/
Шифр_книги            INT,
Название              NVARCHAR(100),
Фамилия_автора        NVARCHAR(100),
Имя_автора            NVARCHAR(100),
Отчество_автора       NVARCHAR(100),
Жанр                  NVARCHAR(100),
Издательство          NVARCHAR(100),
```

```
Год_издания              INT,
Страницы                 INT,
Количество_экземпляров   INT,
Местонахождение          NVARCHAR(100),  )
CREATE TABLE CHITATEL/*Создание таблицы CHITATEL*/(
ID                       INT IDENTITY(1,1) CONSTRAINT
PK_CHITATEL_ID PRIMARY KEY,
ФИО_читателя   NVARCHAR(100),
Адрес                    NVARCHAR(100),
Дата_рождения   DATE,
Пол                      NVARCHAR(3)  CONSTRAINT
[CH_CHITATEL_ПОЛ] CHECK ([Пол] IN ('Муж', 'Жен')),
Место_рождения NVARCHAR(100))
CREATE TABLE  JOURNAL/*Создание таблицы JOURNAL*/(
ID                       INT IDENTITY(1,1) CONSTRAINT
PK_DVIGENIE_ID PRIMARY KEY,
Шифр_книги       int  CONSTRAINT FK_DVIGENIE_KNIGI
REFERENCES dbo.KNIGI(ID), /*Связь по вторичному
ключу.Столбец Шифр_книги заполняется значениями из столбца
ID таблицы KNIGI*/
ФИО_читателя     int  CONSTRAINT FK_DVIGENIE_CHITATEL
REFERENCES dbo.CHITATEL(ID),
Дата_выдачи      DATE,
Срок_сдачи       DATE)
```

3. Если всё введено верно внизу в окне появится фраза Выполнение команд успешно завершено.

Достоинство построения таблиц методом Transact-SQL состоит в том, что связи прописаны в коде, и при построении диаграммы базы данных достаточно просто выбрать нужные таблицы. (Рис. 32)

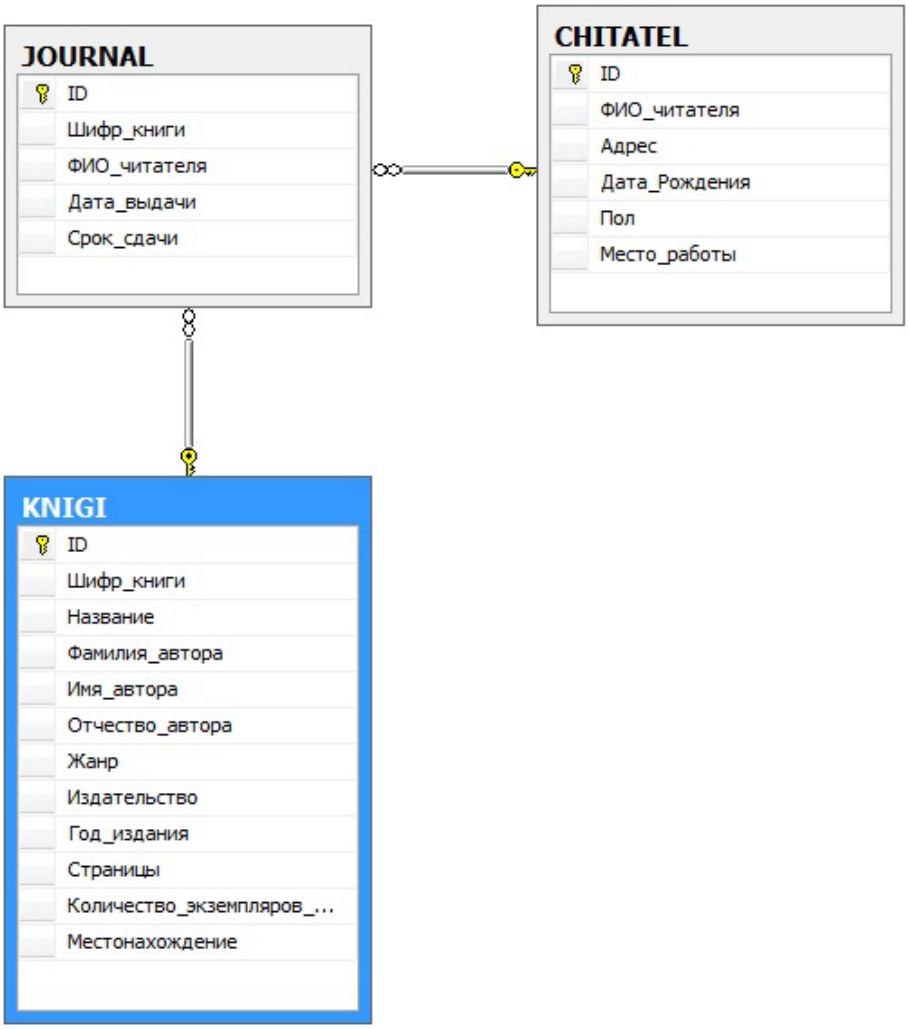



Рис. 32 - Диаграмма БД

В последующем мы будем пользоваться таблицами JOURNAL, KNIGI, CHITATEL.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Базовые определения.
2. Ограничения целостности. Основные типы ограничений целостности с простейшими примерами на языке SQL.
3. Этапы проектирования БД.
4. Команды и синтаксис языка SQL для создания базы данных и таблиц.
5. Домены и пользовательские типы данных (с примером на языке SQL).

Примечание: переход к защите теоретической части осуществляется после защиты практической части и наличии отчета, выполненного по форме, установленной преподавателем.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК


1. Уроки SQL и баз данных http://www.site-do.ru/db/db.php
2. Как создать базу данных в MS SQL Server http://npk-kaluga.ru/CreateBase_MSSQL.htm
3. Как создать новую таблицу в базе данных MS SQL Server http://npk-kaluga.ru/CreateTable_MSSQL.htm
4. Создание новой диаграммы базы данных (визуальные инструменты для баз данных) http://msdn.microsoft.com/ru-ru/library/ms189078.aspx
5. Создание базы данных http://technet.microsoft.com/ru-ru/library/ms186312(v=sql.110).aspx